ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

"**IMPLEMENTACIONES EN MATLAB DE LOS ALGORITMOS ADAPTATIVOS PARA LOS SISTEMAS DE ANTENAS INTELIGENTES**"

INFORME DE PROYECTO DE GRADO

Previa a la obtención del Título de:

INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

Presentado por:

JUAN FRANCISCO ALVAREZ ALVARADO

MARIBEL DEL ROSARIO CHUEZ GONZALEZ

GUAYAQUIL – ECUADOR

AÑO

2011

AGRADECIMIENTO

A Dios primeramente, porque él nos cuida, nos protege y nos ama infinitamente, de una manera que va mas allá de nuestro entendimiento, se que él me ha protegido y guiado toda mi vida, y espero estar siempre dentro de su misericordia.

A mi padres Pedro Manuel Alvarez Ghilardy y Maria Primitiva Alvarado Torres, que me han enseñado y guiado, y han sido padres maravillosos. A mis hermanos y hermanas por sus consejos y palabras de aliento.

Al ingeniero Pedro Vargas nuestro director de proyecto de graduación por su ayuda, consejos y guía, para hacer bien este proyecto.

JUAN FRANCISCO ALVAREZ ALVARADO

AGRADECIMIENTO

Quiero agradecer a primeramente a Dios, porque él me ha dado la oportunidad de estudiar, de superarme, también quiero agradecer a mi esposo y a mis niños por su comprensión y cariño en todo este tiempo elaborando la presente proyecto de graduación.

Al ingeniero Pedro Vargas nuestro director del proyecto de graduación por su ayuda, consejos y guía, por su paciencia con nosotros, gracias por su apoyo para hacer bien este proyecto.

A mis amigos(as) de la espol , a los profesores que con su entrega y sacrificio nos han hecho mejores personas con se conocimiento y consejos.

MARIBEL DEL ROSARIO CHUEZ GONZALEZ

DEDICATORIA

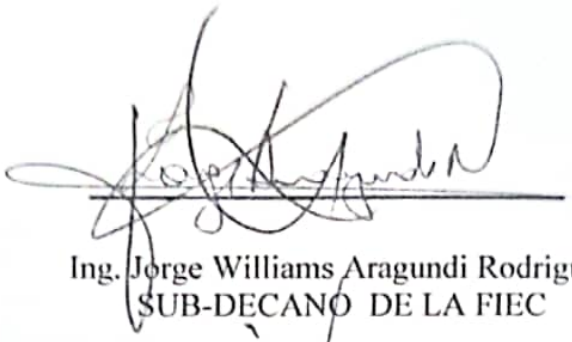
Dedico este trabajo a mis padres Pedro Manuel Alvarez Ghilardi y a Maria Primitiva Alvarado Torres de Alvarez

Juan Francisco Alvarez Alvarado

Dedico este trabajo a mis padres, a mi esposo German, a mis hijos Steven, Shirley y Gregorito quienes son lo más importante en mi vida.

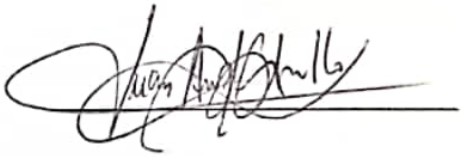
Maribel del Rosario Chuez Gonzalez

TRIBUNAL DE SUSTENTACIÓN



Ing. Jorge Williams Aragundi Rodriguez
SUB-DECANO DE LA FIEC

Ing. Pedro Vargas
DIRECTOR DE TESIS



Ing. Juan C. Avilés C.
VOCAL PRINCIPAL

Escaneado con CamScanner

DECLARACIÓN EXPRESA

"La responsabilidad del contenido de este Trabajo de Graduación, nos corresponde exclusivamente; y el patrimonio intelectual de la misma a la ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL"

(Reglamento de Graduación de la ESPOL)

Juan Francisco Alvarez Alvarado

Maribel del Rosario Chuez Gonzalez

Escaneado con CamScanner

RESUMEN

En el presente documento haremos una prueba de algoritmos adaptativos ya existentes utilizando el programa matlab.

Los algoritmos adaptativos pueden ser clasificados en categorías, basándose en distintas técnicas. Las técnicas basadas en información se clasifican en algoritmos basados en la señal de referencia y en algoritmos adaptativos ciegos.

Nuestro proyecto de graduación analiza los algoritmos adaptativos pero sobre todo los basados en técnicas de información como son los de señal de referencia y los algoritmos adaptativos ciegos. Dentro de los algoritmos adaptativos también hay dos grandes grupos, los que son llamados algoritmos adaptativos especiales y los algoritmos adaptativos generales. Se los considera de uso especial porque permiten obtener una convergencia más rápida del array de antenas. Mientras que los algoritmos generales tienen una convergencia más lenta que los especiales.

Los algoritmos basados en la señal de referencia se basan en la minimización del error mínimo cuadrado entre la señal recibida y la señal de referencia. Por lo tanto, se requiere que la señal de referencia esté disponible. La señal de referencia tiene una alta correlación con la señal deseada, por ejemplo, el algoritmo SMI (Matriz de inversión simple) , LMS (Least Mean square) y RLS (Recursive least square). La señal de referencia no es la verdadera señal deseada, de hecho es una señal que la representa de cerca o tiene una alta correlación con esta. La señal de referencia, requerida por el algoritmo, es generado de varias maneras.

Los algoritmos adaptativos ciegos no requieren señal de referencia, generan por si mismos la señal de referencia requerida desde las señales decepcionadas, para obtener la señal deseada. Por ejemplo, algoritmos CMA (modulo algorítmico constante), ciloestabilidad y decisión directa.

Los tipos de algoritmos adaptativos que analizamos en las simulaciones son: Algoritmo Mínimo cuadrado (LMS), Algoritmo Mínimo Cuadrado Recursivo (RLS), Algoritmo de modulo constante (CMA) y el algoritmo de Inversión de Matriz Directa (DMI).

En este documento se determina la capacidad de red, la cobertura y el desempeño de las antenas inteligentes utilizando algoritmos adaptativos

en implementaciones en matlab, en la actualidad ante el gran auge de las telecomunicaciones y el crecimiento continuo del número de usuarios de sistemas de comunicaciones móviles y la implementación de nuevas plataformas servicios móviles (3G) y el gran aumento de números de celulares, se ha presentado el problema de cómo aumentar la convergencia de los arreglos de antenas, y tener simulaciones adecuadas y próximas a la realidad que permitan simular correctamente la realidad del array de antenas. Esto se logra utilizando algoritmos adaptativos, los cuales permitirán calcular más rápidamente la convergencia de los pesos de las antenas, y una herramienta para hacer estas simulaciones es por medio del programa matlab.

ÍNDICE GENERAL

RESUMEN..... I

ÍNDICES

ÍNDICE GENERAL.....IV

ÍNDICE DE FIGURAS.....IX

ÍNDICE DE TABLAS.....XIV

ÍNDICE DE PANTALLAS.....XIX

ABREVIATURAS.....XXXVII

INTRODUCCIÓN

Descripción del Proyecto.....XLI

Objetivos XLIV

Organización del Proyecto.....XLVI

DESARROLLO

C A P Í T U L O 1

ANTENAS INTELIGENTES

1.1 Generalidades sobre Antenas Inteligentes 4

1.1.1 Definición de antenas inteligentes..... 7

1.1.2 Ventajas Potenciales de los sistemas de antenas..... 10

1.1.3 Problemas de las antenas inteligentes.....18

1.1.4 Funcionamiento de las Antenas Inteligentes.....21

1.1.5 Tipos de Alcance 23

1.2 Tipos de Antenas Inteligentes 26

1.3 Sistema de haz conmutado (SWITCHED BEAM) 27

1.4 Sistema de haz de seguimiento 35

1.5 Haz Adaptativo 39

C A P Í T U L O 2

ALGORITMOS ADAPTATIVOS

2.1 Algoritmos adaptativos basados en la señal de referencia..... 51

2.1.1 Concepto de Algoritmos adaptativos basados en la señal de referencia 51

2.1.2 Clasificación de los algoritmos basados en la señal de referencia…51

2.2 Algoritmos adaptativos ciegos 53

2.2.1 Concepto de Algoritmos adaptativos ciegos. 53

2.2.2 Clasificación de los algoritmos adaptativos ciegos……..…….55

2.3 Conformador de haces adaptativos..... 56

2.3.1 Estructura de los haces adaptativos……... 57

2.3.2 Los haces adaptativos en las antenas inteligentes…….. 67

2.3.3 Gráficos de haces adaptativos……... 70

2.3.3.1 Conformador de haz de referencia temporal (LMS) 70

2.3.3.2 Conformador de Haz de MSINR (algoritmo maximin)… ...72

2.3.3.3 Conformador de Haz de referencia espacial (cancelador de lóbulos laterales)……...78

2.3.3.4 Conversión a frecuencia Intermedia Posterior….....81

2.4 Señal de arribo 86

2.4.1 Descripción del sistema……... 87

2.4.2 Conformación de la señal de arribo……..... 91

2.4.3 Direccionamiento de la señal de arribo…….. 96

C A P Í T U L O 3

EL ALGORITMO ADAPTATIVO MÍNIMO CUADRADO (LMS)

3.1 El algoritmo LMS……..……..101

3.1.1 Base teórica del algoritmo adaptativo LMS …….103

3.1.2 Análisis matemático del algoritmo LMS 108

3.1.3 Variaciones con el algoritmo LMS 120

3.2 Filtro de Wiener 133

3.3 La solución de Wiener..... 138

3.4 Descripción de las diferentes partes del programa en matlab 140

3.5 Pantallas de visualización del algoritmo adaptativo mínimo cuadrado lms 150

3.6 Programa en matlab del algoritmo adaptativo mínimo cuadrado lms 152

3.7 Descripción del ejemplo a correr en el programa..... 161

3.8 Pruebas y resultados de las implementaciones LMS..... 161

C A P Í T U L O 4

EL ALGORITMO ADAPTATIVO MÍNIMO CUADRADO RECURSIVO (RLS)

4.1 El algoritmo RLS……...……198

4.1.1 Base teórica del algoritmo adaptativo RLS..... 202

4.1.2 Análisis matemático del algoritmo RLS 203

4.1.3 Variaciones con el algoritmo RLS 223

4.2 Filtro de Kalman……..... 230

4.3 Pantallas de visualización del algoritmo adaptativo mínimo cuadrado recursivo 234

4.4 Programa en matlab del algoritmo adaptativo mínimo cuadrado recursivo RLS.237

4.5 Descripción del ejemplo a correr en el programa..... …262

4.6 Pruebas y resultados de las implementaciones RLS…….……263

C A P Í T U L O 5

EL ALGORITMO ADAPTATIVO DE MÓDULO CONSTANTE (CMA)

5.1 El algoritmo CMA…….……..……..……..… 279

5.1.1 Base teórica del algoritmo adaptativo CMA 284

5.1.2 Análisis matemático del algoritmo CMA ..... .285

5.1.3 Variaciones con el algoritmo CMA 292

5.2 Pantallas de visualización del algoritmo adaptativo de módulo constante CMA

…….…….297

5.3 Programa en matlab del algoritmo adaptativo Algoritmo de modulo constante CMA …… 300

5.4 Descripción del ejemplo a correr en el programa..... 324

5.5 Pruebas y resultados de las implementaciones CMA..... 325

C A P I T U L O 6

EL ALGORITMO ADAPTATIVO INVERSIÓN DE MATRIZ DIRECTA (DMI)

6.1 El algoritmo DMI ……..……338

6.1.1 Base teórica del algoritmo adaptativo DMI 339

6.1.2 Análisis matemático del algoritmo DMI 340

6.1.3 Variaciones con el algoritmo DMI 346

6.2 Pantallas de visualización del algoritmo adaptativo inversión de matriz directa.348

6.3 Programa en matlab del algoritmo adaptativo DMI…….……..….350

6.4 Descripción del ejemplo a correr en el programa..... 360

6.5 Pruebas y resultados de las implementaciones DMI 361

6.6 Comparacion entre los resultados obtenidos por los algoritmo LMS, RLS, CMA y DMI……..……..371

CONCLUSIONES Y RECOMENDACIONES

ANEXOS

ANEXO A: Desarrollo de las ecuaciones del algoritmo LMS para valores discretos

ANEXO B: Desarrollo de la solucion de wiener

ANEXO C: El Programa Simulink y sus Principales Librerias

ANEXO D: Funciones más comunes utilizadas en matlab y en el presente proyecto de graduación

ANEXO E: Análisis del algoritmo rls para valores discretos

ANEXO F: Pantallas de visualización de las implementaciones en matlab

ANEXO G: Programas en matlab de los algoritmos LMS, RLS, CMA y DMI

problema1_lms

problema2_lms

problema3_lms

problema1_rls

problema2_rls

problema3_rls

problema1_cma

problema2_cma

problema3_cma

problema1_dmi

problema2_dmi

problema3_dmi

ANEXO H: Conceptos importantes sobre filtros adaptativos

REFERENCIAS BIBLIOGRÁFICAS

ÍNDICE DE FIGURAS

Figura 1.1 Antena Inteligente: D.S.P + Arreglo de antenas 4

Figura 1.2 Conformación de las antenas inteligentes 5

Figura 1.3 Diagrama de bloque de las antenas inteligentes..... 9

Figura 1.4 Prototipos de antena inteligente de la firma Allgon 10

Figura 1.5a Estación base de array de antenas : antena ensamblada 12

Figura 1.5b Estación base de array de antenas: equipo de recepción 12

Figura 1.6 Subzonas de acuerdo a la configuración de antenas..... 24

Figura 1.7 Lóbulo de radiación de una Antena Inteligente 25

Figura 1.8 Haz conmutado..... 27

Figura 1.9 Programa que controla el sistema de haz conmutado 29

Figura 1.10 Patron de radiación de un arreglo lineal uniforme de 7 elementos con un haz conmutado 29

Figura 1.11 Movimiento del haz por cambios de fase 30

Figura 1.12 Matriz de Butler de 8x8 alimentando a un arreglo lineal de 8 elementos 32

Figura 1.13 Matriz de Blass. Alimentador de haces multiples. 33

Figura 1.14 Acoplador direccional 33

Figura 1.15 Acoplador hibrido de 3 db y 90 grados 34

Figura 1.16 Swiches de microondas 34

Figura 1.17 Sistema de haz de seguimiento 35

Figura 1.18 Programa de control del sistema de haz de seguimiento 37

Figura 1.19 Patron de radiación de un arreglo lineal uniforme de 7 elementos: haz de seguimiento 38
Figura 1.20a Antena de haz conmutado 39
Figura 1.20b Antena de haz de seguimiento 39
Figura 1.21 Antena de haz adaptativo 40
Figura 1.22 Haz adaptativo 41
Figura 1.23 Programa de control del sistema de haz adaptativo 41
Figura 1.24 Patrón de radiación de un arreglo lineal uniforme de 7 elementos usando haz adaptativo 42
Figura 2.1 Situación real de comunicación con terminal móvil 49
Figura 2.2 Clasificación de los algoritmos adaptativos 55
Figura 2.3 Discriminación de señales semejantes 58
Figura 2.4 Lóbulo de radiación hacia la señal de interés. 59
Figura 2.5 Diagrama de bloques del receptor de la estación base y lazo de generación de la referencia 62
Figura 2.6 Sistema de arreglo adaptativo 63
Figura 2.7a Conformador de haz digital 66
Figura 2.7b Conformador de haz digital 66
Figura 2.8 Antena Adaptativa 67
Figura 2.9 Conformador de haz de referencia temporal con algoritmo LMS para señales de modulación F.H. 70
Figura 2.10 Conformador de haz de MSINR con algoritmo maximin para señales con modulación F.H 75

Figura 2.11 Conformador de haz de referencia espacial con cancelador de lobulos laterales para señales con modulación F.H (interferencias fijas)..... 78

Figura 2.12 Conformador de haz con etapa de conversión a frecuencia intermedia posterior para señales con modulación F.H. 82

Figura 2.13 Conformador de haz con Filtered – X Filtered – e LMS algorith para señales con modulación F.H. 84

Figura 2.14 Conformador de haz Master slave adaptative sidelobe canceller para señales con modulación F.H. 86

Figura 2.15 Arquitectura modular de la antena adaptativa con la que se lleva a cabo la información. Enlace ascendente. 89

Figura 2.16 Arquitectura modular de la antena adaptativa con la que se lleva a cabo la información. Enlace descendente 90

Figura 2.17 Esquemas de bloques de la antena inteligente 91

Figura 2.18 Esquema de bloque del conformador de enlace ascendente 93

Figura 2.19a Esquema de bloque del conformador de enlace descendente 95

Figura 2.19b Esquema de bloque del conformador de enlace descendente 97

Figura 2.19c Esquema de bloque del conformador de enlace descendente 97

Figura 2.19d Esquema de bloque del conformador de enlace descendente 98

Figura 3.1 Sistema de arreglo adaptativo 104

Figura 3.2 Ejemplo en matlab del ajuste de una curva.....112

Figura 3.3 Ejemplo de regresión lineal 114

Figura 3.4 Regresión lineal.....115

Figura 3.5 Determinación de dk.....117

Figura 3.6 Curva de aprendizaje para el algoritmo lms 132

Figura 3.7 Curva de aprendizaje para el algoritmo rls 132

Figura 3.8 Diagrama de bloques de el filtrado optimo.....135

Figura 3.9 Diagrama del filtro de wiener 136

Figura 4.1a Multisensor aplicacion.....205

Figura 4.1b Multisensor aplicacion.....205

Figura 4.2 Estimador lineal para un sistema de parámetro M.....206

Figura 4.3 Combinador lineal adaptativo 216

Figura 4.4 Representación de la superficie de error en matlab.....218

Figura 4.5 Representación de las curvas de nivel en matlab 219

Figura 4.6 Peso optimo con la función contour 222

Figura 4.7 Parámetros del filtro de kalman 233

Figura 5.1 Error cuadrático medio de la señal 64-QAM recuperada con un igualador cma 282

Figura 5.2 Diagrama de bloques del cancelador ciego cma-lms 283

Figura 5.3a Constelaciones contenidas por el combinador combinado cma-lms. Salida del componente cma. 284

Figura 5.3b Constelaciones contenidas por el combinador combinado cma-lms. Salida del sistema completo cma-lms……..284

Figura 5.4 Igualación SISO por el método de Shalvi - Weinstein 293

Figura 6.1 Arquitectura de un lazo de control del algoritmo DMI. 340

Figura 6.2 Multi entrada y multi salida de un sistema mimo G y K son 2 multi entradas y multi salidas mimo 340

Figura 6.3 Región de garantía de fase de un algoritmo dmi.....345

Figura 6.4 Diagrama eléctrico de una aplicación del algoritmo dmi de Blight, Daily and Gangass. 347

Figura A.1 Detalle de la implementación del LMS para el peso del filtro

Figura B.1 diagrama de un filtro adaptativo con una señal de entrada más ruido, con una señal de referencia

Figura C.1 Diagrama de Bloques

Figura C.2 Respuesta del sistema dinámico

Figura G.1 Filtro Adaptativo

Figura G.2 Funcion Gaussiana

Figura G.3 Forma tridimensional de una función gaussiana

ÍNDICE DE TABLAS

Tabla I Cuadro comparativo de algunos algoritmos adaptativos. 56

Tabla II Ajuste de una ecuación lineal por mínimos cuadrados.....111

Tabla III Muestra aleatoria variable x vs y.....113

Tabla IV El algoritmo NLMS 125

Tabla V El algoritmo VSLMS 127

Tabla VI Variantes del algoritmo LMS 129

Tabla VII Datos de entrada para el Problema3_lms_a……..……166

Tabla VIII Datos obtenidos por el ejemplo Problema3_lms_a 166

Tabla IX Datos de entrada para el Problema2_lms_a 171

Tabla X Datos obtenidos por el ejemplo Problema2_lms_a.……..…....172

Tabla XI Datos de entrada para el Problema2_lms_b……....…..……172

Tabla XII Datos obtenidos por el ejemplo Problema2_lms_b……..….173

Tabla XIII Datos de entrada para el Problema2_lms_c…….……..…174

Tabla XIV Datos obtenidos por el ejemplo Problema2_lms_c…….....….175

Tabla XV Datos de entrada para el Problema2_lms_d…….....……176

Tabla XVI Datos obtenidos por el ejemplo Problema2_lms_d…….....…176

Tabla XVII Datos de entrada para el Problema2_lms_e…….…….....…177

Tabla XVIII Datos obtenidos por el ejemplo Problema2_lms_e…….……..…178

Tabla XIX Datos de entrada para el Problema2_lms_f…….…..……179

Tabla XX Datos obtenidos por el ejemplo Problema2_lms_f…….……180

Tabla XXI Datos de entrada para el Problema2_lms_g…….……..…181

Tabla XXII Datos obtenidos por el ejemplo Problema2_lms_g……...….182

Tabla XXIII Datos de entrada para el Problema2_lms_h……..…….183

Tabla XXIV Datos obtenidos por el ejemplo Problema2_lms_h…….....…….183

Tabla XXV Datos de entrada para el Problema1_lms_a…….……..185

Tabla XXVI Datos obtenidos por el ejemplo Problema1_lms_a……..…….185

Tabla XXVII Datos de entrada para el Problema1_lms_b ……..…..….186

Tabla XXVIII Datos obtenidos por el ejemplo Problema1_lms_b…..…..…...…187

Tabla XXIX Datos de entrada para el Problema1_lms_c ……..188

Tabla XXX Datos obtenidos por el ejemplo Problema1_lms_c…….....…188

Tabla XXXI Datos de entrada para el Problema1_lms_d ……..……..190

Tabla XXXII Datos obtenidos por el ejemplo Problema1_lms_d…….....…..…191

Tabla XXXIII Datos de entrada para el Problema1_lms_e …….....…..192

Tabla XXXIV Datos obtenidos por el ejemplo Problema1_lms_e…….….....…193

Tabla XXXV Datos de entrada para el Problema1_lms_f …….……...…..194

Tabla XXXVI Datos obtenidos por el ejemplo Problema1_lms_f……...……..…195

Tabla XXXVII Pasos para resolver el filtro de kalman…….....……..234

Tabla XXXVIII Datos de entrada para el Problema1_rls_a……..…..……..265

Tabla XXXIX Datos obtenidos por el ejemplo Problema1_rls_a…….....…265

Tabla XL Datos de entrada para el Problema1_rls_b……..…..…...…..266

Tabla XLI Datos obtenidos por el ejemplo Problema1_rls_b…….....…267

Tabla XLII Datos de entrada para el Problema1_rls_c…….……..……..268

Tabla XLIII Datos obtenidos por el ejemplo Problema1_rls_c…….....…268

Tabla XLIV Datos de entrada para el Problema2_rls_a…….…..…..270

Tabla XLV Datos obtenidos por el ejemplo Problema2_rls_a…..….....…270

Tabla XLVI Datos de entrada para el Problema2_rls_b……..…….…..271

Tabla XLVII Datos obtenidos por el ejemplo Problema2_rls_b…….....…272

Tabla XLVIII Datos de entrada para el Problema3_rls_a……..…..273

Tabla XLIX Datos obtenidos por el ejemplo Problema3_rls_a…….….....…274

Tabla L Datos de entrada para el Problema3_rls_b……..……..275

Tabla LI Datos obtenidos por el ejemplo Problema3_rls_b…….....…275

Tabla LII Datos de entrada para el Problema3_cma_a……..……...328

Tabla LIII Datos obtenidos por el ejemplo Problema3_cma_a……...…....329

Tabla LIV Datos de entrada para el Problema3_cma_b…….……330

Tabla LV Datos obtenidos por el ejemplo Problema3_cma_b…….....…330

Tabla LVI Datos de entrada para el Problema2_cma_a……...……..…….333

Tabla LVII Datos obtenidos por el ejemplo Problema2_cma_a…….…...……334

Tabla LVIII Datos de entrada para el Problema1_cma_a…….…….335

Tabla LIX Datos obtenidos por el ejemplo Problema1_cma_a…….....…335

Tabla LX Datos de entrada para el Problema3_dmi_a……..…..…..361

Tabla LXI Datos obtenidos por el ejemplo Problema3_dmi_a…….…..…362

Tabla LXII Datos de entrada para el Problema3_dmi_b……..…...363

Tabla LXIII Datos obtenidos por el ejemplo Problema3_dmi_b…….……..363

Tabla LXIV Datos de entrada para el Problema3_dmi_c……...…...…365

Tabla LXV Datos obtenidos por el ejemplo Problema3_dmi_c….....…366

Tabla LXVI Datos de entrada para el Problema2_dmi_a…….…….367

Tabla LXVII Datos obtenidos por el ejemplo Problema2_dmi_a…….....…368

Tabla LXVIII Datos de entrada para el Problema1_dmi_a……...…….369

Tabla LXIX Datos obtenidos por el ejemplo Problema1_dmi_a…….....….370

Tabla LXX Datos de entrada para realizar la comparación de los resultados de los programas problema3_lms, problema3_rls, problema3_cma, Problema3_dmi (ejemplo problema3_lms_x, ejemplo problema3_rls _x ejemplo problema3_cma_x y ejemplo problema3_dmi_x)….372

Tabla LXXI Datos obtenidos por el (ejemplo problema3_lms_x, ejemplo problema3_rls_x, ejemplo problema3_cma_x y ejemplo problema3_dmi_ x) ……..…373

Tabla LXXII Datos de entrada para realizar la comparación de los resultados de los programas problema3_lms, problema3_rls, problema3_cma, Problema3_dmi (ejemplo problema3_lms_z, ejemplo problema3_rls _z ejemplo problema3_cma_z y ejemplo problema3_dmi_z)….374

Tabla LXXIII Datos obtenidos por el (ejemplo problema3_lms_z, ejemplo problema3_rls_z, ejemplo problema3_cma_z y ejemplo problema3_dmi_ z) ……..……375

Tabla LXXIV Datos de entrada para realizar .la comparación. de los resultados de los programas problema2_lms, problema2_rls, problema2_cma, Problema2_dmi (ejemplo problema2_lms_x,ejemplo problema2_rls _x ejemplo problema2_cma_x y ejemplo problema2_dmi_x)….377

Tabla LXXV Datos obtenidos por el (ejemplo problema2_lms_x, ejemplo problema2_rls_x, ejemplo problema2_cma_x y ejemplo problema2_dmi_ x) ……378

Tabla LXXVI Datos de entrada para realizar la comparación de los resultados de los programas problema1_lms, problema1_rls, problema1_cma, Problema1_dmi (ejemplo problema1_lms_x, ejemplo problema1_rls _x, ejemplo problema1_cma_x y ejemplo problema1_dmi_x).…380

Tabla LXXVII Datos obtenidos por el (ejemplo problema1_lms_x, ejemplo problemal_rls_x, ejemplo problema1_cma_x y ejemplo problema1_dmi_ x)……...……...……381

Tabla LXXVIII Datos de entrada para realizar la comparación de los resultados de los programas problema1_lms, problema1_rls, problema1_cma, Problema1_dmi (ejemplo problema1_lms_z, ejemplo problema1_rls _z, ejemplo problema1_cma_z y ejemplo problema1_dmi_z).…382

Tabla LXXIX Datos obtenidos por el (ejemplo problema1_lms_z, ejemplo problema1_rls_z, ejemplo problema1_cma_z y ejemplo problema1_dmi_ z) ……..…….……384

ÍNDICE DE PANTALLAS

Pantalla 3.1 Problema3_lms_a ……439

Pantalla 3.2 Datos del programa problema3_lms_a 440

Pantalla 3.3 Potencias obtenidas con el programa problema3_lms_a.....441

Pantalla 3.4 Ejemplo problema3_lms_a. Comparación de los pesos estimados 442

Pantalla 3.5 Ejemplo problema3_lms_a. Curva de error de una señal senosoidal en un algoritmo lms 443

Pantalla 3.6 Ejemplo problema3_lms_a. Comparación entre la salida del sistema de una señal senosoidal (lms) con la señal de referencia del sistema 444

Pantalla 3.7 Problema2_lms_a 445

Pantalla 3.8 Datos del programa problema2_lms_a 446

Pantalla 3.9 Potencias obtenidas con el programa problema2_lms_a.....447

Pantalla 3.10 Ejemplo problema2_lms_a. Comparación de los pesos estimados 448

Pantalla 3.11 Ejemplo problema2_lms_a. Curva de error de una señal gaussiana en un algoritmo lms 449

Pantalla 3.12 Ejemplo problema2_lms_a. Comparación entre la salida del sistema de una señal gaussiana (lms) con la señal de referencia del sistema 450

Pantalla 3.13 Problema2_lms_b.....451

Pantalla 3.14 Datos del programa problema2_lms_b.....452

Pantalla 3.15 Potencias obtenidas con el programa problema2_lms_b.....453

Pantalla 3.16 Ejemplo problema2_lms_b. Comparación de los pesos estimados.....454

Pantalla 3.17 Ejemplo problema2_lms_b. Curva de error de una señal gaussiana en un algoritmo lms 455
Pantalla 3.18 Ejemplo problema2_lms_b. Comparación entre la salida del sistema de una señal gaussiana (lms) con la señal de referencia del sistema 456
Pantalla 3.19 Problema2_lms_c 457
Pantalla 3.20 Datos del programa problema2_lms_c 458
Pantalla 3.21 Potencias obtenidas con el programa problema2_lms_c 459
Pantalla 3.22 Ejemplo problema2_lms_c. Comparación de los pesos estimados 460
Pantalla 3.23 Ejemplo problema2_lms_c. Curva de error de una señal gaussiana en un algoritmo lms 461
Pantalla 3.24 Ejemplo problema2_lms_c. Comparación entre la salida del sistema de una señal gaussiana (lms) con la señal de referencia del sistema 462
Pantalla 3.25 Problema2_lms_d 463
Pantalla 3.26 Datos del programa problema2_lms_d 464
Pantalla 3.27 Potencias obtenidas con el programa problema2_lms_d 465
Pantalla 3.28 Ejemplo problema2_lms_d. Comparación de los pesos estimados 466
Pantalla 3.29 Ejemplo problema2_lms_d. Curva de error de una señal gaussiana en un algoritmo lms 467
Pantalla 3.30 Ejemplo problema2_lms_d. Comparación entre la salida del sistema de una señal gaussiana (lms) con la señal de referencia del sistema 468
Pantalla 3.31 Problema2_lms_e 469
Pantalla 3.32 Datos del programa problema2_lms_e 470
Pantalla 3.33 Potencias obtenidas con el programa problema2_lms_e 471

Pantalla 3.34 Ejemplo problema2_lms_e. Comparación de los pesos estimados 472
Pantalla 3.35 Ejemplo problema2_lms_e. Curva de error de una señal gaussiana en un algoritmo lms 473
Pantalla 3.36 Ejemplo problema2_lms_e. Comparación entre la salida del sistema de una señal gaussiana (lms) con la señal de referencia del sistema 474
Pantalla 3.37 Problema2_lms_f 475
Pantalla 3.38 Datos del programa problema2_lms_f 476
Pantalla 3.39 Potencias obtenidas con el programa problema2_lms_f 477
Pantalla 3.40 Ejemplo problema2_lms_f. Comparación de los pesos estimados.....478
Pantalla 3.41 Ejemplo problema2_lms_f. Curva de error de una señal gaussiana en un algoritmo lms 479
Pantalla 3.42 Ejemplo problema2_lms_f. Comparación entre la salida del sistema de una señal gaussiana (lms) con la señal de referencia del sistema 480
Pantalla 3.43 Problema2_lms_g 481
Pantalla 3.44 Datos del programa problema2_lms_g.....482
Pantalla 3.45 Potencias obtenidas con el programa problema2_lms_g.....483
Pantalla 3.46 Ejemplo problema2_lms_g. Comparación de los pesos estimados 484
Pantalla 3.47 Ejemplo problema2_lms_g. Curva de error de una señal gaussiana en un algoritmo lms 485
Pantalla 3.48 Ejemplo problema2_lms_g. Comparación entre la salida del sistema de una señal gaussiana (lms) con la señal de referencia del sistema 486
Pantalla 3.49 Problema2_lms_h 487
Pantalla 3.50 Datos del programa problema2_lms_h.....488

Pantalla 3.51 Potencias obtenidas con el programa problema2_lms_h.....489

Pantalla 3.52 Ejemplo problema2_lms_h. Comparación de los pesos estimados.....490

Pantalla 3.53 Ejemplo problema2_lms_h. Curva de error de una señal gaussiana en un algoritmo lms 492

Pantalla 3.54 Ejemplo problema2_lms_h. Comparación entre la salida del sistema de una señal gaussiana (lms) con la señal de referencia del sistema 493

Pantalla 3.55 Problema1_lms_a 494

Pantalla 3.56 Datos del programa problema1_lms_a.....495

Pantalla 3.57 Potencias obtenidas con el programa problema1_lms_a.....496

Pantalla 3.58 Ejemplo problema1_lms_a. Comparación de los pesos estimados.....497

Pantalla 3.59 Ejemplo problema1_lms_a. Curva de error de una señal senosoidal en un algoritmo lms 498

Pantalla 3.60 Ejemplo problema1_lms_a. Comparación entre la salida del sistema de una señal senosoidal (lms) con la señal de referencia del sistema 499

Pantalla 3.61 Problema1_lms_b 500

Pantalla 3.62 Datos del programa problema1_lms_b.....501

Pantalla 3.63 Potencias obtenidas con el programa problema1_lms_b.....502

Pantalla 3.64 Ejemplo problema1_lms_b. Comparación de los pesos estimados.....503

Pantalla 3.65 Ejemplo problema1_lms_b. Curva de error de una señal senosoidal en un algoritmo lms 504

Pantalla 3.66 Ejemplo problema1_lms_b. Comparación entre la salida del sistema de una señal senosoidal (lms) con la señal de referencia del sistema 505

Pantalla 3.67 Problema1_lms_c 506

Pantalla 3.68 Datos del programa problema1_lms_c 507
Pantalla 3.69 Potencias obtenidas con el programa problema1_lms_c 508
Pantalla 3.70 Ejemplo problema1_lms_c. Comparación de los pesos estimados 509
Pantalla 3.71 Ejemplo problema1_lms_c. Curva de error de una señal senosoidal en un algoritmo lms 510
Pantalla 3.72 Ejemplo problema1_lms_c. Comparación entre la salida del sistema de una señal senosoidal (lms) con la señal de referencia del sistema 511
Pantalla 3.73 Problema1_lms_d 512
Pantalla 3.74 Datos del programa problema1_lms_d 513
Pantalla 3.75 Potencias obtenidas con el programa problema1_lms_d 514
Pantalla 3.76 Ejemplo problema1_lms_d. Comparación de los pesos estimados 515
Pantalla 3.77 Ejemplo problema1_lms_d. Curva de error de una señal senosoidal en un algoritmo lms 516
Pantalla 3.78 Ejemplo problema1_lms_d. Comparación entre la salida del sistema de una señal senosoidal (lms) con la señal de referencia del sistema 517
Pantalla 3.79 Problema1_lms_e 518
Pantalla 3.80 Datos del programa problema1_lms_e 519
Pantalla 3.81 Potencias obtenidas con el programa problema1_lms_e 520
Pantalla 3.82 Ejemplo problema1_lms_e. Comparación de los pesos estimados 521
Pantalla 3.83 Ejemplo problema1_lms_e. Curva de error de una señal senosoidal en un algoritmo lms 522
Pantalla 3.84 Ejemplo problema1_lms_e. Comparación entre la salida del sistema de una señal senosoidal (lms) con la señal de referencia del sistema 523

Pantalla 3.85 Problema1_lms_f 524

Pantalla 3.86 Datos del programa problema1_lms_f 525

Pantalla 3.87 Potencias obtenidas con el programa problema1_lms_f 526

Pantalla 3.88 Ejemplo problema1_lms_f. Comparación de los pesos estimados.....527

Pantalla 3.89 Ejemplo problema1_lms_f. Curva de error de una señal senosoidal en un algoritmo lms 528

Pantalla 3.90 Ejemplo problema1_lms_f. Comparación entre la salida del sistema de una señal senosoidal (lms) con la señal de referencia del sistema 529

Pantalla 4.a Obtención de los pesos óptimos wo_min y w1_min……260

Pantalla 4.1 Problema1_rls_a 530

Pantalla 4.2 Datos del programa problema1_rls_a.....531

Pantalla 4.3 Potencias obtenidas con el programa problema1_rls_a.....532

Pantalla 4.4 Ejemplo problema1_rls_a. Comparación de los pesos estimados.....533

Pantalla 4.5 Ejemplo problema1_rls_a. Convergencia de los pesos estimados 534

Pantalla 4.6 Ejemplo problema1_rls_a. Curva de error de una señal senosoidal en un algoritmo rls 536

Pantalla 4.7 Ejemplo problema1_rls_a. Comparación entre la salida del sistema de una señal senosoidal (rls) con la señal de referencia del sistema 537

Pantalla 4.8 Problema1_rls_b 538

Pantalla 4.9 Datos del programa problema1_rls_b 539

Pantalla 4.10 Potencias obtenidas con el programa problema1_rls_b 540

Pantalla 4.11 Ejemplo problema1_rls_b. Comparación de los pesos estimados.....541

Pantalla 4.12 Ejemplo problema1_rls_b. Convergencia de los pesos estimados.....542

Pantalla 4.13 Ejemplo problema1_rls_b. Curva de error de una señal senosoidal en un algoritmo rls 543

Pantalla 4.14 Ejemplo problema1_rls_b. Comparación entre la salida del sistema de una señal senosoidal (rls) con la señal de referencia del sistema 544

Pantalla 4.15 Problema1_rls_c 545

Pantalla 4.16 Datos del programa problema1_rls_c.....546

Pantalla 4.17 Potencias obtenidas con el programa problema1_rls_c.....547

Pantalla 4.18 Ejemplo problema1_rls_c. Comparación de los pesos estimados.....548

Pantalla 4.19 Ejemplo problema1_rls_c. Convergencia de los pesos estimados 549

Pantalla 4.20 Ejemplo problema1_rls_c. Curva de error de una señal senosoidal en un algoritmo rls 550

Pantalla 4.21 Ejemplo problema1_rls_c. Comparación entre la salida del sistema de una señal senosoidal (rls) con la señal de referencia del sistema 551

Pantalla 4.22 Problema2_rls_a 552

Pantalla 4.23 Datos del programa problema2_rls_a.....553

Pantalla 4.24 Potencias obtenidas con el programa problema2_rls_a.....554

Pantalla 4.25 Ejemplo problema2_rls_a. Comparación de los pesos estimados.....555

Pantalla 4.26 Ejemplo problema2_rls_a. Convergencia de los pesos estimados 556

Pantalla 4.27 Ejemplo problema2_rls_a. Curva de error de una señal gaussiana en un algoritmo rls 557

Pantalla 4.28 Ejemplo problema2_rls_a. Comparación entre la salida del sistema de una señal gaussiana (rls) con la señal de referencia del sistema 558

Pantalla 4.29 Problema2_rls_b 559

Pantalla 4.30 Datos del programa problema2_rls_b 560

Pantalla 4.31 Potencias obtenidas con el programa problema2_rls_b 561

Pantalla 4.32 Ejemplo problema2_rls_b. Comparación de los pesos estimados.....562

Pantalla 4.33 Ejemplo problema2_rls_b. Convergencia de los pesos estimados.....563

Pantalla 4.34 Ejemplo problema2_rls_b. Curva de error de una señal gaussiana en un algoritmo rls 564

Pantalla 4.35 Ejemplo problema2_rls_b. Comparación entre la salida del sistema de una señal gaussiana (rls) con la señal de referencia del sistema 565

Pantalla 4.36 Problema3_rls_a 566

Pantalla 4.37 Datos del programa problema3_rls_a.....567

Pantalla 4.38 Potencias obtenidas con el programa problema3_rls_a.....568

Pantalla 4.39 Ejemplo problema3_rls_a. Comparación de los pesos estimados.....569

Pantalla 4.40 Ejemplo problema3_rls_a. Convergencia de los pesos estimados 571

Pantalla 4.41 Ejemplo problema3_rls_a. Curva de error de una señal senosoidal en un algoritmo rls 572

Pantalla 4.42 Ejemplo problema3_rls_a. Comparación entre la salida del sistema de una señal senosoidal (rls) con la señal de referencia del sistema 573

Pantalla 4.43 Problema3_rls_b 574

Pantalla 4.44 Datos del programa problema3_rls_b 575

Pantalla 4.45 Potencias obtenidas con el programa problema3_rls_b 576

Pantalla 4.46 Ejemplo problema3_rls_b. Comparación de los pesos estimados.....577

Pantalla 4.47 Ejemplo problema3_rls_b. Convergencia de los pesos estimados.....578

Pantalla 4.48 Ejemplo problema3_rls_b. Curva de error de una señal senosoidal en un algoritmo rls 579

Pantalla 4.49 Ejemplo problema3_rls_b. Comparación entre la salida del sistema de una señal senosoidal (rls) con la señal de referencia del sistema 580

Pantalla 5.1 Problema3_cma_a 581

Pantalla 5.2 Datos del programa problema3_cma_a 582

Pantalla 5.3 Potencias obtenidas con el programa problema3_cma_a 583

Pantalla 5.4 Ejemplo problema3_cma_a. Comparación de los pesos estimados 584

Pantalla 5.5 Ejemplo problema3_cma_a. Comparación entre la salida del sistema de una señal senosoidal (cma) con la señal de referencia del sistema 585

Pantalla 5.6 Ejemplo problema3_cma_a. Convergencia de los pesos estimados 586

Pantalla 5.7 Ejemplo problema3_cma_a. Curva de error de una señal senosoidal en un algoritmo cma 587

Pantalla 5.8 Problema3_cma_b 588

Pantalla 5.9 Datos del programa problema3_cma_b.....589

Pantalla 5.10 Potencias obtenidas con el programa problema3_cma_b.....590

Pantalla 5.11 Ejemplo problema3_cma_b. Comparación de los pesos estimados 591

Pantalla 5.12 Ejemplo problema3_cma_b. Comparación entre la salida del sistema de una señal senosoidal (cma) con la señal de referencia del sistema utilizando opción_señal=2 592

Pantalla 5.13 Ejemplo problema3_cma_b. Comparación entre la salida del sistema de una señal senosoidal (cma) con la señal de referencia del sistema utilizando opción_señal=1 594

Pantalla 5.14 Ejemplo problema3_cma_b. Curva de error de una señal senosoidal en un algoritmo cma.....595
Pantalla 5.15 Ejemplo problema3_cma_a. Convergencia de los pesos estimados ..596
Pantalla 5.16 Problema2_cma_a 597
Pantalla 5.17 Datos del programa problema2_cma_a 598
Pantalla 5.18 Potencias obtenidas con el programa problema2_cma_a 599
Pantalla 5.19 Ejemplo problema2_cma_a. Comparación de los pesos estimados 600
Pantalla 5.20 Ejemplo problema2_cma_a. Convergencia de los pesos estimados ..601
Pantalla 5.21 Ejemplo problema2_cma_a. Curva de error de una señal gaussiana en un algoritmo cma.....602
Pantalla 5.22 Ejemplo problema2_cma_a. Comparación entre la salida del sistema de una señal gaussiana (cma) con la señal de referencia del sistema, opción_ruido=1 603
Pantalla 5.23 Problema1_cma_a 604
Pantalla 5.24 Datos del programa problema1_cma_a 605
Pantalla 5.25 Potencias obtenidas con el programa problema1_cma_a 606
Pantalla 5.26 Ejemplo problema1_cma_a. Comparación de los pesos estimados 607
Pantalla 5.27 Ejemplo problema1_cma_a. Comparación entre la salida del sistema de una señal senosoidal (cma) con la señal de referencia del sistema 608
Pantalla 5.28 Ejemplo problema1_cma_a. Convergencia de los pesos estimados ..609
Pantalla 5.29 Ejemplo problema1_cma_a. Curva de error de una señal senosoidal en un algoritmo cma.....610

Pantalla 6.1 Problema3_dmi_a 612

Pantalla 6.2 Datos del programa problema3_dmi_a.....613

Pantalla 6.3 Potencias obtenidas con el programa problema3_dmi_a 614

Pantalla 6.4 Ejemplo problema3_dmi_a. Comparación de los pesos estimados.....615

Pantalla 6.5 Ejemplo problema3_dmi_a. Comparación entre la salida del sistema de una señal senosoidal (dmi) con la señal de referencia del sistema 616

Pantalla 6.6 Ejemplo problema3_dmi_a. Curva de error de una señal senosoidal en un algoritmo dmi.....617

Pantalla 6.7 Problema3_dmi_b 618

Pantalla 6.8 Datos del programa problema3_dmi_b 619

Pantalla 6.9 Potencias obtenidas con el programa problema3_dmi_b 620

Pantalla 6.10 Ejemplo problema3_dmi_b. Comparación de los pesos estimados 621

Pantalla 6.11 Ejemplo problema3_dmi_b. Comparación entre la salida del sistema de una señal senosoidal (dmi) con la señal de referencia del sistema 622

Pantalla 6.12 Ejemplo problema3_dmi_b. Curva de error de una señal senosoidal en un algoritmo dmi 623

Pantalla 6.13 Problema3_dmi_c 624

Pantalla 6.14 Datos del programa problema3_dmi_c.....625

Pantalla 6.15 Potencias obtenidas con el programa problema3_dmi_c 626

Pantalla 6.16 Ejemplo problema3_dmi_c. Comparación de los pesos estimados.....627

Pantalla 6.17 Ejemplo problema3_dmi_c. Comparación entre la salida del sistema de una señal senosoidal (dmi) con la señal de referencia del sistema 628

Pantalla 6.18 Ejemplo problema3_dmi_b. Curva de error de una señal senosoidal en un algoritmo dmi 629
Pantalla 6.19 Problema2_dmi_a 630
Pantalla 6.20 Datos del programa problema2_dmi_a.....631
Pantalla 6.21 Potencias obtenidas con el programa problema2_dmi_a 632
Pantalla 6.22 Ejemplo problema2_dmi_a. Comparación de los pesos estimados.....633
Pantalla 6.23 Ejemplo problema2_dmi_a. Comparación entre la salida del sistema de una señal gaussiana (dmi) con la señal de referencia del sistema.....634
Pantalla 6.24 Ejemplo problema2_dmi_a. Curva de error de una señal gaussiana en un algoritmo dmi 635
Pantalla 6.25 Problema1_dmi_a 636
Pantalla 6.26 Datos del programa problema1_dmi_a.....637
Pantalla 6.27 Potencias obtenidas con el programa problema1_dmi_a 637
Pantalla 6.28 Ejemplo problema1_dmi_a. Comparación de los pesos estimados.....638
Pantalla 6.29 Ejemplo problema1_dmi_a. Curva de error de una señal senosoidal en un algoritmo dmi 639
Pantalla 6.30 Ejemplo problema1_dmi_a. Comparación entre la salida del sistema de una señal senosoidal (dmi) con la señal de referencia del sistema 640
Pantalla 6.31 Datos del programa problema3_lms_x, problema3_rls_x, problema3_cma_x y problema3_dmi_x 641
Pantalla 6.32 Potencias obtenidas con el programa problema3_lms_x.....642
Pantalla 6.33 Potencias obtenidas con el programa problema3_rls_x 642
Pantalla 6.34 Potencias obtenidas con el programa problema3_cma_x.....643

Pantalla 6.35 Potencias obtenidas con el programa problema3_dmi_x 643

Pantalla 6.36 Ejemplo problema3_lms_x. Comparación de los pesos estimados.....645

Pantalla 6.37 Ejemplo problema3_rls_x. Comparación de los pesos estimados.....646

Pantalla 6.38 Ejemplo problema3_cma_x. Comparación de los pesos estimados 647

Pantalla 6.39 Ejemplo problema3_dmi_x. Comparación de los pesos estimados 648

Pantalla 6.40 Ejemplo problema3_lms_x. Comparación entre la salida del sistema de una señal senosoidal (lms) con la señal de referencia del sistema 649

Pantalla 6.41 Ejemplo problema3_rls_x. Comparación entre la salida del sistema de una señal senosoidal (rls) con la señal de referencia del sistema.....650

Pantalla 6.42 Ejemplo problema3_cma_x. Comparación entre la salida del sistema de una señal senosoidal (cma) con la señal de referencia del sistema 651

Pantalla 6.43 Ejemplo problema3_dmi_a. Comparación entre la salida del sistema de una señal senosoidal (dmi) con la señal de referencia del sistema 652

Pantalla 6.44 Ejemplo problema3_lms_x. Curva de error de una señal senosoidal en un algoritmo lms 653

Pantalla 6.45 Ejemplo problema3_rls_x. Curva de error de una señal senosoidal en un algoritmo rls 653

Pantalla 6.46 Ejemplo problema3_cma_x. Curva de error de una señal senosoidal en un algoritmo cma.....654

Pantalla 6.47 Ejemplo problema3_dmi_x. Curva de error de una señal senosoidal en un algoritmo dmi 654

Pantalla 6.48 Ejemplo problema3_rls_x. Convergencia de los pesos estimados.....655

Pantalla 6.49 Datos del programa problema3_lms_z, problema3_rls_z, problema3_cma_z y problema3_dmi_z.....656
Pantalla 6.50 Potencias obtenidas con el programa problema3_lms_z.....657
Pantalla 6.51 Potencias obtenidas con el programa problema3_rls_z.....657
Pantalla 6.52 Potencias obtenidas con el programa problema3_cma_z 658
Pantalla 6.53 Potencias obtenidas con el programa problema3_dmi_z 658
Pantalla 6.54 Ejemplo problema3_lms_z. Comparación de los pesos estimados 660
Pantalla 6.55 Ejemplo problema3_rls_z. Comparación de los pesos estimados.....661
Pantalla 6.56 Ejemplo problema3_cma_z. Comparación de los pesos estimados 662
Pantalla 6.57 Ejemplo problema3_dmi_z. Comparación de los pesos estimados.....663
Pantalla 6.58 Ejemplo problema3_lms_z. Comparación entre la salida del sistema de una señal senosoidal (lms) con la señal de referencia del sistema.....664
Pantalla 6.59 Ejemplo problema3_rls_z. Comparación entre la salida del sistema de una señal senosoidal (rls) con la señal de referencia del sistema.....665
Pantalla 6.60 Ejemplo problema3_cma_z. Comparación entre la salida del sistema de una señal senosoidal (cma) con la señal de referencia del sistema 666
Pantalla 6.61 Ejemplo problema3_dmi_z. Comparación entre la salida del sistema de una señal senosoidal (dmi) con la señal de referencia del sistema 667
Pantalla 6.62 Ejemplo problema3_lms_z. Curva de error de una señal senosoidal en un algoritmo lms 668
Pantalla 6.63 Ejemplo problema3_rls_z. Curva de error de una señal senosoidal en un algoritmo rls 669

Pantalla 6.64 Ejemplo problema3_cma_z. Curva de error de una señal senosoidal en un algoritmo cma.....670

Pantalla 6.65 Ejemplo problema3_dmi_z. Curva de error de una señal senosoidal en un algoritmo dmi 671

Pantalla 6.66 Ejemplo problema3_rls_z. Convergencia de los pesos estimados 672

Pantalla 6.67 Datos del programa problema2_lms_x, problema2_rls_x, problema2_cma_x y problema2_dmi_x 673

Pantalla 6.68 Potencias obtenidas con el programa problema2_lms_z.....674

Pantalla 6.69 Potencias obtenidas con el programa problema2_rls_z.....674

Pantalla 6.70 Potencias obtenidas con el programa problema2_cma_z 675

Pantalla 6.71 Potencias obtenidas con el programa problema2_dmi_z 675

Pantalla 6.72 Ejemplo problema2_lms_z. Comparación de los pesos estimados 677

Pantalla 6.73 Ejemplo problema2_rls_z. Comparación de los pesos estimados.....678

Pantalla 6.74 Ejemplo problema2_cma_z. Comparación de los pesos estimados 679

Pantalla 6.75 Ejemplo problema2_dmi_z. Comparación de los pesos estimados.....680

Pantalla 6.76 Ejemplo problema2_lms_z. Comparación entre la salida del sistema de una señal gaussiana (lms) con la señal de referencia del sistema 681

Pantalla 6.77 Ejemplo problema2_rls_z. Comparación entre la salida del sistema de una señal gaussiana (rls) con la señal de referencia del sistema.....681

Pantalla 6.78 Ejemplo problema2_cma_z. Comparación entre la salida del sistema de una señal gaussiana (cma) con la señal de referencia del sistema 682

Pantalla 6.79 Ejemplo problema2_dmi_z. Comparación entre la salida del sistema de una señal gaussiana (dmi) con la señal de referencia del sistema.....682

Pantalla 6.80 Ejemplo problema2_lms_z. Curva de error de una señal gaussiana en un algoritmo lms 683

Pantalla 6.81 Ejemplo problema2_rls_z. Curva de error de una señal gaussiana en un algoritmo rls 684

Pantalla 6.82 Ejemplo problema2_cma_z. Curva de error de una señal gaussiana en un algoritmo cma.....685

Pantalla 6.83 Ejemplo problema2_dmi_z. Curva de error de una señal gaussiana en un algoritmo dmi 685

Pantalla 6.84 Ejemplo problema2_rls_z. Convergencia de los pesos estimados 686

Pantalla 6.85 Datos del programa problema1_lms_x, problema1_rls_x, problema1_cma_x y problema1_dmi_x 687

Pantalla 6.86 Potencias obtenidas con el programa problema1_lms_x.....688

Pantalla 6.87 Potencias obtenidas con el programa problema1_rls_x 688

Pantalla 6.88 Potencias obtenidas con el programa problema1_cma_x.....689

Pantalla 6.89 Potencias obtenidas con el programa problema1_dmi_x 689

Pantalla 6.90 Ejemplo problema1_lms_x. Comparación de los pesos estimados.....691

Pantalla 6.91 Ejemplo problema1_rls_x. Comparación de los pesos estimados.....692

Pantalla 6.92 Ejemplo problema1_cma_x. Comparación de los pesos estimados 693

Pantalla 6.93 Ejemplo problema1_dmi_x. Comparación de los pesos estimados 694

Pantalla 6.94 Ejemplo problema1_lms_x. Comparación entre la salida del sistema de una señal senosoidal (lms) con la señal de referencia del sistema 695

Pantalla 6.95 Ejemplo problema1_rls_x. Comparación entre la salida del sistema de una señal senosoidal (rls) con la señal de referencia del sistema.....696

Pantalla 6.96 Ejemplo problema1_cma_x. Comparación entre la salida del sistema de una señal senosoidal (cma) con la señal de referencia del sistema 697

Pantalla 6.97 Ejemplo problema1_dmi_x. Comparación entre la salida del sistema de una señal senosoidal (dmi) con la señal de referencia del sistema 698

Pantalla 6.98 Ejemplo problema1_lms_x. Curva de error de una señal senosoidal en un algoritmo lms 699

Pantalla 6.99 Ejemplo problema1_rls_x. Curva de error de una señal senosoidal en un algoritmo rls 699

Pantalla 6.100 Ejemplo problema1_cma_x. Curva de error de una señal senosoidal en un algoritmo cma.....700

Pantalla 6.101 Ejemplo problema1_dmi_x. Curva de error de una señal senosoidal en un algoritmo dmi 700

Pantalla 6.102 Ejemplo problema1_rls_x. Convergencia de los pesos estimados ...701

Pantalla 6.103 Datos del programa problema1_lms_z, problema1_rls_z, problema1_cma_z y problema1_dmi_z.....702

Pantalla 6.104 Potencias obtenidas con el programa problema1_lms_z 703

Pantalla 6.105 Potencias obtenidas con el programa problema1_rls_z 703

Pantalla 6.106 Potencias obtenidas con el programa problema1_cma_z 704

Pantalla 6.107 Potencias obtenidas con el programa problema1_dmi_z 704

Pantalla 6.108 Ejemplo problema1_lms_z. Comparación de los pesos estimados ...706

Pantalla 6.109 Ejemplo problema1_rls_z. Comparación de los pesos estimados 707

Pantalla 6.110 Ejemplo problema1_cma_z. Comparación de los pesos estimados ..708

Pantalla 6.111 Ejemplo problema1_dmi_z. Comparación de los pesos estimados...709

Pantalla 6.112 Ejemplo problema1_lms_z. Comparación entre la salida del sistema de una señal senosoidal (lms) con la señal de referencia del sistema 710

Pantalla 6.113 Ejemplo problema1_rls_z. Comparación entre la salida del sistema de una señal senosoidal (rls) con la señal de referencia del sistema.....711

Pantalla 6.114 Ejemplo problema1_cma_z. Comparación entre la salida del sistema de una señal senosoidal (cma) con la señal de referencia del sistema .712

Pantalla 6.115 Ejemplo problema1_dmi_z. Comparación entre la salida del sistema de una señal senosoidal (dmi) con la señal de referencia del sistema ..713

Pantalla 6.116 Ejemplo problema1_lms_z. Curva de error de una señal senosoidal en un algoritmo lms 714

Pantalla 6.117 Ejemplo problema1_rls_z. Curva de error de una señal senosoidal en un algoritmo rls 714

Pantalla 6.118 Ejemplo problema1_cma_z. Curva de error de una señal senosoidal en un algoritmo cma.....715

Pantalla 6.119 Ejemplo problema1_dmi_z. Curva de error de una señal senosoidal en un algoritmo dmi 715

Pantalla 6.120 Ejemplo problema1_rls_z. Convergencia de los pesos estimados ...716

ABREVIATURAS

LMS	algoritmo adaptativo mínimo cuadrado
RLS	algoritmo adaptativo mínimo cuadrado recursivo
CMA	algoritmo adaptativo de modulo constante
DMI	algoritmo adaptativo de inversión de matriz directa
COTA	valor que se obtiene de la potencia de la señal, sirve para hallar El valor de mu, (constate de ajuste del algoritmo lms)
mu	constante de ajuste del algoritmo lms
PCM	Modulación de Pulso de Código
fir	respuesta de pulso finita
IIfir	respuesta de pulso infinita
Potencia	calculo de la potencia del sistema
h	Valores iniciales
entrada	Señal de entrada
n	ruido del sistema
ruido	ruido adicional en las antenas
senal_referencia	señal de referencia del sistema
e	error evaluado por el programa
error	error ingresado por el usuario
P	Vector de valores de delta, matriz identidad en el algoritmo rls
delta	constante del algoritmo rls y cma

lamda	constante del algoritmo rls y cma
beamforming	zona de formación de un haz
AM	Amplitud Modulación
PCM	Modulación de Pulso de Código
RF	Radio Frecuencia
scrambling	señal revuelta o código revuelto
broadcast	todos los usuarios de un grupo, zona de difusión
W	Valores de los Pesos
.y(n)	valor a la salida del filtro
N	número de elementos de los lazos, entrada, pesos, cálculo de la curva de error

RESUMEN

En el presente documento haremos una prueba de algoritmos adaptativos ya existentes utilizando el programa matlab.

Los algoritmos adaptativos pueden ser clasificados en categorías, basándose en distintas técnicas. Las técnicas basadas en información se clasifican en algoritmos basados en la señal de referencia y en algoritmos adaptativos ciegos.

Nuestro proyecto de graduación analiza los algoritmos adaptativos pero sobre todo los basados en técnicas de información como son los de señal de referencia y los algoritmos adaptativos ciegos. Dentro de los algoritmos adaptativos también hay dos grandes grupos, los que son llamados algoritmos adaptativos especiales y los algoritmos adaptativos generales. Se los considera de uso especial porque permiten obtener una convergencia más rápida del array de antenas.

Mientras que los algoritmos generales tienen una convergencia más lenta que los especiales.

Los algoritmos basados en la señal de referencia se basan en la minimización del error mínimo cuadrado entre la señal recibida y la señal de referencia. Por lo tanto, se requiere que la señal de referencia esté disponible. La señal de referencia tiene una alta correlación con la señal deseada, por ejemplo, el algoritmo SMI (Matriz de inversión simple) , LMS (Least Mean square) y RLS (Recursive least square). La señal de referencia no es la verdadera señal deseada, de hecho es una señal que la representa de cerca o tiene una alta correlación con esta. La señal de referencia, requerida por el algoritmo, es generado de varias maneras.

Los algoritmos adaptativos ciegos no requieren señal de referencia, generan por si mismos la señal de referencia requerida desde las señales decepcionadas, para obtener la señal deseada. Por ejemplo, algoritmos CMA (modulo algorítmico constante), ciloestabilidad y decisión directa.

Los tipos de algoritmos adaptativos que analizamos en las simulaciones son: Algoritmo Mínimo cuadrado (LMS), Algoritmo

Mínimo Cuadrado Recursivo (RLS), Algoritmo de modulo constante (CMA) y el algoritmo de Inversión de Matriz Directa (DMI).

En este documento se determina la capacidad de red, la cobertura y el desempeño de las antenas inteligentes utilizando algoritmos adaptativos en implementaciones en matlab, en la actualidad ante el gran auge de las telecomunicaciones y el crecimiento continuo del número de usuarios de sistemas de comunicaciones móviles y la implementación de nuevas plataformas servicios móviles (3G) y el gran aumento de números de celulares, se ha presentado el problema de cómo aumentar la convergencia de los arreglos de antenas, y tener simulaciones adecuadas y próximas a la realidad que permitan simular correctamente la realidad del array de antenas. Esto se logra utilizando algoritmos adaptativos, los cuales permitirán calcular más rápidamente la convergencia de los pesos de las antenas, y una herramienta para hacer estas simulaciones es por medio del programa matlab.

INTRODUCCIÓN

En esta sección describiremos brevemente la función que tiene cada uno de los elementos que conforman el proyecto. Expondremos además las bases sobre las cuales se justifica el proyecto y el aporte que este brinda al campo educativo para incentivar a los jóvenes a crear, investigar y desarrollar los campos de el estudio de antenas inteligentes, el uso de algoritmos adaptativos y el desarrollo de herramientas computacionales en matlab..

Este proyecto en su forma global involucra una variedad de tecnologías, para lo cual primero expondremos el material teórico de que son antenas inteligentes y que son algoritmos adaptativos. Luego realizamos implementaciones en matlab de algunos algoritmos adaptativos, estas implementaciones realizadas en matlab nos permitirán cumplir nuestros objetivos generales y específicos.

Para la resolución de las implementaciones del presente proyecto se utilizo como base las aplicaciones lms y rls de la central matlab, junto con los valores recomendados por esta para antenas. También se utilizo información de antenas adaptativas para comunicaciones de Magdalena Salarzar Palma catedrática de la universidad Carlos III de España, se considero interesante agregar a este trabajo técnicas de optimización de señal del laboratorio de comunicaciones de la universidad de Alcala por Maria Pilar Jarabo Amores, en el cual ella utiliza una técnica para determinar la potencia de las señales utilizadas, esta potencia junto con el orden de la señal lo utiliza para determinar un valor que denomina cota,

con este valor de cota la simulación en matlab encuentra automáticamente el valor de la constante de ajuste mu que utiliza el algoritmo lms (este método solo sirve para el algoritmo lms), el valor de mu es ingresado normalmente a mano en el programa lms (asi está en la central matlab), por lo que nos pareció muy útil colocar este método para encontrar el valor de mu en nuestra tesis para que el programa lo encuentre automáticamente, el laboratorio de señales de la universidad de alcala también presentan un método para ingresar señales senosoidales en algoritmos lms, al encontrar este método mi compañera de tesis y yo decidimos utilizarlo y aplicarlo además en los algoritmos rls, cma y dmi, principalmente ver su comportamiento en los algoritmos cma y dmi, ya que no hay muchas implementaciones disponibles en cma y dmi. Esto lo hacemos con el objetivo de implementar una misma señal con condiciones parecidas de entrada en los algoritmos lms, rls, cma y dmi (dentro de lo posible) ya que algunos algoritmos tienen ciertas características propias (algunos algoritmos funcionan mejor entre 400 y 600 valores de muestra, y otros entre 40 y 200), aun así tienen muchas características comunes como son los valores iniciales de entrada, la señal de referencia, la señal de entrada, los valores de ruido, todo esto nos permitirá comparar los resultados obtenidos y sacar conclusiones. Cuando revisábamos los foros de telecomunicaciones en internet vimos que normalmente se coloca la señal de ruido a la entrada o a la salida del algoritmo utilizado, decidimos hacer que en nuestras implementaciones se pudiera escoger si se ingresaba el ruido a la entrada del sistema o a la salida del filtro. En una página francesa alguien comento que kalman y wiener pronosticaban que podía cambiarse la señal de referencia con otra señal muy parecida a la original, al leer esto decidimos agregar un cambio de señal en una de nuestras

aplicaciones que permitiera cambiar la señal de referencia por otra señal, y ver si se cumplía lo pronosticado por kalman y wiener, (ellos también pronosticaron que se podía hacer pero que aumentaría el error del sistema).

DESCRIPCIÓN DEL PROYECTO

En la actualidad ante el gran auge de las telecomunicaciones y el crecimiento continuo del número de usuarios de sistemas de comunicaciones móviles y la implementación de nuevas plataformas servicios móviles (3G) y el gran aumento de números de celulares, se ha presentado el problema de cómo aumentar la convergencia de los arreglos de antenas, y tener simulaciones adecuadas y próximas a la realidad que permitan simular correctamente la realidad del array de antenas. Esto se logra utilizando algoritmos adaptativos, los cuales permitirán calcular más rápidamente la convergencia de los pesos de las antenas, y una herramienta para hacer estas simulaciones es por medio del programa matlab, con lo cual nos podemos hacer algunas preguntas .¿Cuál es la capacidad de red, cobertura y desempeño de las antenas inteligentes utilizando algoritmos adaptativos con simulaciones en matlab.?. Que tanto de diferencia de convergencia hay entre un método y otro.

Actualmente existen implementaciones en matlab de algoritmos LMS y RLS, pero no hay disponibles implementaciones en CMA y en DMI en la central matlab, sabiendo esto nos podemos plantear algunas preguntas como son: ¿Se podría implementar algoritmos adaptativos CMA y DMI en matlab?, ¿Se podría aplicar los mismos valores de entrada en estos cuatro algoritmos adaptativos y

comparar los resultados obtenidos?. ¿ se podría comparar la velocidad de convergencia, la potencia del sistema entre 3 y 10 pesos de antenas (numero de antenas)?. ¿Cuál sería el comportamiento de la velocidad de convergencia del algoritmo si variamos el numero de antenas (valores de entrada), ¿Variando el numero de antenas en estos algoritmos adaptativos se podría ver el comportamiento de las curvas en algunos tipos de gráficos como son: curvas de error, curvas de pesos estimados, curva de la señal a la salida del filtro vs la señal de referencia del sistema?, conociendo que hay implementaciones en matlab con señales gaussianas, y viendo que no hay implementaciones con señales senosoidales ¿se podría hacer implementaciones con señales senosoidales en estos algoritmos y comparar sus resultados en los diferentes algoritmos ?. En los algoritmos adaptativos LMS y RLS que existen se basan principalmente en el numero de antenas, en el numero de muestras de la señal, en los valores iniciales, en un vector pivote, en algunos de ellos en valores de paso y en constantes de ajuste como son u y mu en el LMS y Landa y P en el RLS, ¿Qué complicaciones y diferencias podrían ocurrir al implementar los algoritmos CMA y DMI?. Normalmente algunos programadores ingresan valores de ruido a la entrada del sistema (para simular señales reales) y otros lo colocan a la salida del filtro en los algoritmos adaptativos, ¿Podríamos comparar los resultados obtenidos al ingresar los ruidos a la entrada y a la salida del filtro?, Wiener y Kalman dedujeron que si se cambia la señal de referencia con una señal parecida a la original también se puede obtener la convergencia de las señales, ¿ qué tanto aumenta el error en la convergencia al cambiar la señal de referencia por otra parecida a la original?

En telecomunicaciones y en telefonía son utilizados los algoritmos adaptativos, sería útil saber su comportamiento tanto para señales gaussianas como senosoidales, y comparar los resultados obtenidos por diferentes métodos adaptativos, como son el LMS (algoritmo adaptativo mínimo cuadrado), el RLS (algoritmo adaptativo minimo cuadrado recursivo), el CMA (algoritmo de modulo constante) y el DMI (algoritmo de matrix inversa), ver su comportamiento, velocidad de convergencia y comparar los resultados que se obtienen con los mismos datos de entrada.

El costo de instalar un sistema de antenas es muy caro, por lo que sería conveniente desarrollar simulaciones e implementaciones de antenas inteligentes que representen lo mejor posible la realidad de un conjunto de antenas antes de instalarlas para poder ahorrar costos, y ante el problema de encontrar simulaciones adecuadas que permitan determinar el comportamiento real de las antenas se ha planteado utilizar el programa matlab para implementar algunos algoritmos adaptativos.

El desarrollo de implementaciones de estos algoritmos adaptativos podría ser útil para mostrárselos a los estudiantes en alguna clase del laboratorio de telecomunicaciones, y que vean cómo se comportan las señales senosoidales con algunos valores de pesos (los pesos son los valores de entrada de las antenas).

En el presente proyecto escogimos 4 algoritmos adaptativos para nuestras simulaciones, en primer lugar el algoritmo adaptativo LMS (algoritmo mínimo cuadrado), el cual es muy popular y que tienen la particularidad de ser del tipo lineal, escogimos en segundo lugar el algoritmo RLS (algoritmo mínimo cuadrado recursivo) el cual es el desarrollo lógico del anterior. Escogimos otros dos algoritmos que no son tan conocidos, pero que presentan características muy importantes estos son el algoritmo CMA (algoritmo adaptativo de modulo constante) y el algoritmo adaptativo de inversión de matriz directa DMI. Le dedicamos un capitulo a cada uno de estos algoritmos y exponemos una descripción del ejemplo a correr en el programa, características, tablas de datos, corridas de varios ejemplo y los resultados del mismo. Este mismo procedimiento lo utilizamos para los cuatro algoritmos adaptativos. Al final hacemos una exposición de todo lo que nos pareció importante al hacer los algoritmos, que dificultades se presentaron al implementarlos y como los resolvimos, luego al compararlos hicimos observaciones, sacamos conclusiones e hicimos recomendaciones al respecto.

OBJETIVOS

Objetivo General

El objetivo es implementar en matlab distintos tipos de algoritmos adaptativos basados en la señal de referencia y algoritmos adaptativos ciegos, en los cuales se hará un análisis y se determinara la capacidad de

red, la cobertura y el desempeño de las antenas inteligentes utilizando algoritmos adaptativos en implementaciones en matlab.

Objetivos específicos.

- Representar el comportamiento de los arreglos de antenas utilizando algoritmos adaptativos en matlab.

- Representar en matlab la potencia de transmisión en su relación con la capacidad y cobertura utilizando algoritmos adaptativos.

- Representar en matlab la convergencia de los valores de entrada de un arreglo lineal de n antenas.

- Representar via Matlab el error cuadrático medio del sistema de antena lineal.

- Visualizar via Matlab el seguimiento de la señal deseada.

- Correlacionar el número de antenas con la capacidad, cobertura y desempeño utilizando algoritmos adaptativos.

- Correlacionar la sumatoria de antenas dispuestas en arreglos (matriz) con la cobertura, capacidad y desempeño utilizando algoritmos adaptativos.

ORGANIZACIÓN DEL PROYECTO

El presente proyecto tiene el siguiente desarrollo:

En el capitulo 1 hablaremos sobre las antenas inteligentes, generalidades, definición, de los tipos de antenas inteligentes. De haz conmutado, de haz de seguimiento y de haz adaptativo. Y los elementos de cada uno de ellos.

En el capitulo 2 hablaremos sobre los algoritmos adaptativos, algoritmos adaptativos basados en la señal de referencia, algoritmos adaptativos ciegos, conformador de haces adaptativos.

En el capitulo 3 hablaremos sobre el algoritmo adaptativo LMS (algoritmo mínimo cuadrado), mostraremos el programa en matlab del algoritmo LMS, haremos una descripción del ejemplo a correr en el programa tanto para señales gaussianas como senosoidales, corridas del programa en diferentes ejemplos y los resultados obtenidos de la corrida. Los gráficos obtenidos en las implementaciones en matlab se colocaran en los anexos del presente proyecto. Se realizo 3 implementaciones o programas con el algoritmo lms, llamados problema1_lms, problema2_lms y problema3_lms.

Los programas problema1_lms y problema3_lms trabajan con señales senosoidales. El programa problema2_lms trabaja con señales gaussianas.

El Programa o implementación problema3_lms permite cambiar la señal de referencia.

En el capítulo 4 hablaremos sobre el algoritmo adaptativo RLS (algoritmo minimo cuadrado recursivo), mostraremos el programa en matlab del algoritmo RLS, haremos una descripción del ejemplo a correr en el programa tanto para señales gaussianas como senosoidales, corridas del programa en diferentes ejemplos y los resultados obtenidos de la corrida. Los gráficos obtenidos en las implementaciones en matlab se colocaran en los anexos del presente proyecto.

Se realizo 3 implementaciones o programas con el algoritmo rls, llamados problema1_rls, problema2_rls y problema3_rls.

Los programas problema1_rls y problema3_rls trabajan con señales senosoidales. El programa problema2_rls trabaja con señales gaussianas.

El Programa o implementación problema3_rls permite cambiar la señal de referencia.

En el capítulo 5 hablaremos sobre el algoritmo adaptativo CMA (algoritmo de modulo constante), mostraremos el programa en matlab del algoritmo CMA, haremos una descripción del ejemplo a correr en el programa tanto para señales gaussianas como senosoidales, corridas del programa en diferentes ejemplos y los resultados obtenidos de la corrida. Los gráficos obtenidos en

las implementaciones en matlab se colocaran en los anexos del presente proyecto.

Se realizo 3 implementaciones o programas con el algoritmo CMA, llamados problema1_cma, problema2_cma y problema3_cma.

Los programas problema1_cma y problema3_cma trabajan con señales senosoidales. El programa problema2_cma trabaja con señales gaussianas. El algoritmo cma con señales gaussianas tiende a no converger.

El Programa o implementación problema3_cma permite cambiar la señal de referencia.

En el capítulo 6 hablaremos sobre el algoritmo adaptativo DMI (algoritmo de Inversión de Matriz Directa), mostraremos el programa en matlab del algoritmo DMI, haremos una descripción del ejemplo a correr en el programa tanto para señales gaussianas como senosoidales, corridas del programa en diferentes ejemplos y los resultados obtenidos de la corrida. Los gráficos obtenidos en las implementaciones en matlab se colocaran en los anexos del presente proyecto.

Se realizo 3 implementaciones o programas con el algoritmo DMI, llamados problema1_dmi, problema2_dmi y problema3_dmi.

Los programas problema1_dmi y problema3_dmi trabajan con señales senosoidales. El programa problema2_dmi trabaja con señales gaussianas.

El Programa o implementación problema3_dmi permite cambiar la señal de referencia.

Se hará un análisis de los resultados obtenidos con los esperados. Por último las conclusiones, recomendaciones y futuros trabajos.

CAPÍTULO I

ANTENAS INTELIGENTES

A diferencia de otros sistemas como el GSM, los nuevos sistemas de comunicaciones móviles de Tercera Generación (3G) ofrecen una gran variedad de servicios, desde los tradicionales servicios de voz hasta servicios multimedia de alta binaria. En estos escenarios multiservicio, la demanda de tráfico irá creciendo progresivamente a lo largo de los próximos años, lo que obligará a los operadores a optimizar sus procesos de planificación de red y a introducir nuevas tecnologías que mejoren la eficiencia espectral de los despliegues celulares.

Las antenas inteligentes constituyen una tecnología novedosa que sin duda puede contribuir significativamente a mejorar la capacidad de las redes 3G. A diferencia de las antenas sectoriales desplegadas en los emplazamientos de estaciones base, los esquemas de antenas inteligentes basados en conformación actúan como un filtros espaciales que pueden separar señales operando en el mismo canal de tiempo, frecuencia o código siempre que los usuarios móviles están separados angularmente. Los beneficios que pueden obtenerse con el desligue de antenas inteligentes son:

Mayor radio de cobertura, mejora de la capacidad, reducción en la potencia transmitida, control de potencia más eficiente y la posibilidad de ofrecer nuevos servicios de valor añadido basados en localización.

En el caso particular de sistemas basados en CDMA, todos los usuarios transmiten simultáneamente ocupando toda la banda de frecuencias disponible, de manera que la interferencia cocanal experimentada por los usuarios crece a medida que lo hace el número de usuarios. Gracias a la capacidad de cancelación de interferencias, las antenas inteligentes son especialmente adecuadas para este tipo de sistemas limitados por interferencia, como el caso de WCDMA (Wideband CDMA), el modo FDD de UMTS.

Sin embargo, y a pesar de las ventajas que aportaría una antena inteligente, el despliegue de estos sistemas en redes de comunicaciones móviles no es una realidad y se encuentran muy pocos desarrollos prácticos. En las últimas décadas, la antena inteligente se ha estudiado desde el punto de vista del algoritmo de conformación, bajo una perspectiva de procesado de señal, sin considerar el impacto que la antena produce sobre el sistema. La antena inteligente no sólo tiene impacto en la estructura del receptor, sino que también tienen un efecto en aspectos de sistema, como la planificación y dimensionado, y la gestión de recursos radio (control de potencia, traspaso). Actualmente, no se dispone de

ninguna herramienta de planificación con antenas inteligentes y su impacto en el sistema no ha sido evaluado.

Esto es debido en parte a la dificultad que supone la introducción de antenas inteligentes adaptativas en un simulador de sistema. Por ello, el operador de la red móvil no puede estimar la mejora de prestaciones (capacidad, cobertura) que obtendría desplegando antenas inteligentes en sus estaciones base.

La principal dificultad para introducir un esquema de antena adaptativa en el simulador de red aparece ligada al hecho que el diagrama de radiación de la antena inteligente no se conoce a priori y además depende de un gran número de variables: algoritmo de conformación, distribución espacial de los usuarios y perfiles de servicios demandados. En otros trabajos de investigación, la introducción de antenas inteligentes en el nivel de sistema se hace en base a hipótesis que en la gran parte de los casos no responden fielmente a lo que sucede en la realidad. La mayor limitación que puede encontrarse estriba en el hecho de que no se considera la naturaleza dinámica del diagrama de radiación de la antena adaptativa, y éste se modela mediante por tramos rectos o con diagramas no realizables. En otros casos, se modelan escenarios de despliegue canónicos en los que el diagrama se supone conocido a priori sin considerar los parámetros de los que éste depende.

1.1 Generalidades sobre Antenas Inteligentes

Un arreglo de antenas puede estar constituido por elementos activos y pasivos.

Las antenas inteligentes trabajan con arreglos donde todos los elementos son activos.

Sin embargo al hablar de “inteligencia”, las antenas no son inteligentes sino lo que es inteligente es el sistema dado que puede interactuar con el medio.

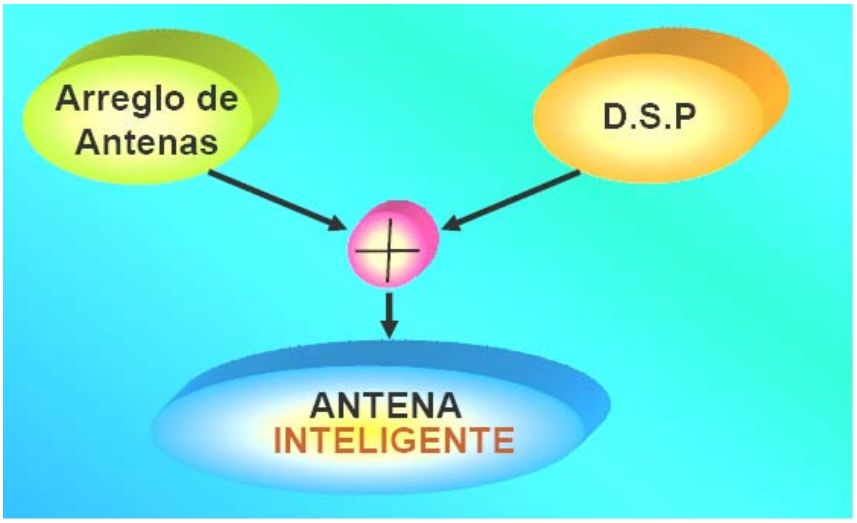


Figura 1.1 Antena Inteligente: D.S.P + Arreglo de antenas

Por tanto la inteligencia radica en la posibilidad de variar el patrón de radiación una vez que se ha implementado la antena.

Los sistemas de antenas inteligentes son arreglos de antenas que mediante el control de la fase y la amplitud de la excitación de cada uno de los elementos que conforma el arreglo permite variar la forma del patrón de radiación en tiempo real.

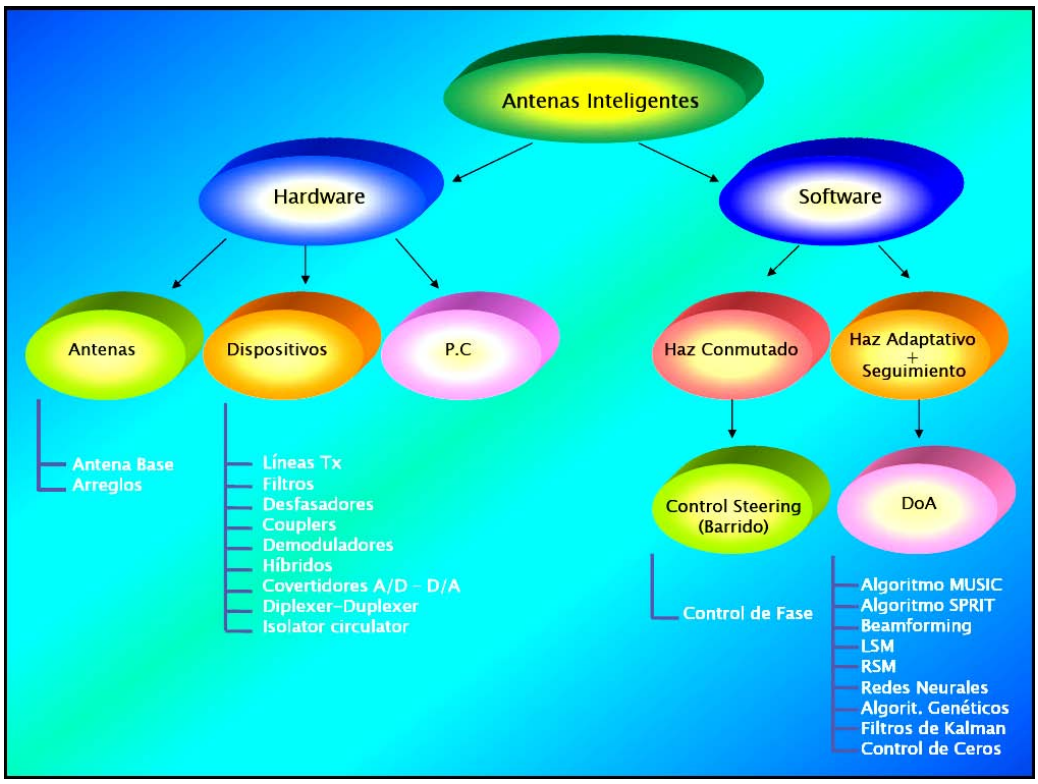


Figura 1.2 Conformación de las antenas inteligentes

El crecimiento continúo del número de usuarios de sistemas de comunicaciones móviles y la implementación de nuevas plataformas de servicios móviles han provocado la necesidad de aumentar sus capacidades al más alto nivel posible.

Los arreglos utilizan antenas múltiples, o elementos múltiples, para lograr un alto rendimiento que incluyen ganancias altas.

Los elementos deben ser examinados cuidadosamente para permitir seguir la dirección del Haz de radiación de los arreglos por encima de un rango angular amplio dentro de las frecuencias del ancho de banda del arreglo.

Si el espaciamiento entre elementos es demasiado grande, entonces los lóbulos radiados indeseables (la antena emite en otras direcciones que en la dirección del lóbulo deseado) se dirigirán hacia donde radie el arreglo.

Las interacciones electromagnéticas entre los elementos próximos del arreglo espaciados estrechamente varían el modelo de radiación de los elementos individuales y también afectan los modelos de radiación del arreglo.

El arreglo puede usar elementos omnidireccionales o elementos direccionales que se orienten radialmente hacia el exterior.

El arreglo consiste típicamente en 30 a 100 elementos uniformemente espaciados.

Además de combinar los elementos para formar un lóbulo de radiación simple, ellos pueden ser usados individualmente para proveer diversidad de ganancias en canales que presentan desvanecimiento debido a la interferencia por múltiple trayectoria.

Mediante sistemas de antenas inteligentes se consigue aumentar la capacidad de conexión a múltiples usuarios simultáneamente.

1.1.1 Definición de antenas inteligentes.

Una antena inteligente es la combinación de un arreglo de antenas (arrays) con una unidad de procesamiento digital de señales (DSP) que optimiza los diagramas de transmisión y recepción dinámicamente en respuesta a una señal de interés en el entorno.

La antena inteligente en vez de disponer de un diagrama de radiación fijo, es capaz de generar o seleccionar haces muy directivos enfocados hacia el usuario deseado, e incluso adaptarse a las condiciones radioeléctricas en cada momento.

Las antenas inteligentes hacen uso de:

• Arreglos de antenas

• Procesamiento de señales en radiofrecuencia

• Procesamiento de señales digitales con el propósito de mejorar la capacidad de los canales y la calidad de los servicios de comunicación, esencialmente de carácter móvil.

Ejemplos:

• Telefonía celular

• Redes inalámbricas

El caso de la telefonía celular

• Consiste en dividir un área geográfica en celdas o células hexagonales en

cuyo centro se encuentra una estación base (EB).

• La EB puede transmitir y recibir señales de RF y establecer un enlace con un usuario (móvil) presente dentro de su área de cobertura.

• Cada EB tiene la capacidad de dar servicio a cierta cantidad de usuarios simultáneamente, y de conectarlos a otros usuarios en la misma o en otras células, o al servicio público de telefonía conmutado.

En el diagrama de bloque de la figura 1.3 vemos un área multidisciplinar entre las cuales se pueden mencionar: Arreglos de antenas, antenas adaptivas, propagación, teoría de las comunicaciones, procesamiento digital de señales, entre otros.

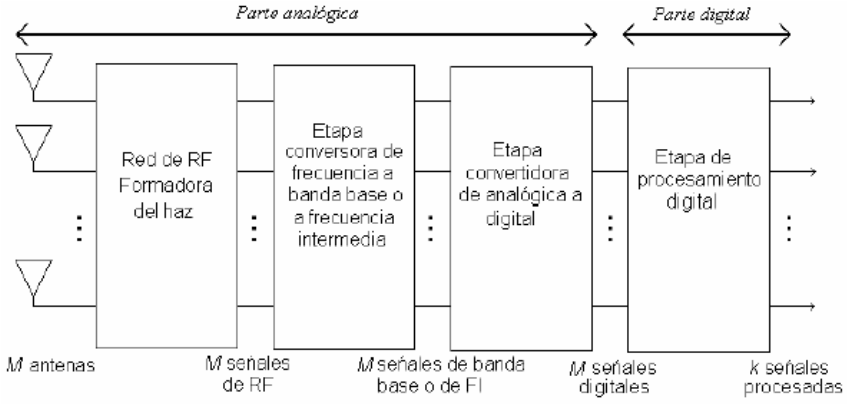


Figura 1.3 diagrama de bloque de las antenas inteligentes

Figura 1.4 Prototipos de antena inteligente de la firma Allgon

1.1.2 Ventajas Potenciales de los sistemas de antenas inteligentes:

Incremento de la zona de cobertura

Reducción de la potencia transmitida

Reducción de la propagación multitrayecto

Reducción del nivel de interferencia

Mejora de la seguridad

Introducción de nuevos servicios

Mayor complejidad de los transceptores

Mayor complejidad de los procedimientos de gestión

Cambios en los métodos de planificación

Incremento de la zona de cobertura.

Dado que la ganancia es mayor que en el caso de antenas omnidireccionales o sectorizadas, para igual potencia transmitida, la señal se podría recibir a una mayor distancia. Este hecho podría permitir reducir el número de estaciones base necesarias para cubrir una zona, siempre y cuando no sea tráfico el factor limitante.

Con un arreglo de antenas la ganancia es mayor que en el caso de una antena omnidireccional o sectorizada, así transmitiendo a una misma potencia, se pueda recibir la señal a una mayor distancia.

El testbed (prototipo de prueba), mostrado en la Figura 1.5, opera en la banda de los celulares, en la cual se obtuvo una ganancia entre 4.6 a 10.9 dB.

Figura 1.5 Estación base de array de antenas a) antena ensamblada, b) equipo de recepción

Reducción de la potencia transmitida

Con el aumento de la ganancia producto del arreglo de antenas, se incrementa la sensibilidad de la antena de recepción, por lo tanto se puede transmitir a una potencia más baja e incidir directamente en el consumo de baterías.

El sistema de antenas inteligentes puede radiar una potencia menor por lo cual se puede reducir las especificaciones de los amplificadores de potencia

del sistema asociadas al sistema de antenas, generando una reducción de costos en las etapas de amplificación

La mayor ganancia de la antena permitirá incrementar la sensibilidad de la estación base, por lo que los móviles podrán transmitir con menor potencia, ahorrando batería. De igual modo, gracias a la ganancia del array, es posible que la estación base transmitida igual potencia, pese a que cada elemento del Array este radiando una potencia muy inferior. Así, se relajaría las especificaciones sobre los amplificadores de potencia utilizados, que podrían resultar más baratos.

Reducción de la propagación multitrayecto

En el caso del enlace ascendente, la antena inteligente de la estación base podría discriminar las componentes multitrayecto de la señal recibida desde móvil, o incluso explotarlas (mediante receptores 2D-Rake, por ejemplo). Esto dependerá de la configuración de antena escogida.

Debido a la menor dispersión angular de la radiación desde el sistema de antenas inteligentes, se reducirán significativamente los trayectos múltiples de la información que llegaría al equipo móvil. Esto permite simplificar el sistema de ecualización del sistema móvil.

Dependiendo de la configuración del sistema de antenas inteligentes, se pueden tener dos situaciones:

- Captación de la Onda principal de la señal de interés, eliminando las señales multitrayecto propias y las señales interferentes de otros usuarios.

- Captación de la onda principal de la señal de interés aprovechando la captación de sus señales multitrayecto, para reforzar la señal principal, y eliminar las señales interferentes de otros usuarios.

Reducción del nivel de interferencia

La mejor selectividad espacial de la antena permitirá a la estación base discriminar las señales de usuarios interferentes a favor de la señal del móvil deseado (en el caso del enlace ascendente), y también reducir el nivel de potencia transmitida en las direcciones de esos otros usuarios (en el caso del enlace descendente). De cualquier modo, se conseguiría aumentar la relación C/I, lo cual tiene dos consecuencias fundamentales:

Una mejora en la C/I implica directamente una mejora en la tasa de error (BER), lo que hace que la calidad del servicio aumente.

La reducción de la C/I puede explotarse directamente (mediante técnicas de multiplicación espacial) o indirectamente (realizando un plan de frecuencias más ajustados, en el caso de GSM) para aumentar la capacidad del sistema.

La selectividad espacial que proporciona el sistema de antenas inteligentes, permite discernir las señales interferentes provenientes de otros usuarios con esto se logra hacer insensible a la antena receptora en esas direcciones y evitar que esas señales sean procesadas en el sistema de recepción.

También permite reducir la potencia de transmisión en la dirección de esos usuarios para evitarles interferencias.

La reducción del nivel de interferencia reduce la tasa de error (BER), lo que permite aumentar la calidad de la transmisión de la información.

Mejora de la seguridad

Gracias a que la transmisión entre la estación base y el Terminal móvil es direccional, no será posible que un equipo ajeno intercepte la comunicación, a menos que se situé en la misma dirección en la que apunta la antena.

Además sería posible una localización precisa de usuarios que estén haciendo uso fraudulento de los servicios que ofrece la red de comunicación inalámbrica.

Introducción de nuevos servicios

Puesto que la red podría tener acceso a información acerca de la posición de los móviles, es posible pensar en servicios tales como radio localización en llamadas de emergencia, tarifación geográfica, publicidad de servicios cercanos, información en lugares turísticos, gestión avanzada de flotas, etc.

No obstante, la implantación de estas antenas en la red móvil no está exenta re inconvenientes, como los que se detallan.

Esta afirmación ha de utilizarse con cuidado, ya que, aunque se introdujeran antenas inteligentes en el sistema, seguiría siendo imprescindible utilizar antenas tradicionales para permitir los canales de difusión (BCCH en GSM o BCH en UMTS), tal y como se apunta.

El tipo de técnica (algoritmo) que se emplee en la conformación del diagrama de radiación de la antena, se verá en el apartado dedicado a algoritmos de conformación de haz

Mayor complejidad de los transceptores

En comparación con los sistemas radiantes convencionales, los sistemas de antenas inteligentes son mucho más complejos y difíciles de diseñar. Sera necesaria una cadena de transmisión/recepción independiente para cada elemento del array, y todas ellas deberán estar balanceadas y calibradas en tiempo real. Además, es imprescindible el uso de potentes procesadores (DSPs, por ejemplo) para ejecutar los algoritmos de optimización, conformación de haz, detección del ángulo de llegada, etc. En definitiva, se llega a la conclusión de que no será posible diseñar independientemente el sistema radiante y la propia estación base.

Mayor complejidad de los procedimientos de gestión

El hecho de que exista un haz de radiación enfocado hacia cada usuario implica que las funciones de red deben revisarse, en particular, las que afectan a la gestión de recursos radio (RRC) y a la gestión de movilidad (MM). Por ejemplo, algunos procedimientos que pueden verse afectados son los de selección y reelección de celda, establecimiento de conexiones, handover, paging, etc.

Cambios en los métodos de planificación

La introducción de un sistema de antena inteligente implicara tener muy en cuenta sus características, a la hora de realizar la planificación de la red celular. En particular, habrá que contar con el aumento de alcance, la eliminación de fuentes de interferencia, el seguimiento angular de los usuarios, etc.

1.1.3 Problemas de las antenas inteligentes

Los problemas que se presenta en las antenas inteligentes son:

Modelado de canales de recepción

Determinación de la dirección arribo de las señales

Formado de haz en la recepción

Síntesis de patrones de radiación en la transmisión

Modelado de canales de recepción.

Un problema que se presenta en las antenas es el modelado de los canales de comunicación, incluyendo los efectos de propagación en diversas condiciones (exteriores, interiores, zonas densamente pobladas, y zonas rurales, etc.)

Determinación de la dirección arribo de las señales.

Otro problema muy común que se presenta en las antenas inteligentes es la determinación de la señal de arribo, en donde se trata de determinar las características de las señales incidentes, identificando las señales de interés y las señales interferentes. Un algoritmo utilizado para este propósito es el llamado MUSIC (Multiple Signal Clasification)

Formado de haz en la recepción.

La formación de los haz de recepción es un problema que se forma colocando lóbulos en las direcciones de las señales de interés y produciendo nulos en las direcciones donde inciden señales que no son de interés.

Síntesis de patrones de radiación en la transmisión.

Este tipo de problemas se presenta al producir patrones direccionales con lóbulos laterales suficientemente pequeños.

Existen varias formas de implementar un sistema radiante con estas características, como se detalla a continuación.

La implantación de antenas inteligentes en una red de comunicaciones móviles se limita, en principio, a las estaciones base (o Nodos B en UMTS), debido a que necesariamente se deben emplear sistemas radiantes de mayor tamaño (arrays de varios elementos) . Esto tiene la ventaja añadida de que pueden introducirse las antenas inteligentes de forma transparente para los usuarios, que no tendrán que cambiar de Terminal para beneficiarse de esta tecnología. No obstante, algunos autores si han contemplado la posibilidad de incorporar antenas inteligentes a los terminales móviles.

La característica de las antenas inteligentes de tener unos haces de radiación con una mayor directividad (es decir, mayor ganancia y mayor selectividad angular), hace pensar en las siguientes ventajas potenciales de estos sistemas

1.1.4 Funcionamiento de las Antenas Inteligentes.

El principio básico de funcionamiento de las antenas inteligentes es que cada antena recibe una señal separada y definida. Dependiendo como está configurado el sistema inalámbrico, el receptor puede usar una señal para mejorar la calidad de otra señal, o podría combinar los datos de señales múltiples para ampliar el ancho de banda.

La señal que reciben las antenas es una señal de radiofrecuencia (RF) sin procesar. Esta RF se encamina inicialmente a circuitos que la manejan como una señal analógica, tal como un radio. Algunos dispositivos con antenas inteligentes aplican sus conceptos inteligentes en esta etapa analógica. Después del procesamiento inicial, la RF se convierte en una señal digital que luego se envía al dispositivo host como una cadena de datos. La mayoría de los dispositivos que usan las antenas inteligentes aplican sus conceptos inteligentes en esta en conjunto con circuitos digitales.

En cuanto a los cambios debido a la introducción de las antenas inteligentes en un sistema de telefonía móvil, por ejemplo, se supondrá una reducción en el número de estaciones base necesarias para dar cobertura a una zona de servicio y un aumento en el número de usuarios que puede atenderse. Por otro lado, es necesario definir y cuantificar un conjunto de parámetros que

caractericen las prestaciones de la antena inteligente en diferentes entornos, para poder seleccionar el esquema de conformación más apropiado en cada caso.

Las Smart Antenas, como suelen ser llamadas también, se diferencian de las antenas convencionales ya que pueden trabajar de dos modos distintos los que se describen brevemente a continuación:

• Modo omnidireccional. La antena en este modo funciona exactamente igual que las antenas convencionales es decir, emite señal con la misma intensidad hacia todas direcciones.

• Modo direccional. En este modo, la antena emite señal en una sola dirección y con un cierto ángulo de apertura. La consecuencia de transmitir en este modo se traduce en un mayor alcance hacia la dirección donde emite la antena debido a que ésta concentra todo su espectro de potencia en un rango de cobertura mucho menor.

1.1.5 Tipos de Alcance.

Según en el modo en que trabaje la antena, su alcance será uno u otro. Si la antena trabaja en modo direccional su alcance será mucho mayor que si lo hace en modo omnidireccional ya que, en este caso, concentra toda su potencia en un rango menor. Definimos zona, como la región donde se encuentran todos los usuarios. Esta zona se divide en dos subzonas.

- Subzona Broadcast. Esta zona se corresponde con el rango de alcance de la antena en modo omnidireccional.

- Subzona Beamforming. Esta zona está dividida en n beams. Un beam se define como el rango de alcance de la antena en modo direccional para un cierto ángulo de apertura. Según el ángulo de apertura que se utilice habrá más o menos beams.

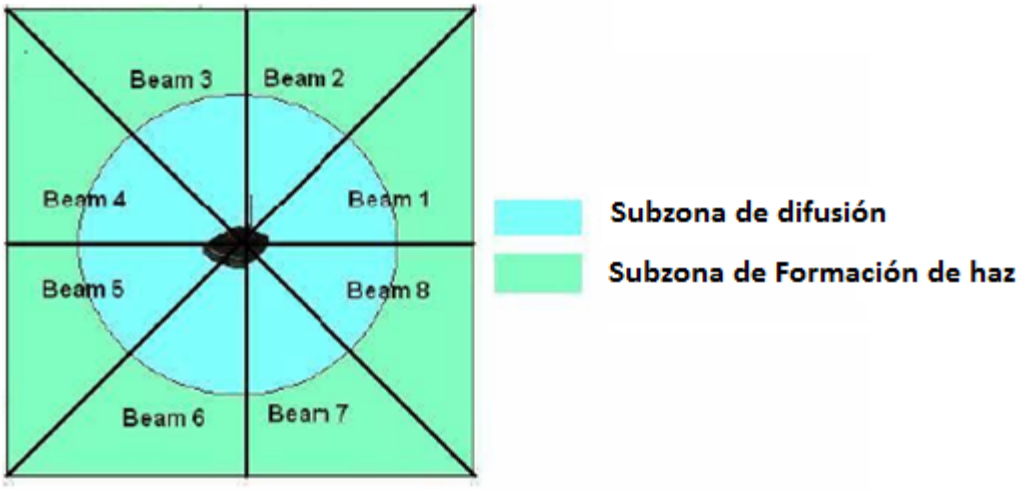


Figura 1.6 Subzonas de acuerdo a la configuración de Antenas.

Cabe comentar que pese a que en el dibujo la cobertura direccional (beams) es de forma triangular, en realidad la cobertura es un lóbulo redondeado donde existe una distancia máxima. (beamforming es la zona de formación de haz)

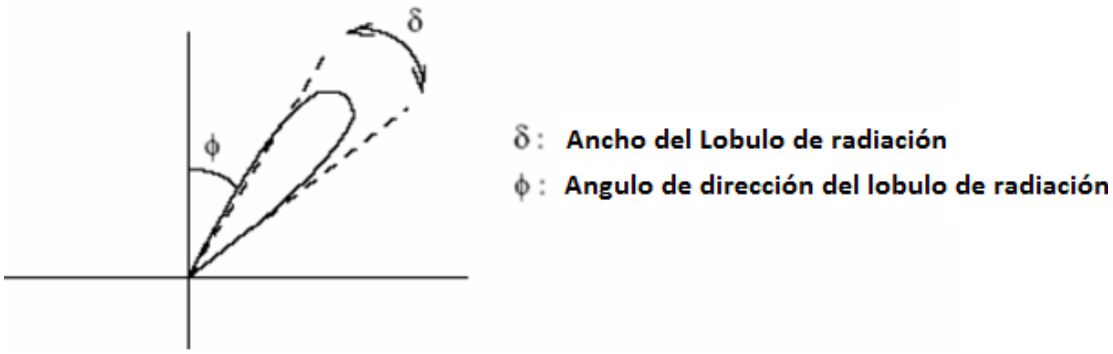


Figura 1.7 Lóbulo de radiación de una Antena Inteligente.

La característica principal de este tipo de antenas es que pueden orientar la señal que emiten hacia una cierta dirección donde se encuentra el usuario con el que se realiza la comunicación. Para esto, cada usuario debe tener, además de su propio identificador, una firma espacial que indique las coordenadas de la posición dentro de la zona. El AP utiliza la firma espacial de cada usuario para saber hacia dónde debe enfocar la antena en cada caso.

Obviamente, existe una fase previa a la transmisión de datos entre el AP y los usuarios. En esta fase, el AP debe descubrir cuáles son los usuarios que se encuentran dentro de la zona y obtener sus firmas espaciales.

Una vez el AP tiene conocimiento de la firma espacial de todos y cada uno de los usuarios se encuentra en condiciones de iniciar una transferencia de información.

Este proyecto se basa en el análisis de distintos métodos de descubrimiento de usuarios dentro de la fase previa descrita anteriormente.

A continuación se describirán brevemente cada uno de éstos tipo de antenas.

1.2 TIPOS DE ANTENAS INTELIGENTES

- De haz conmutado
- De haz de Seguimiento
- De haz Adaptativo

1.3 Sistema de haz conmutado (SWITCHED BEAM)

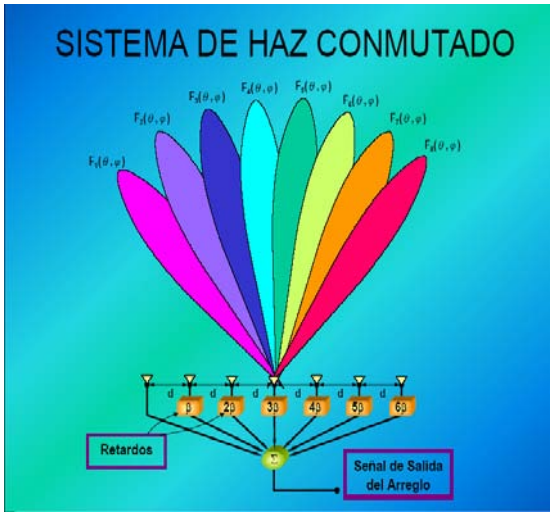


Fig. 1.8 Haz conmutado

En cada posición discreta del haz se activa el sistema de recepción para detectar la posible existencia de señales. En caso de recibir señal, el sistema guarda información correspondiente a la posición del haz (ángulo + identificación de usuario) y se establece la comunicación con el usuario en un intervalo de tiempo. Después de este intervalo se conmuta al siguiente haz para detectar la existencia de otros posibles usuarios hasta llegar al límite angular de la zona de cobertura. Este proceso se repite permanentemente en el tiempo.

El sistema de Haz conmutado es la técnica más simple entre las técnicas de antenas inteligentes. El sistema radiante genera varios haces fijos, cada uno de

ellos apuntando en una dirección distinta, de modo que entre todos se cubre toda la zona deseada (un sector o una celda). La inteligencia del sistema se encarga de seleccionar el haz que mejor servicio da a cada usuario en particular, en función de algún parámetro de control (mayor nivel de potencia recibida, mejor SNR y mejor C/I).

Esta técnica no garantiza que el móvil se encuentre en la dirección de máxima radiación del haz que le da servicio, ni que las señales interferentes se van notablemente reducidas (ya que siempre es posible que alguna entre por uno de los lóbulos secundarios). De hecho, sería posible recibir una señal interferente por un punto del diagrama de radiación con mayor ganancia que la señal deseada, empeorando apreciablemente las prestaciones del sistema.

Una versión más avanzada de esta técnica consistiría en seleccionar con un haz la señal deseada y con otras algunas de sus componentes multitrayecto, de forma que puedan procesarse todas como un receptor Rake.

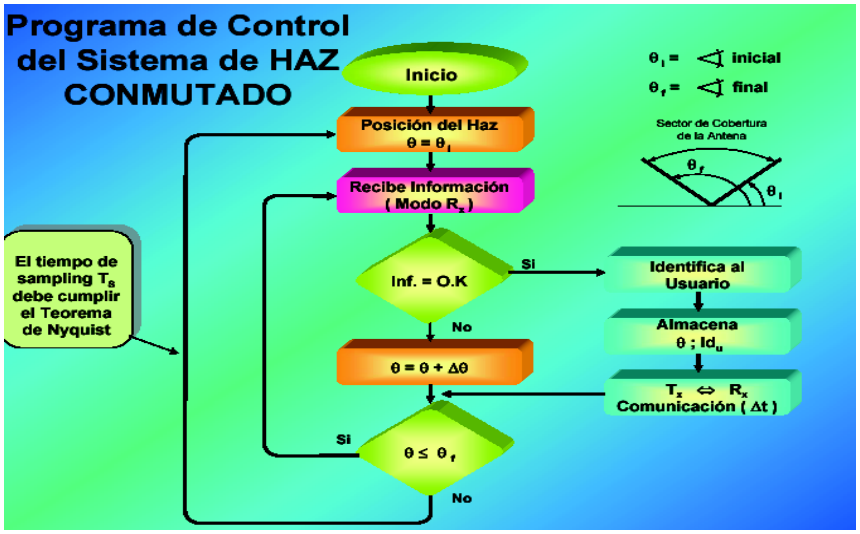


Fig 1.9 Programa que controla el sistema de haz conmutado

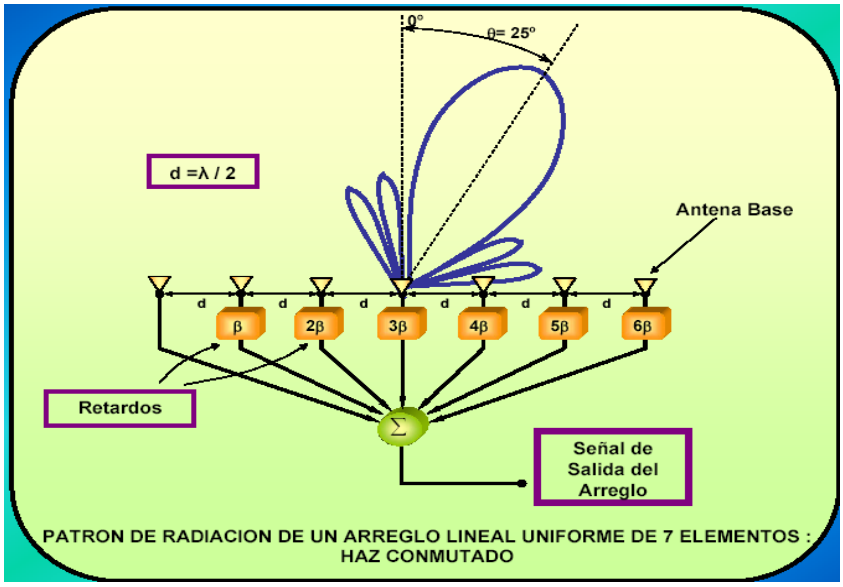


Fig 1.10 Patrón de radiación de un arreglo lineal uniforme de 7 elementos con un haz conmutado.

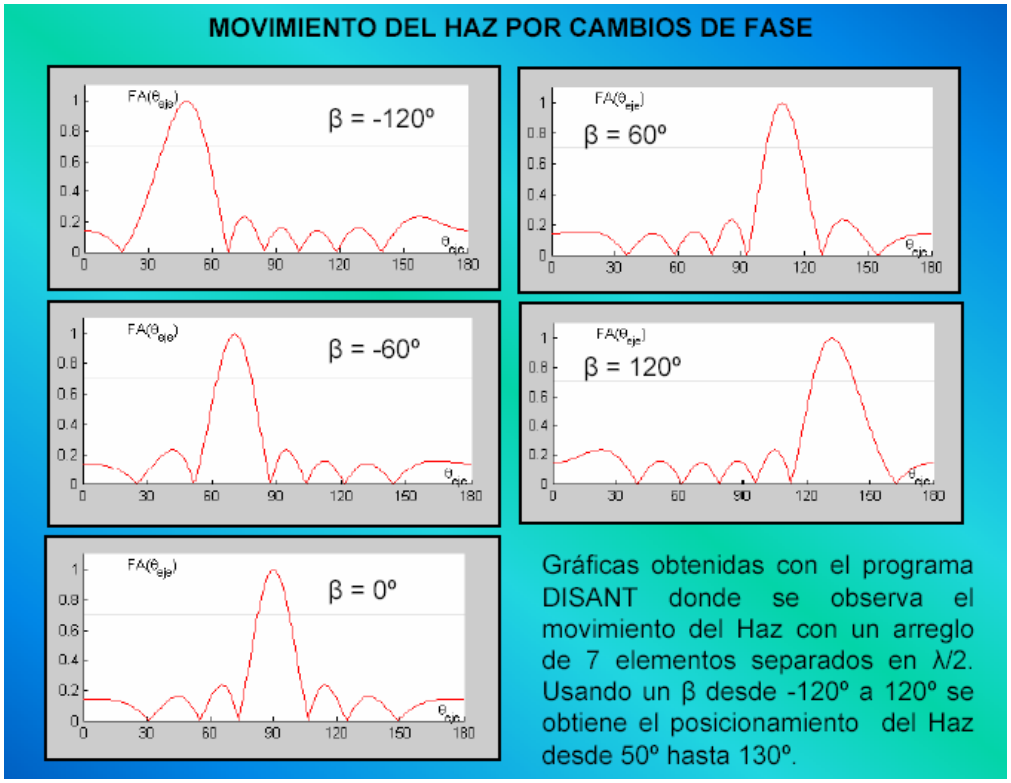


Fig 1.11 Movimiento del haz por cambios de fase

Elementos del sistema de haz conmutado

Elementos del sistema de haz Conmutado:

- Matriz de Butler
- Matriz de Blass
- Acopladores Direccionales
- Híbridos 3dB-90º
- Líneas de Transmisión
- Switches de Microondas (Diodos PIN)
- Sistema de Control de Fase

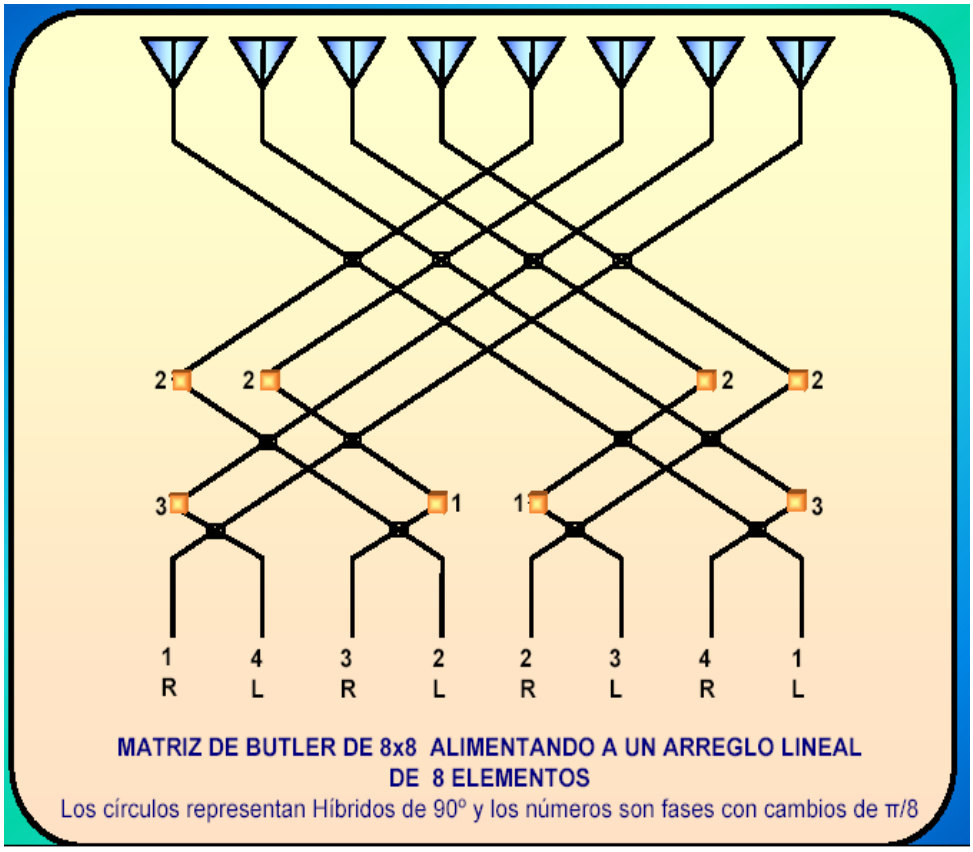


Fig 1.12 Matriz de Butler de 8x8 alimentando a un arreglo lineal de 8 elementos.

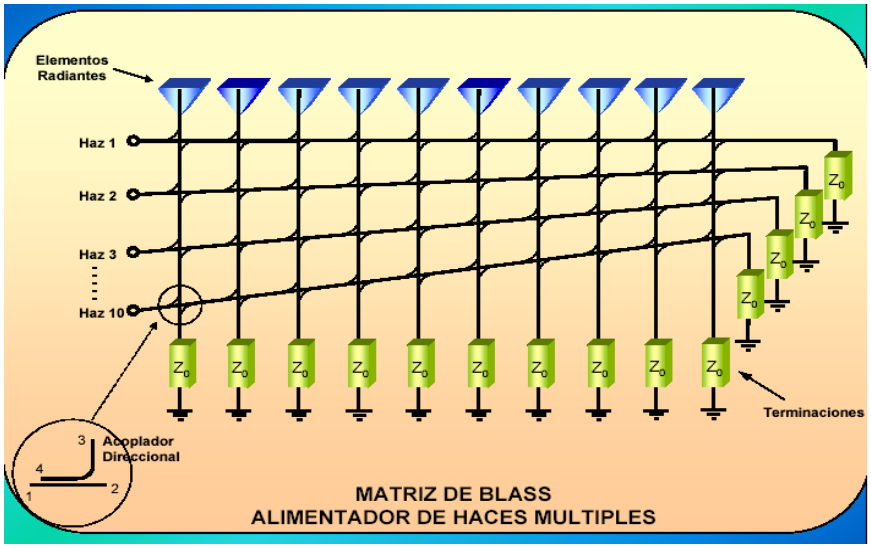


Fig 1.13 Matriz de Blass. Alimentador de haces múltiples.

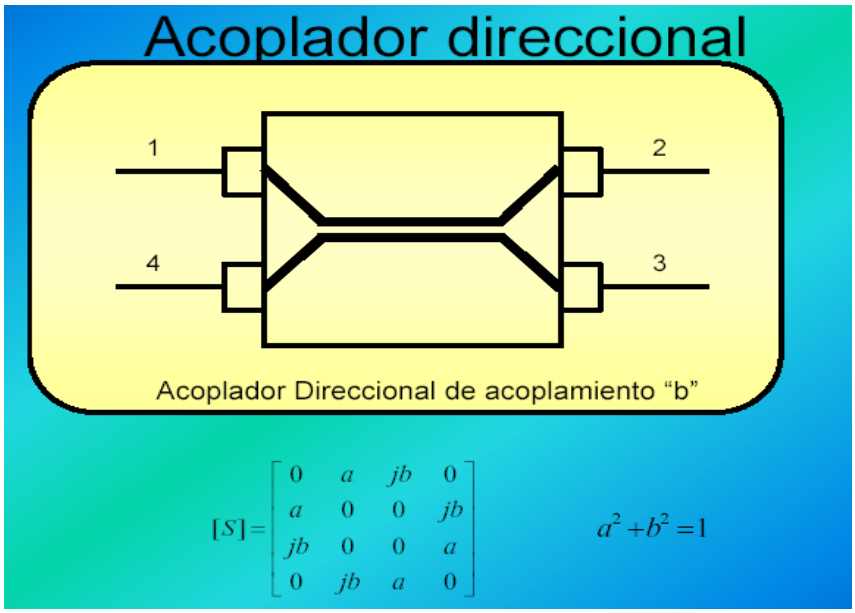


Fig 1.14 Acoplador direccional

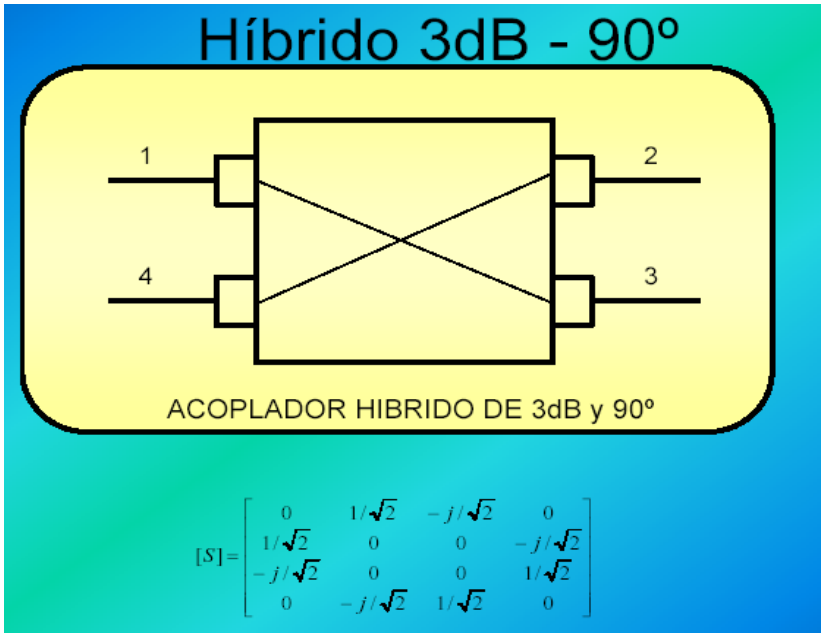


Fig 1.15 acoplador hibrido de 3 db y 90 grados

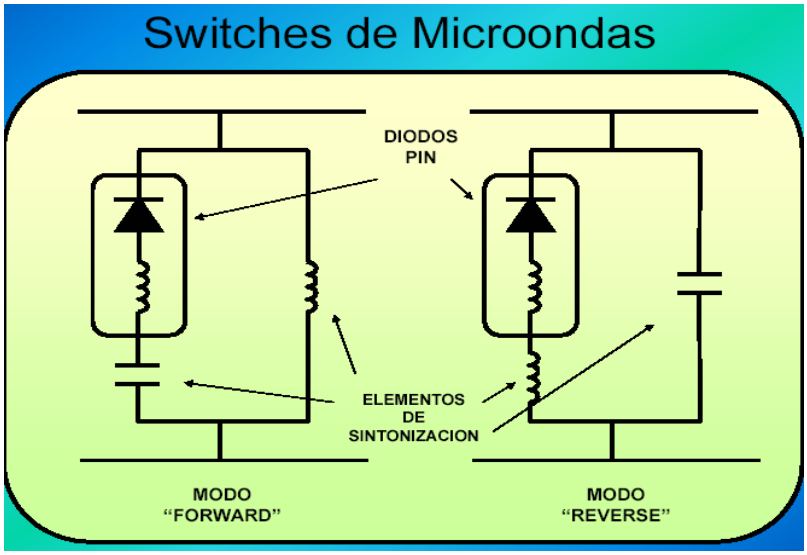


Fig 1.16 Switches de microondas.

1.4 Sistema Haz de seguimiento

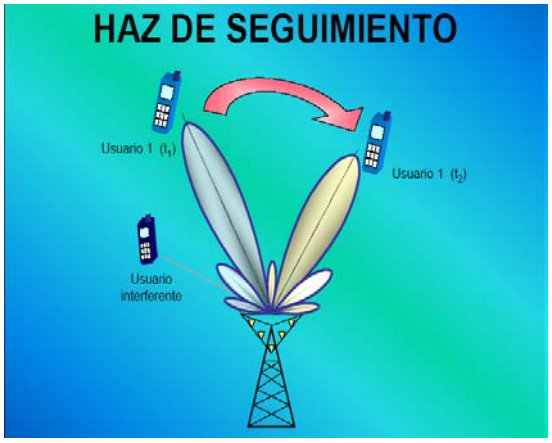


Fig 1.17 Sistema de haz de seguimiento

Este sistema es un poco más complejo que el anterior. Está conformado por un arreglo de antenas con una red de excitación que permite controlar electrónicamente las fases de las corrientes de excitación que llegan a los elementos del arreglo para modificar la dirección del haz convenientemente y establecer una comunicación con el usuario respectivo.

Requiere el uso de un array progresivo (paced array); es decir, un array en el que se pueden controlar electrónicamente las fases con las que se alimentan los distintos elementos, de modo que pueden modificarse a voluntad la dirección en la que apunta el lóbulo principal de la antena. A su vez, es necesario utilizar

algún algoritmo de detección de la dirección de llegada (DoA), de modo que pueda reorientarse dinámicamente el haz para apuntar al usuario deseado.

Con esta técnica si se puede garantizar que el usuario se encuentra iluminado en todo momento por el lóbulo principal y con máxima ganancia (dentro de las limitaciones de los algoritmos que se empleen). Sin embargo, tampoco puede evitarse que las interferencias entren por algún lóbulo secundario del diagrama de radiación.

Para aprovechar las señales multitrayecto sería necesario detectar y seguir con otros haces dichas componentes y luego procesarlas con un receptor Rake.

A diferencia del sistema de haz conmutado, el sistema haz de seguimiento ejecuta algoritmos DoA (Direction of Arrival) para identificar la dirección de arribo de las señales de los usuarios.

Otra diferencia es que los cambios de fase para en el sistema conmutado se realizan a ángulos fijos, es decir corresponden a ángulos prefijado en el sistema y en el sistema de Haz de seguimiento el posicionamiento del haz tiene mayor resolución angular.

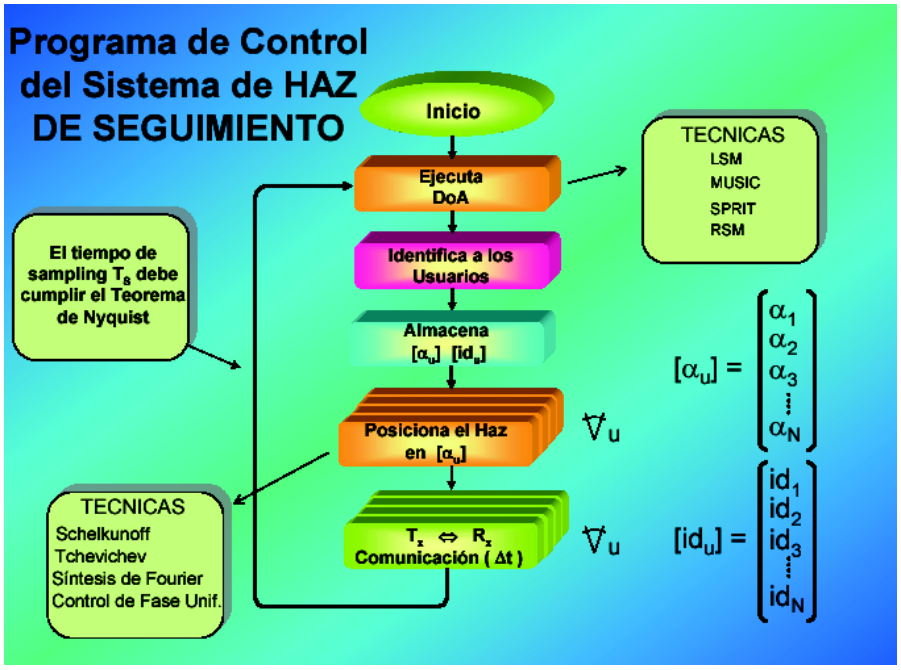
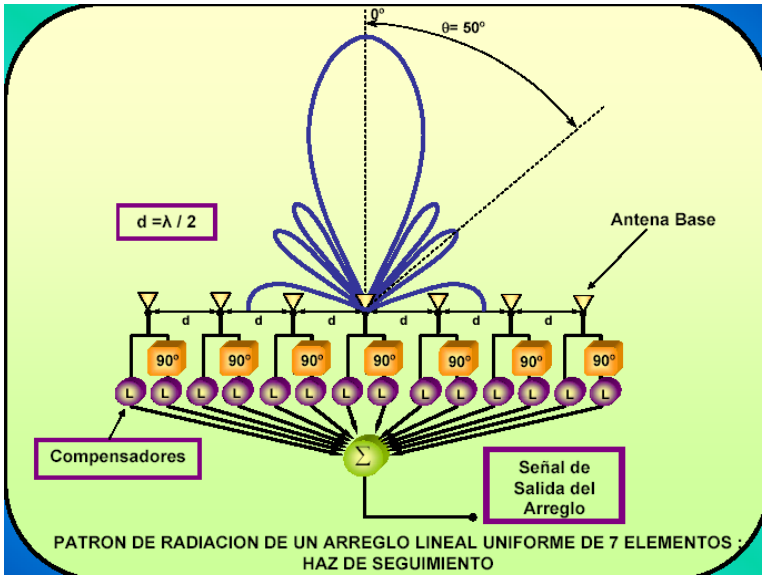


Fig 1.18 Programa de Control del sistema de haz de seguimiento



1.19 Patrón de radiación de un arreglo lineal uniforme de 7 elementos: haz de seguimiento

Elementos del sistema de haz de seguimiento

Los elementos del Sistema de Haz de Seguimiento:

- Desfasadores
- Compensadores
- Líneas de Transmisión

- Sistema de Detección de Dirección de Arribo (DoA)

- Sistema de Control de fase

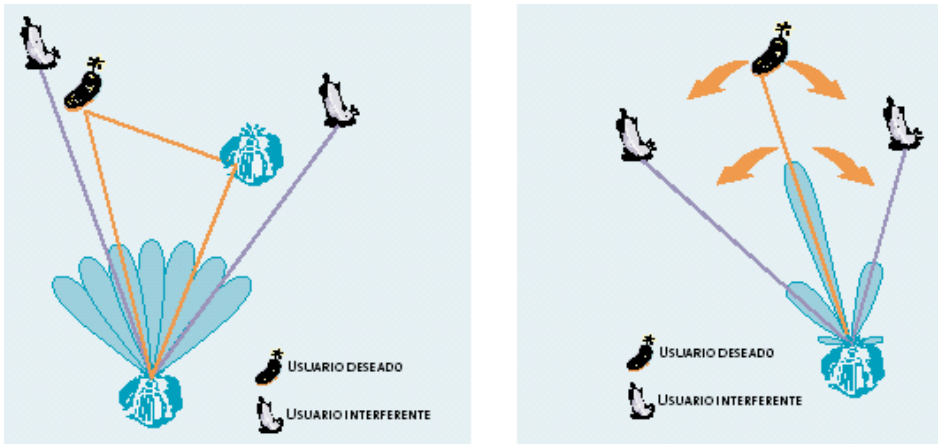


Fig 1.20 a) Antena de haz conmutado y b) Antena de haz de seguimiento

1.5 Haz Adaptativo

El Sistema de haz adaptativo constituye el máximo nivel de inteligencia que se le podría dar a un sistema de antenas. En este caso, en las antenas adaptativas la salida de cada elemento del array se pondera con un factor de peso cuyo valor se asigna dinámicamente, de modo que se conforma el diagrama de radiación para maximizar algún parámetro de la señal (por ejemplo, la SINR). De este modo, el diagrama sintetizado

habitualmente presentará un lóbulo principal en la dirección del usuario deseado, lóbulos secundarios en las direcciones de los componentes multitrayecto (si se quieren procesar con un receptor Rake) y mínimos (e incluso nulos) de radiación en las direcciones de las fuentes de interferencia.

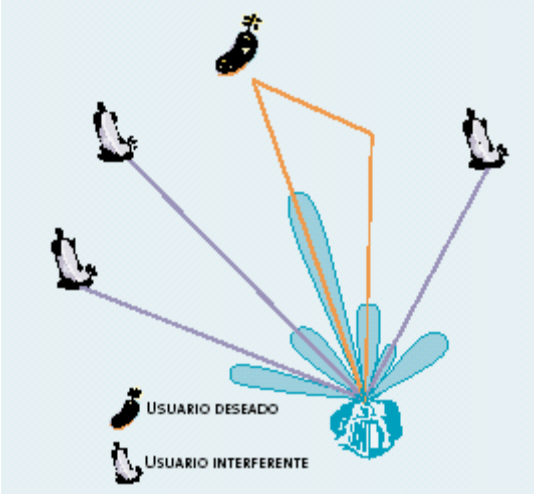


Fig 1.21 Antena de haz adaptativo

Fig 1.22 Haz adaptativo

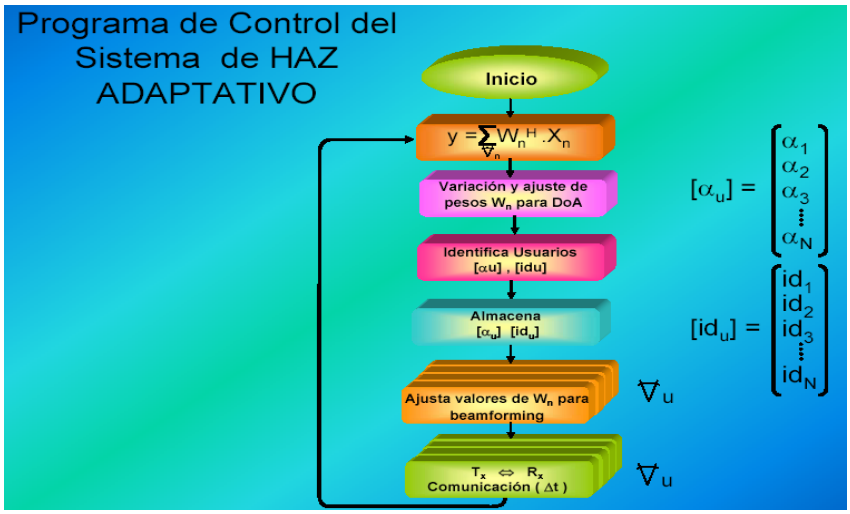


Fig 1.23 Programa de Control del sistema de haz adaptativo

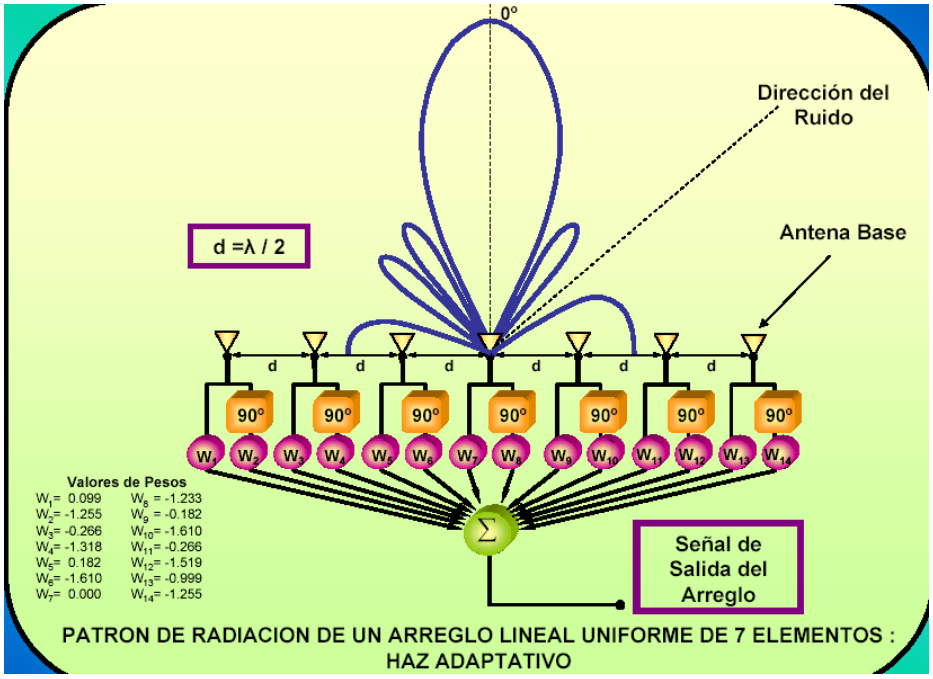


Fig 1.24 Patrón de radiación de un arreglo lineal uniforme de 7 elementos usando haz adaptativo.

Elementos de sistema de haz adaptativo

Los elementos del Sistema de Haz adaptativo:

- Mezcladores
- Oscilador local

- Filtros Pasa-Banda

- Down converter

- Convertidor Analógico/Digital

- Sistema de detección de dirección de arribo (DoA)

- Sistema de Conformación de Haz

Selectividad espacial en las antenas inteligentes

Como lo hemos mencionado, las características básicas que se busca en el diseño de un sistema de antena inteligente es la capacidad de seleccionar espacialmente a los distintos usuarios. Existen varias formas de implementar un sistema con esta capacidad, las cuales se describen a continuación por orden de complejidad.

No siempre será posible eliminar toda la interferencia, ya que el número de fuentes interferentes que se pueden suprimir está directamente relacionado con el número de elementos de la antena.

Esta técnica requiere el uso de complicados algoritmos, tanto para la detección de las señales deseada e interferente como para la optimización de los pesos que conforman el haz. Estos algoritmos suelen conllevar una gran carga computacional, mientras que deben procesarse en tiempo real, por lo que suponen una seria limitación. Por ser el caso más interesante, algunos de los algoritmos utilizados se comentan más adelante.

Una vez conocidos los tipos de antenas inteligentes, es necesario estudiar los modos de introducción de esta tecnología en una red de comunicaciones móviles.

Existen tres modos de aplicarla, en función del grado de aprovechamiento de la selectividad espacial que ofrece:

- Receptor de alta sensibilidad (HSR). Esta configuración consiste en utilizar antenas inteligentes solo en el enlace ascendente. De este modo, gracias a la mayor directividad de la antena, se consigue mejorar la sensibilidad global de la cadena de recepción de la estación base. Esto supone varias ventajas:

- En primer lugar, al mejorar la sensibilidad en el enlace ascendente, aumentara la extensión de la zona de cobertura. Esta mejora podría llegar a

ser tan grande como para que fuera el enlace descendente el más restrictivo a la hora de calcular la cobertura de una estación base.

- En segundo lugar, la mayor ganancia de la antena significa también que los móviles más cercanos podrían emitir con menor potencia manteniendo la calidad del enlace, con el consiguiente ahorro de baterías.

- Por último, se lograría una mejora de la relación C/I, lo que implicaría menores tasas de error y una mejor calidad. No podría emplearse la mejora en la C/I para incrementar la capacidad de un sistema CDMA, ya que dicha mejora solo está presente en el enlace de subida y no en el de bajada.

- Rechazo de interferencias por filtrado espacial (SFIR) En esta configuración se emplean antenas inteligentes tanto en el enlace ascendente como el descendente, con lo cual se consigue aprovechar la mejora por selectividad espacial en ambas direcciones.

En este caso, la mejora que se experimenta en la C/I, además de reducir la BER del sistema, puede utilizarse directamente para aumentar la capacidad de un sistema CDMA como es UMTS. Esto también podría lograrse indirectamente en GSM, si se hace un plan de frecuencias más ajustado:

al ser menor la distancia de reutilización, puede aumentar el número de portadoras por estación base.

- Acceso múltiple por división espacial (SDMA). Esta sería la configuración más compleja, pues consiste en aprovechar al máximo las propiedades de selectividad espacial de las antenas de ambos enlaces para ubicar. simultáneamente a varios usuarios en el mismo canal. Es decir, que podría haber varios usuarios utilizando al mismo tiempo la misma frecuencia y el mismo código de scrambling (o el mismo timeslot en GSM), estando discriminados únicamente por su posición angular respecto de la estación base.

En este caso, el aumento en la capacidad se produce de forma directa, debido a que se ha añadido una nueva dimensión para la gestión del espectro.

La introducción de SDMA supondría la necesidad de contar también con complicados sistemas de gestión de usuarios, de asignación de canales, etc.

La aplicación de SDMA a UMTS es bastante dudosa: al ser un sistema CDMA existen gran cantidad de usuarios compartiendo simultáneamente la misma frecuencia y que se distinguen solo por su código de scrambling

(código revuelto). Por tanto, sería muy complejo implementar un sistema capaz de diferenciar a cada usuario por su situación espacial, además de poco necesario, ya que los códigos scrambling (códigos revueltos) producen una separación suficiente y existen códigos de sobra para todos los usuarios.

CAPÍTULO 2

ALGORITMOS ADAPTATIVOS

Las antenas inteligentes constituyen una de las más prometedoras tecnologías que permiten alta capacidad en sistemas inalámbricos, reduciendo efectivamente la multitrayectoria y la interferencia cocanal. Esto se logra focalizando la radiación sólo en la dirección deseada y ajustándose a los cambios de tráfico en el medio.

Las antenas inteligentes emplean un conjunto de elementos radiadores organizados en forma de arreglos, las señales desde estos elementos son combinadas para formar un patrón de haz movible que sigue al usuario deseado. El proceso de combinar señales y luego focalizarlas en una dirección particular es referida como conformador de haces digitales (digital beamforming)

El mayor desafío para aplicar sistemas de antenas inteligentes en comunicaciones inalámbricas es la gran cantidad de tráfico y el tiempo disponible para el cálculo complejo involucrado. Sin embargo, la llegada de poderosos y económicos procesadores digitales y el desarrollo de técnicas basadas en software han hecho que los sistemas de antenas inteligentes sean una realidad en sistemas de comunicaciones celulares.

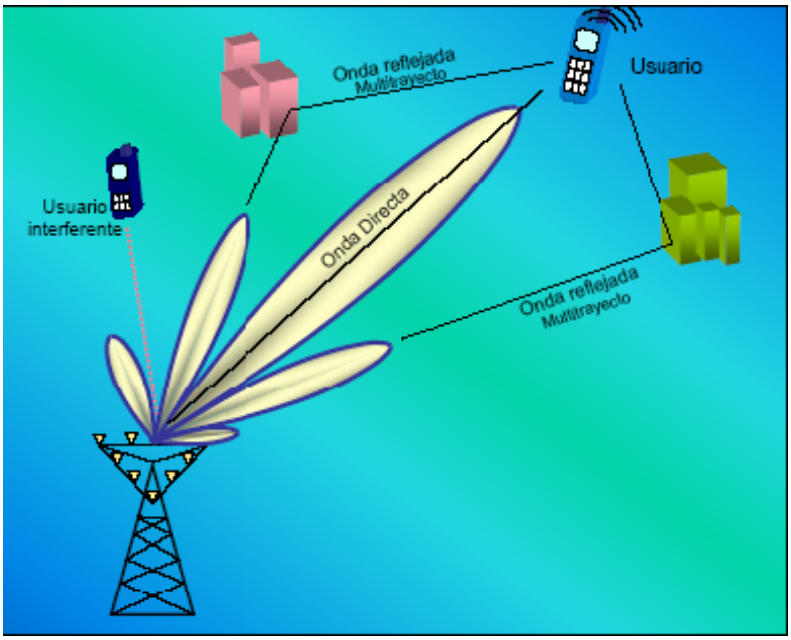


Figura 2.1 Situación real de comunicación con terminal móvil.

El conformador de haces adaptativos es una técnica que rige a los arreglos de antenas para lograr una recepción máxima en una dirección específica, estimando la señal de llegada desde la dirección deseada (en presencia de ruido), mientras que las señales de igual frecuencia proveniente de otras direcciones son rechazadas.

La separación espacial se utiliza para diferenciar la señal deseada de las señales interferentes. A través de la ponderación de las señales desde cada antena, se filtran las señales no deseadas. En conformadores de haces adaptativos, los pesos óptimos se calculan iterativamente usando algoritmos complejos.

En la literatura se encuentra una amplia gama de algoritmos. Este estudio se enfocó a los algoritmos adaptativos, dadas sus amplias posibilidades de aplicación.

Los algoritmos adaptativos pueden ser clasificados en categorías, basándose en diferentes técnicas.

Las técnicas basadas en información, clasifican los algoritmos de la siguiente manera:

Algoritmos basados en la señal de referencia.

Algoritmos adaptativos ciegos.

2.1 Algoritmos adaptativos basados en la señal de referencia.

2.1.1 Concepto de Algoritmos adaptativos basados en la señal de referencia.

Este tipo de algoritmos se basa en la minimización del error mínimo cuadrado entre la señal recibida y la señal de referencia. Por lo tanto, se requiere que la señal de referencia esté disponible. La señal de referencia tiene una alta correlación con la señal deseada, por ejemplo, el algoritmo

SMI (Simple Matriz Inversión), LMS (Least Mean Square) y RLS (Recursive Least Squares).

La señal de referencia no es la verdadera señal deseada, de hecho es una señal que la representa de cerca o tiene una alta correlación con ésta. La señal de referencia, requerida por el algoritmo, es generada de varias maneras.

2.1.2 Clasificación de los algoritmos basados en la señal de Referencia.

Existen algunos algoritmos que se basan en la señal de referencia. Los más conocidos son los algoritmos LMS y RLS. La respuesta a estos algoritmos será medida por parámetros como la convergencia, la rapidez y error del sistema. Además, se diseñará un algoritmo híbrido entre los dos anteriormente analizados. Además existen otros algoritmos entre los más conocidos tenemos.

- El filtro de Wiener

- El método de máxima pendiente

- El algoritmo LMS

• Los métodos de mínimos cuadrados y de mínimos cuadrados recursivos (RLS)

• Combinación RLS/LMS

• Igualación (no ciega) de canal

Algoritmos: LMS, RLS, modo guiado por decisión (DDE)

Estructuras: DFE, FSE

• DMI

• Filtros de kalman

• Algoritmo MUSIC

• Algoritmo SPRIT

2.2 Algoritmos adaptativos ciegos

2.2.1 Concepto de algoritmos adaptativos ciegos.

Estos algoritmos no requieren señal de referencia, generan por sí mismos la señal de referencia requerida desde las señales receptadas, para obtener así la señal deseada.

En este tipo de técnicas, más complejas, lo que se explota es alguna característica conocida de la señal deseada, como alguna modulación, algún tipo de ciloestacionariedad, etc.

Las técnicas de igualación ciega, no requieren una fase diferenciada de entrenamiento, sino que utilizan el conocimiento a priori sobre la estadística de la señal transmitida. Si bien es posible revertir también en este caso a un modo dirigido por decisión, el uso de técnicas de igualación completamente ciegas presenta las siguientes ventajas:

- Permite emplear protocolos de comunicación más sencillos, por ejemplo, eliminando la necesidad de retransmitir secuencias de entrenamiento tras desvanecimientos en el canal.

- Disminuye los problemas de interoperabilidad entre equipos, derivados del uso de distintas secuencias de entrenamiento.
- Ahorra ancho de banda en redes de difusión ya que evita intercalar periódicamente secuencias de entrenamiento.

A continuación se describen algunas de las técnicas ciegas que pueden utilizarse para la igualación de canal. Consideraremos únicamente el uso de constelaciones con Modulación en Cuadratura (QAM), ya que estas son habitualmente una opción muy recomendable en sistemas de alta capacidad. Sin embargo, la extensión de nuestros argumentos a sistemas que emplean otras modulaciones es inmediata. A la hora de trabajar con modulaciones QAM resulta cómodo emplear modelos equivalentes en banda base, por lo que en adelante consideraremos el caso general en que las señales procesadas por los filtros son complejas.

2.2.2 Clasificación de los algoritmos adaptativos ciegos.

• Igualación ciega:

Algoritmos Bussgang: Sato, CMA (Godard)
Algoritmos basados en algoritmos de orden superior (HOS)
Algoritmos basado en estadísticos de segundo orden (SOS)
Algoritmos basados en criterios de teoría de la información

• Otra clasificación:

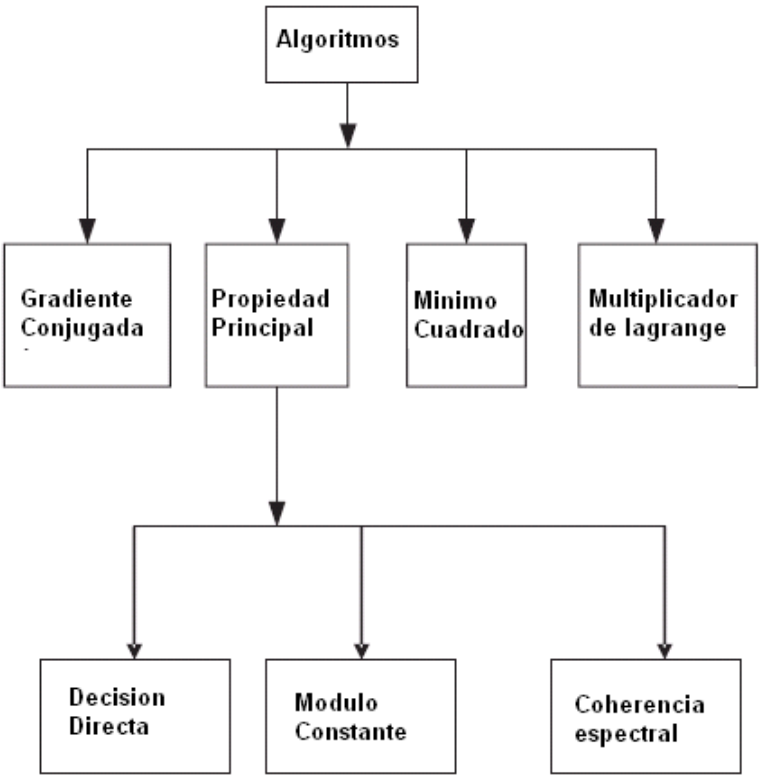


Figura 2.2 clasificación de los algoritmos adaptativos

A continuación detallamos los diversos tipos de algoritmos con sus ventajas y desventajas con sus respectivas ecuaciones de pesos.

Algoritmo	Ecuaciones de Pesos	Ventajas	Desventajas
Mínimo Cuadrado (LMS)	$w(n+1)=w(n)+\mu x(n)[d*(n)-x^H(n)w(n)]$	Siempre converge	Requiere señal de referencia
Inversión de Matriz Directa (DMI)	$\hat{R}_{xx}=\sum_{i=N_1}^{N_2}x(i)x^H(i)$ $\hat{r}_{xd}=\sum_{i=N_1}^{N_2}d^*(i)x(i)$ $w=\hat{R}_{xx}^{-1}\hat{r}_{xd}$	Siempre converge (mas rápidamente que LMS)	Requiere señal de referencia. Complejidad Computacional
Mínimo Cuadrado Recursivo (RLS)	$\hat{w}(n)=\hat{w}(n-1)+q(n)\left[d^*(n)-\hat{w}^H(n-1)x(n)\right]$ $q(n)=\frac{\gamma^{-1}R_{xx}^{-1}(n-1)x(n)}{1+\gamma^{-1}x^H(n)R_{xx}^{-1}(n-1)x(n)}$ $R_{xx}^{-1}(n)=\gamma^{-1}[R_{xx}^{-1}(n-1)-q(n)x(n)R_{xx}^{-1}(n-1)]$	Siempre converge ~10 veces mas rápido que LMS	Requiere señal de referencia y estimación inicial de R_{xx}
Algoritmo de Modulo Constante (CMA)	$w(n+1)=w(n)-\mu x(n)\varepsilon^*(n)$ $\varepsilon(n)=[1-\lvert y(n)\rvert^2]\,y(n)x(n)$	No requiere señal de referencia	Teóricamente puede no converger
MUSIC	$P_{MUSIC}(\theta)=\frac{1}{[a(\theta)]^H[P_N][P_N^H][a(\theta)]}$	Determina todas las Direcciones de arribo	Complejidad computacional
BKP Backpropagation	Red Neural Doble Capa con Algoritmo BKP	Requiere patrones de entrenamiento sola-mente en la fase de aprendizaje	Requiere entrenamientos periódicos

Tabla 2.1 Cuadro comparativo de algunos algoritmos

De estos algoritmos solo hablaremos de algunos de ellos en especial de los más utilizados.

2.3 Conformador de haces adaptativos.

La técnica de haz adaptativo constituye el máximo nivel de inteligencia que se podría dar a un sistema de antenas. En este sistema, las salidas de cada elemento del

arreglo de antenas se ponderan con un factor de peso cuyo valor se asigna dinámicamente para conformar un diagrama de radiación que presente el haz principal hacia la posición del usuario deseado y los haces o lóbulos secundarios hacia las direcciones de las componentes de multitrayecto de la señal deseada y mínimos o nulos de radiación en las direcciones de las fuentes de interferencia.

Esta técnica requiere el uso de algoritmos (DoA) tanto para la detección de las señales de arribo e interferentes como para la optimización de los pesos que conforman el haz.

2.3.1 Estructura de los haces adaptativos

Filtrado espacial en los receptores.

El proceso de Filtrado Espacial de frecuencias se lleva a cabo en un array de antenas o sistema de antenas, que reciben señales las que están acompañadas con otras señales o interferencias y ciertas componentes de ruido.

Cada señal se caracteriza por un ángulo de llegada y por una banda de frecuencias. Es decir, tenemos algo así como una caja negra con N entradas y M salidas. El objetivo que tendrá el sistema es obtener en cada salida una y

sólo una de las señales deseadas, atenuando las interferencias, ruido y resto de las señales deseadas.

Supongamos que se tiene un arreglo lineal de sensores receptores o antenas ubicadas espacialmente sobre un sistema de ejes coordenadas y numerados positivamente; las señales (deseadas y ruido) inciden sobre cada sensor con un ángulo y frecuencia específico. Es decir, cierta señal si tiene un contenido de frecuencia diferente a las otras señales. Luego, la separación de ésta con otras señales se realiza mediante el filtrado y así se logra extraer del resto de las señales recibidas. En caso de que dos o más señales vienen con un contenido frecuencial igual o muy similar, el sistema de conformador de haz será capaz de discriminarlas en función del ángulo de incidencia o sea en función de una variable espacial; así el sistema estará filtrando cada señal espacialmente.

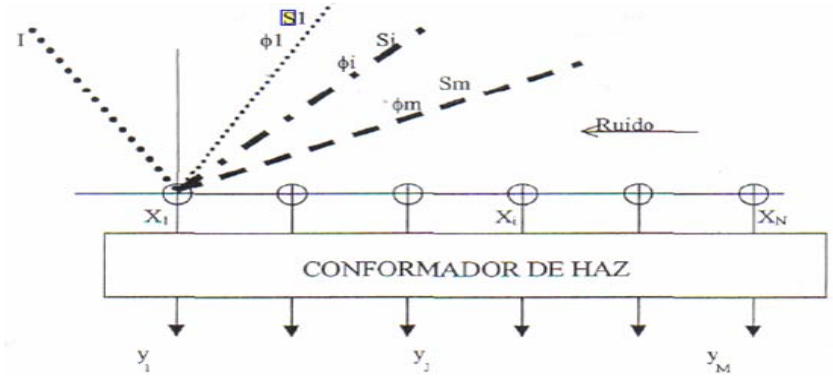


Figura 2.3 Discriminación de señales semejantes.

El beamformer o conformador procesa las señales recibidas en cada sensor y obtiene a partir de ellas una señal resultante. Como resultado se obtiene una antena con un diagrama de radiación que presenta un haz hacia la dirección de interés.

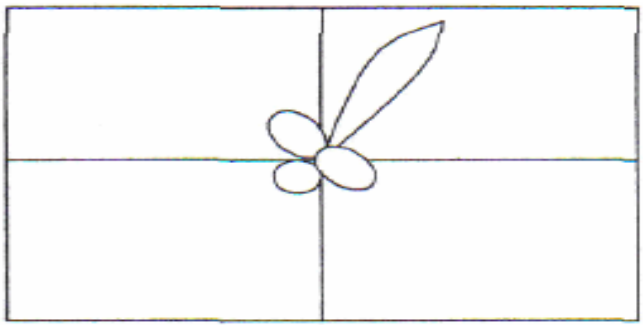


Figura 2.4 Lóbulo de radiación hacia la señal de interés.

Control del Diagrama de Radiación

El control del diagrama de radiación se hace por medio de un algoritmo adaptativo que pretende minimizar una señal de error generada a partir de una referencia en el receptor de la estación base. Para ello, es necesario definir un modelo de canal que tenga en cuenta todas las características del canal. El modelo de señal utilizado es el presentado en la ecuación (2.1), donde se consideran dispersión temporal y angular. El vector de señales recibidas en

cada elemento del array procedente de *N* usuarios con *Ntaps* multitrayectos puede expresarse como:

$$\underline{Y}(t) = \sum_{i=1}^{N} \sum_{l=1}^{Ntaps} c(\phi_i + \Delta\phi_{il})\alpha_{il} \exp(j2\pi v_i t) u_i(t - \tau_{il}) + \underline{N}(t) \quad (2.1)$$

Donde $\boldsymbol{\alpha_{il}}$, $\boldsymbol{\tau_{il}}$ y $\boldsymbol{\nu_{il}}$ son la amplitud, retardo y frecuencia Doppler del trayecto *l*-ésimo, $u_i(t)$ es la señal de datos en banda base del usuario *i*-ésimo, y *N(t)* es el ruido térmico; Φ_i es el ángulo incidente en azimut del trayecto principal del usuario *i*-ésimo y $\Delta\Phi_{il}$ es la dispersión acimutal respecto de Φ_i ; la respuesta del array viene dada por el steering vector de cada uno de los *M* elementos ecuación 2.2.

$$c(\phi) = [1 \quad \ldots \quad \exp(-j(M-1)\pi \sin\phi)]^T \quad (2.2)$$

La señal de referencia en el receptor se genera a partir de los códigos CDMA usados por los diferentes usuarios. Con esta señal obtendremos los pesos apropiados del array para reducir la interferencia recibida de usuarios no deseados y seguir a la señal deseada. Por tanto, se trata de un esquema de referencia temporal. Se supondrá que cada usuario utiliza códigos OSVF (Orthogonal Variable Spreading Factor). Gracias a la ortogonalidad entre los

códigos de los diferentes usuarios, puede generarse una señal de error *e(t)* cuya potencia se minimizará con el procesado adaptativo. Esta señal estará compuesta de la interferencia de otros usuarios y por ruido. Para cada usuario, se correrá la señal recibida con el código correspondiente, se filtra para eliminar la interferencia del resto de usuarios, y se vuelve a correlar la señal filtrada con el mismo código. Esta es la señal de referencia que luego se utilizará para obtener *e(t)*. El algoritmo genera un vector de pesos **w**, que minimizará la potencia del error *e(t)* en un número de iteraciones dado, con lo que el haz se conformará adecuadamente para maximizar la relación señal frente a interferencias (SIR).

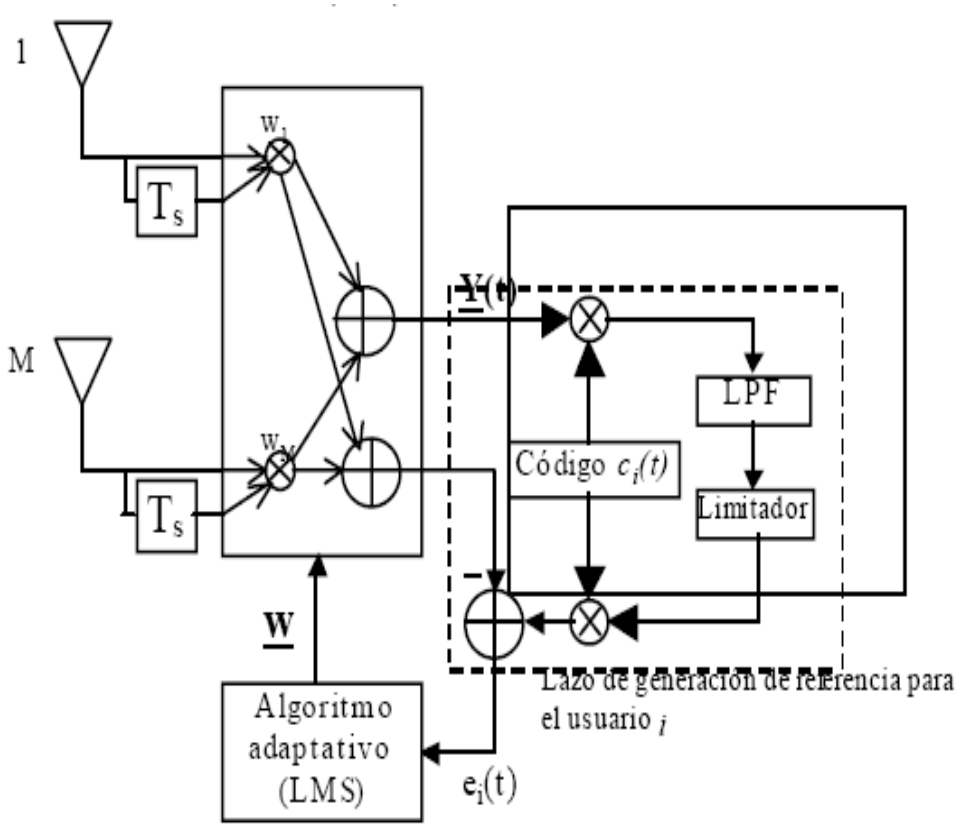


Figura 2.5 *Diagrama de bloques del receptor de la estación base y lazo de generación de la referencia.*

Como se había mencionado anteriormente, el conformador de haces adaptativos es una técnica que permite una máxima radiación hacia el usuario deseado y nulos en la dirección de señales interferentes. Para ilustrar diferentes aspectos del conformador de haces, se considera la configuración mostrada en la figura 2.6.

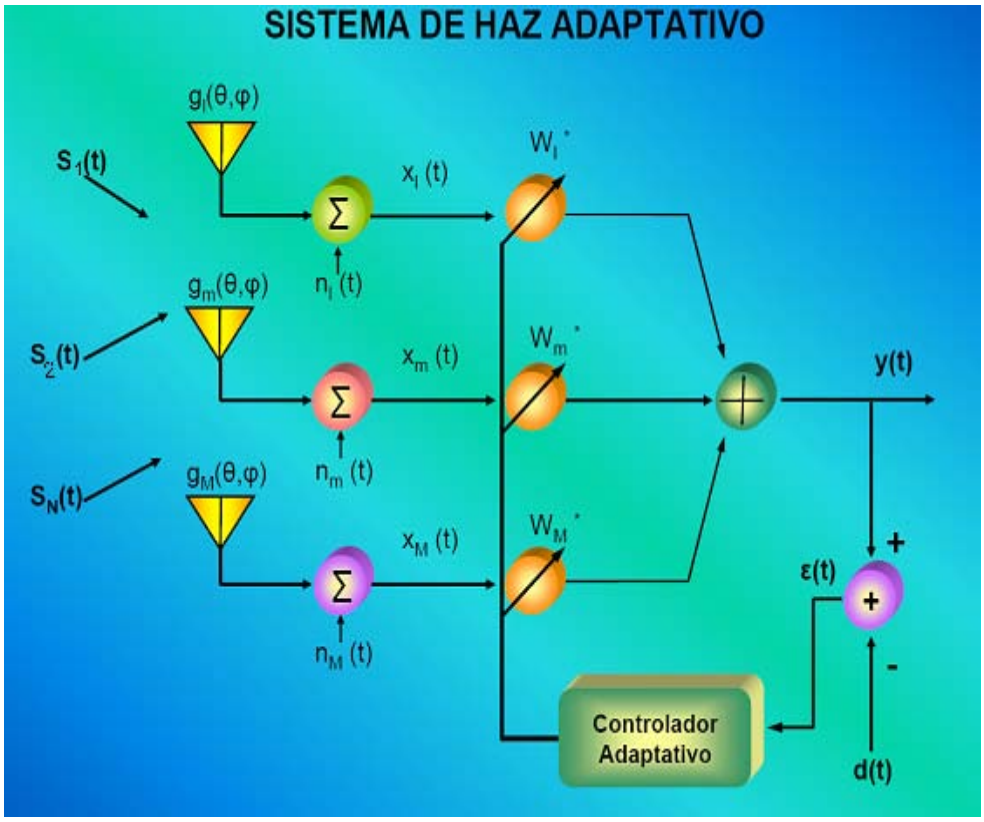


Figura 2.6 Sistema de arreglo adaptativo.

Las $s_N(t)$ representan las señales incidentes en los elementos de antenas. A estas señales se suma el ruido $n_M(t)$, para después ser ambos ponderados por el procesador de señales digitales controlado por un algoritmo adaptativo. De esta manera se obtiene en la salida la suma de las ponderaciones variables de cada elemento de antena, designada como $y(t)$. Los pesos w_m son calculados iterativamente basándose en la salida del arreglo *y(t)*, la señal de referencia *d(t)*, que es una aproximación de la señal deseada, y las ponderaciones pasadas. La señal de referencia se aproxima a la señal deseada usando

secuencias entrenadas o códigos de propagación, los cuales deben ser conocidos por el receptor. El formato de la señal de referencia varía y depende del sistema donde se implementa el conformador de haces adaptativos. La señal de referencia usualmente tiene una alta correlación con la señal deseada, el grado de correlación incluye en la exactitud de la convergencia del algoritmo. La salida del arreglo está dada por y(t) ecuación 2.3:

$$y(t) = w^H x(t) \tag{2.3}$$

Donde w^Hes la transpuesta conjugada compleja del vector de ponderación *w.*

El vector respuesta a los datos muestreados en la salida del arreglo está dado por x(t) ecuación 2.4:

$$x(t) = s(t)a(\theta_0) + \sum_{i=1}^{N_u} u_i(t)a(\theta_i) + n(t) \tag{2.4}$$

En la cual *s(t)* representa la señal deseada incidente al arreglo con un ángulo θ_0; $u_i(t)$denota a las $N_u$señales interferentes no deseadas que llegan al arreglo con un ángulo θ_i; $a(\theta_i)$ es el vector de propagación del arreglo de las

i-ésimas señales de interferencia y $a(\theta_0)$ es el vector de propagación del arreglo de la señal deseada.

El error cuadrático medio entre la salida del conformador de haz y la señal de referencia puede ser expresado de la siguiente manera (ecuación 2.5):

$$\varepsilon^2(t) = \left[d^*(t) - w^H x(t) \right]^2 \qquad (2.5)$$

Minimizando la ecuación (2.5) se obtiene el vector de ponderación o peso óptimo.

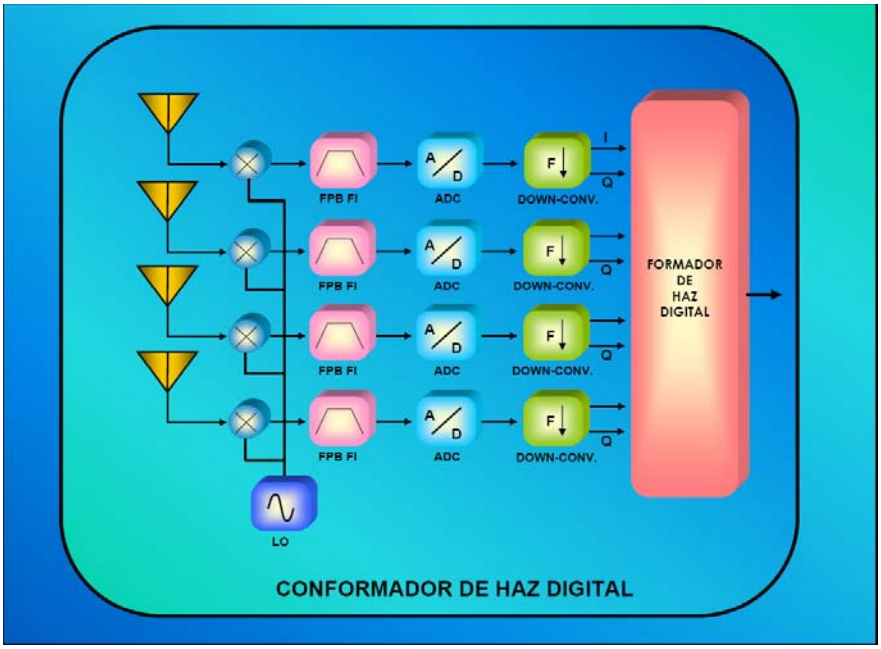


Figura 2.7a Conformador de haz digital.

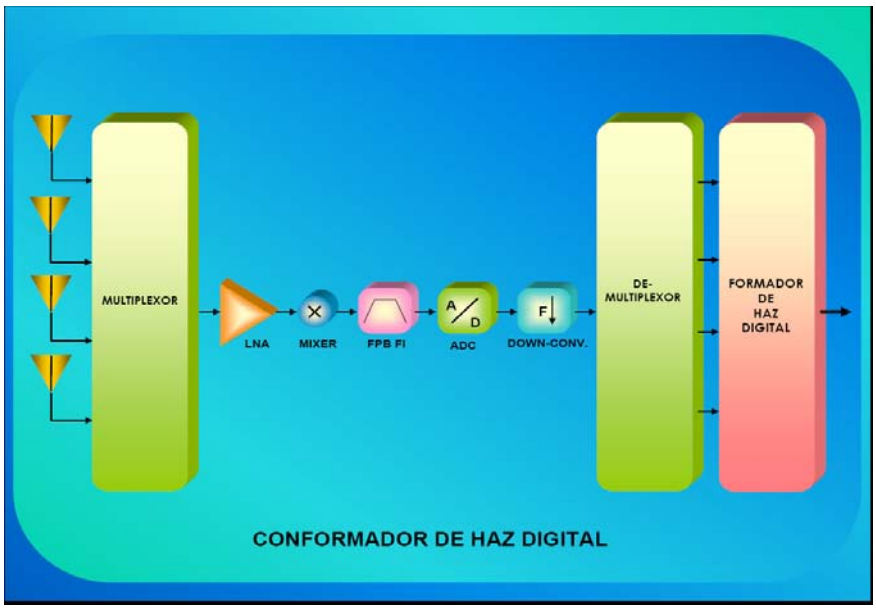


Figura 2.7b Conformador de haz digital.

2.3.2 Los haces adaptivos en las antenas inteligentes

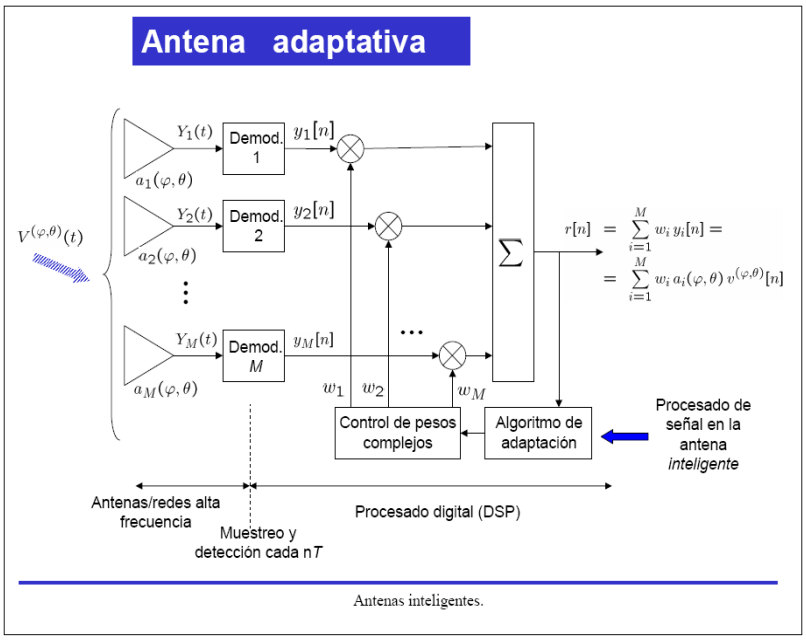


Figura 2.8 Antena adaptativa.

Las antenas adaptativas presentan varios tipos de señales, entre estas tenemos las siguientes:

Modelo de señales en la antena adaptativa, para ***una sola onda incidente***:

- Señal incidente en la antena, que lleva la información en la secuencia *v*[*n*] ecuación (2.6):

$$V^{(\varphi,\theta)}(t) = \sum_{n=-\infty}^{\infty} v^{(\varphi,\theta)}[n]\, g(t - nT) \qquad (2.6)$$

- Señal captada por cada antena *i* ecuación (2.7) :

$$Y_i(t) = a_i(\varphi,\theta)V^{(\varphi,\theta)}(t) \qquad (2.7)$$

- Señal demodulada en la rama *i*: ecuación (2.8) :

$$y_i[n] = a_i(\varphi,\theta)v^{(\varphi,\theta)}[n] \qquad (2.8)$$

- Señal recibida total, después del peso complejo *wi* y la suma de cada rama: ecuación (2.9) :

$$r[n] = \sum_{i=1}^{M} w_i\, y_i[n] = \sum_{i=1}^{M} w_i\, a_i(\varphi,\theta)\, v^{(\varphi,\theta)}[n] \qquad (2.9)$$

El objetivo del procesado de señal en la antena es conseguir los pesos *wi* óptimos de forma que se pueda extraer la máxima información de la señal del usuario deseado.

- Para cada usuario habrá un conjunto (vector) de pesos y un diagrama de radiación. Se trata de encontrar el óptimo.

- Los algoritmos adaptativos genéricos (que se utilizan en medicina) se desarrollan sobre el modelo más genérico., de ahí su nombre, donde varias señales de usuarios en distintas direcciones inciden en las antenas (con su ruido asociado)

Sobre el modelo anterior se haría el procesado de señal adaptativo en el DSP del receptor que proporcionaría los pesos óptimos:

- ***Punto de vista "antena"*:** los pesos de alimentación se eligen para que el diagrama de radiación "apunte" a la dirección deseada, mientras que se sitúan nulos en las direcciones de las interferencias.

- ***Punto de vista "procesado"***: la señales recibidas se combinan para que en la señal de salida se maximice (o minimice) cierto parámetro.

- Los criterios pueden ser de: máxima potencia, máxima relación señal ruido, minimización del error cuadrático medio (MMSE), minimización de la tasa binaria de error (MBER).

El procesado puede tener distintos objetivos:

- El control puede dirigirse a optimizar una señal conocida en algún aspecto: ***Referencia temporal o espacial***

- Cancelar determinadas señales no deseadas dejando pasar otras: ***Filtrado Espacial***

- Obtener información de las señales que llegan a la antenas: ***Detección de Angulo de Llegada***

2.3.3 Gráficos de haces adaptativos.

2.3.3.1 Conformador de Haz de Referencia Temporal (LMS)

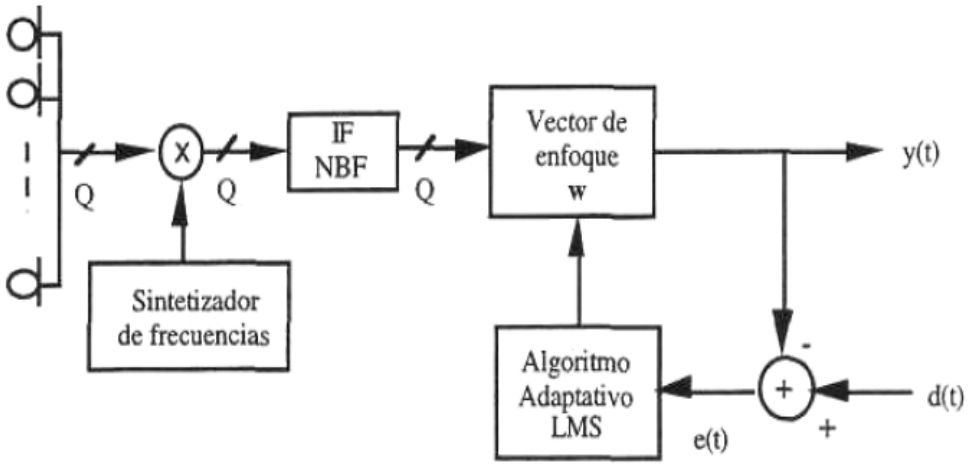


Figura 2.9 Conformador de haz de referencia temporal con algoritmo LMS para señales con modulación FH.

Los autores observaron diversos efectos adversos de la modulación de saltos frecuenciales sobre el conformador de haz de referencia temporal. La modulación FH produce una modulación de amplitud y de fase en la señal recibida por el array. Además, la SINR a la salida del array varía con el tiempo, disminuyendo con cada salto frecuencial y produciendo, por consiguiente, un aumento de la probabilidad de error de bit para la señal demodulada. La razón de

estos efectos es que los cambios en la frecuencia de la señal producen un desplazamiento de fase entre los elementos, equivalente al que producirían cambios en la dirección de incidencia de la señal a frecuencia fija. En consecuencia, justo después del salto frecuencial, el vector de pesos del array no posee las fases adecuadas para maximizar la respuesta deseada, produciéndose discontinuidades en la adaptación del algoritmo. A medida que los saltos en frecuencia son mayores, el vector de enfoque se encuentra más alejado del óptimo a la nueva frecuencia, necesitándose un transitorio mayor después de cada salto. De la misma forma, saltos frecuenciales elevados implican un mayor grado de modulación de amplitud y de fase, y un mayor descenso de la SINR con cada salto.

2.3.3.2 Conformador de Haz de MSINR (Algoritmo Maximin)

Bakhru y Tomen desarrollaron un algoritmo específico para arrays adaptativos utilizados en la recepción de señales con modulación FH, denominado "Maximim Algorithm", el cual se basa en la maximización directa de la SINR, ecuación 2.10:

$$\mathrm{SINR} = \rho_0 = \frac{P_d}{P_{nd}} = \frac{\mathbf{w}^H \mathbf{R}_d \mathbf{w}}{\mathbf{w}^H \mathbf{R}_{nd} \mathbf{w}} \qquad (2.10)$$

siendo Pd y Pnd las potencias de señal deseada y de señal no deseada, respectivamente. **R**d es la matriz de covarianza de la señal deseada y **R**nd es la matriz de covarianza de ruido más interferencias.

La regla de adaptación para el vector de enfoque óptimo, obtenida mediante el algoritmo de gradiente, es igual a:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu_0(n)\rho_0(n)\left[\frac{\mathbf{R}_d \mathbf{w}(n)}{P_d(n)} - \frac{\mathbf{R}_{nd} \mathbf{w}(n)}{P_{nd}(n)}\right] \qquad (2.11)$$

donde n indica un instante particular de muestreo y μ0(n) es un escalar real que regula la convergencia y estabilidad del algoritmo.

Esta regla de adaptación puede expresarse en función de los vectores de señal deseada **x**d y de señal interferente más ruido **x**nd a la entrada de los sensores del array y en función de las componentes de señal deseada yd(n) y no deseada ynd(n) a la salida del array, ecuación 2.12:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu(n)\left[\frac{E[\mathbf{x}_d(n)y_d(n)]}{P_d(n)} - \frac{E[\mathbf{x}_{nd}(n)y_{nd}(n)]}{P_{nd}(n)}\right] \quad (2.12)$$

considerando la constante de adaptación del algoritmo μ(n)=μ0(n)ρ0(n)

El valor de ρ0(n), así como los valores de potencia deseada **P**d(n) y no deseada **P**nd(n), son estimaciones realizadas en la iteración n. Los valores esperados E[**x**d(n)yd(n)] y E[**x**nd(n)ynd(n)] se estiman también en la iteración n, como promedios temporales durante un intervalo de tiempo finito.

Como se ha visto en la sección anterior, la aplicación del criterio de MSINR implica la separación entre las señales deseadas y las no

deseadas. Los autores proponen una posible discriminación entre la señal deseada y las señales interferentes y ruido, basada en las características espectrales de las señales con modulación FH (Figura 2.10). La señal recibida en cada uno de los sensores se aplica a dos filtros diferentes, uno paso banda "Band Pass Filter" (BPF) y otro de banda eliminada "Band Reject Filter" (BRF). El filtro paso banda se diseña con la banda de paso adecuada al contenido frecuencial de la señal deseada, la cual se obtendrá contaminada de interferencias y de ruido. Por otra parte, el filtro de banda eliminada posee dos bandas de paso simétricamente situadas a ambos lados de la banda frecuencial de la señal deseada, quedando ésta suprimida. La señal obtenida a la salida del filtro de banda eliminada será una estimación de la componente de señal no deseada a la salida del filtro paso banda, siempre que la interferente tenga un contenido espectral plano en la banda conjunta de los dos filtros. Una vez separadas la señal deseada de la no deseada, un conjunto de correladores se dedican a la maximización de La potencia a la salida del array debida a la señal deseada, mientras que otro conjunto de correladores minimizan simultáneamente la potencia debida a interferencias y ruido.

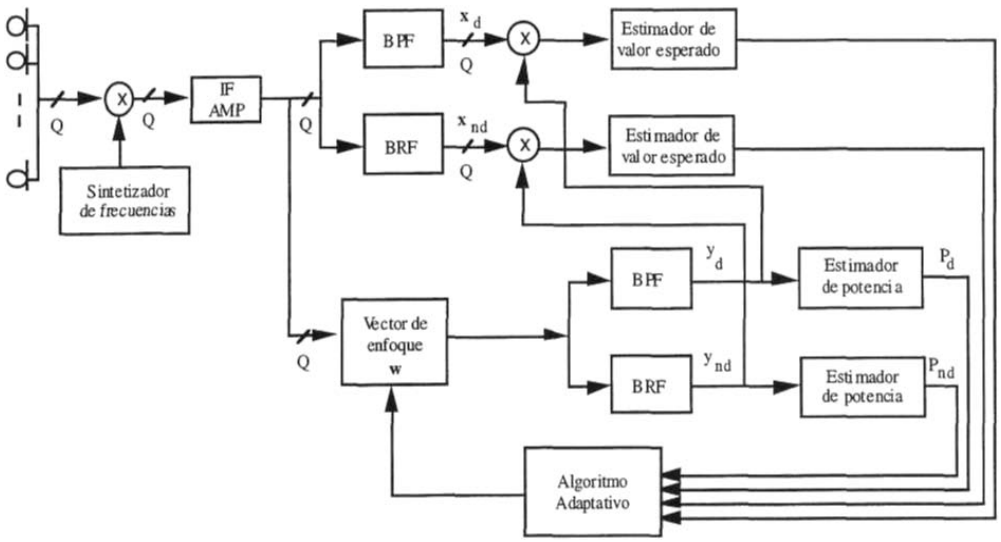


Figura 2.10 Conformador de haz de MSINR con algoritmo Maximin para señales con modulación FH.

Torrieri en 1987 propuso la utilización de diversas técnicas de compensación frecuencial, específicas para el algoritmo Maximin, a fin de evitar las discontinuidades producidas por la modulación FH en los arreglos adaptativos.

La primera de estas técnicas es el procesado dependiente de los parámetros "Parameter-Dependent Processing", que consiste en la aplicación de un filtro adaptativo tras cada uno de los elementos del array. Cada uno de estos filtros debe disponer de un número de parámetros ajustables suficiente para permitir la formación de

nulos en las direcciones de incidencia de las señales interferentes, para todas aquellas frecuencias a las que éstas presentan una potencia significativa. Los filtros adaptativos utilizados son líneas de retardo, debiéndose limitar el número de retardos a fin de reducir la complejidad del sistema y la velocidad de convergencia. Sin embargo, a menor número de retardos, menor será la reducción de las fluctuaciones de la SINR producida por los saltos frecuenciales que se consigue con el procesado dependiente de los parámetros.

La segunda técnica de compensación frecuencial presentada es el procesado espectral "Spectral Processing", que se basa en la división de la banda total de frecuencias en un número determinado de regiones espectrales y la adaptación independiente de los vectores de enfoque para cada una de ellas. Cada vez que se produce un salto frecuencial, el vector de enfoque a la frecuencia anterior se almacena en memoria, recuperándose el vector de enfoque para la nueva región espectral de la memoria, el cual se adapta durante el tiempo de duración del salto. Conceptualmente, esta técnica es de implementación sencilla; básicamente, consiste en la incorporación de una memoria en el procesador del algoritmo Maximin. Sin embargo, cuando se producen los saltos

frecuenciales aparecen unos efectos transitorios debidos a las memorias de los filtros. Para evitar estos transitorios se diseñan los filtros paso banda y banda eliminada para cada región espectral separadamente. En consecuencia, el número de regiones espectrales debe ser reducido a fin de no aumentar considerablemente la complejidad de la implementación. Además, la mejora alcanzada con esta técnica de compensación frecuencial no es significativa.

Finalmente, el procesado anticipado "Anticipative Processing" de Torrieri , consiste en iniciar la adaptación del vector de enfoque óptimo a una determinada frecuencia antes de que ésta sea transmitida, mediante la utilización de una réplica de la secuencia de saltos frecuenciales adelantada aproximadamente el tiempo de duración de un salto. Esta técnica de compensación frecuencial implica el duplicar todos los sistemas de procesado utilizados en el algoritmo Maximin. Es decir, mientras un procesador principal proporciona la respuesta del array, un procesador auxiliar se destina a la adaptación del vector de enfoque para la siguiente frecuencia de salto. Después de cada salto, el vector pesos óptimo estimado en el procesador auxiliar se transfiere al principal. A pesar de que el procesado anticipado parece ser la técnica más

interesante en cuanto a velocidad de convergencia, es la que proporciona peores resultados en cuanto a las fluctuaciones de la SINR.

2.3.3.3 Conformador de Haz de Referencia Espacial ("Cancelador de Lóbulos Laterales)

Eken en 1991 propuso un array adaptativo para su utilización con señales FH basado en la cancelación de lóbulos laterales. El sistema propuesto se compone de dos etapas diferentes (Figura 2.11), cada una destinada a la cancelación de un tipo de interferente.

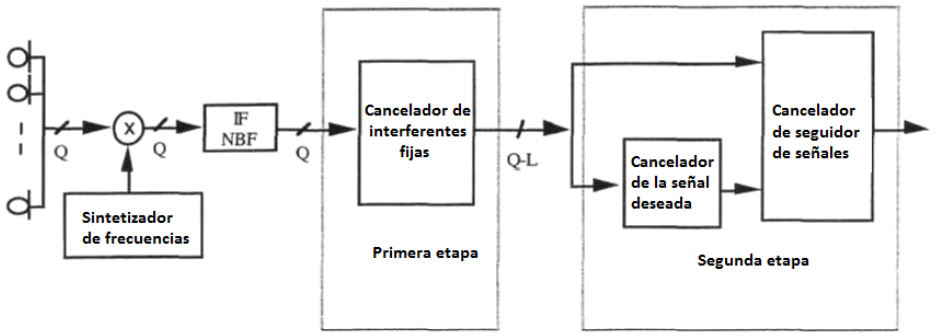


Figura 2.11 Conformador de haz de referencia espacial con cancelador de lóbulos laterales para señales con modulación FH (L interferencias fijas).

La primera etapa suprime las interferencias fijas, es decir, aquéllas que radian a una determinada frecuencia antes de que la señal deseada se transmita a esa frecuencia. Para evitar la posible cancelación de la señal deseada en esta etapa, los nulos en las direcciones de incidencia de las señales interferentes se colocan en el diagrama de radiación a una cierta frecuencia, antes de que se reciba la señal deseada a esa frecuencia. Este diseño anticipado requiere la presencia de dos sintetizadores de frecuencia que permitan obtener simultáneamente el escenario actual y el que se tendrá en el salto siguiente. La segunda etapa se dedica a la cancelación de "follower jammers", interferencias que pueden aparecer radiando a una determinada frecuencia una vez realizado el salto a esa frecuencia. Esta etapa se compone de dos canceladores, a la salida del primero se suprime la señal deseada, de forma que la señal obtenida con este cancelador puede utilizarse para monitorizar la aparición de nuevas señales interferentes. El funcionamiento de este sistema es, por tanto, equivalente al cancelador de lóbulos laterales clásico.

Los tres conformadores de haz para señales moduladas FH descritos anteriormente presentan una primera etapa común a todos ellos, previa al procesado adaptativo particular de cada sistema. Esta

etapa, conocida como "Dehopping", consiste en la conversión a frecuencia intermedia "Intermediate Frequency" (IF) de la señal recibida en cada sensor. Un sintetizador de frecuencias sincronizado con el utilizado en la generación de la modulación FH en transmisión, genera una secuencia de frecuencias igual a la de la señal recibida más la frecuencia intermedia. La mezcla de la salida del sintetizador de frecuencias, en cada uno de los saltos frecuenciales, con la señal a la salida del amplificador de bajo ruido "Low Noise Amplifier" (LNA) de cada sensor, produce la conversión de la señal deseada a IF; a continuación, esta señal se introduce en un filtro de banda estrecha apropiado al contenido frecuencial de la señal deseada. La existencia de esta etapa previa a la conformación de haz propiamente dicha, permite la reducción del ancho de banda de proceso necesario en el resto del sistema. De esta manera, únicamente el interfaz de radio, los LNA y los mezcladores deben utilizar tecnología de banda ancha, el resto de los circuitos del sistema pueden ser implementados con tecnología de banda estrecha, con la consiguiente simplicidad que ello conlleva. Además, el ruido y las señales interferentes fuera de la banda de paso de los filtros utilizados en la conversión son eliminados antes de que puedan introducirse en los sistemas adaptativos, simplificándose así la adaptación. En particular, cabe resaltar que la

implementación de la etapa de conversión a IF del sistema Maximin, a diferencia de la de los conformadores de referencia temporal y de referencia espacial, no reduce el ancho de banda de proceso al mínimo necesario (ancho de banda de la señal deseada). La razón es la necesidad de los filtros paso banda y de banda eliminada para la separación de las señales deseada y no deseada. En consecuencia, el ancho de banda de proceso para el algoritmo Maximin será como mínimo la suma de los anchos de banda de estos filtros.

2.3.3.4 Conversión a Frecuencia Intermedia Posterior

Lim y Widrow presentaron en 1984 un conformador de haz de dos sensores, diseñado como cancelador de lóbulos laterales, en el cual la conversión a frecuencia intermedia se realiza tras el procesado adaptativo, seguido de la demodulación de la señal deseada (Figura 2.12).

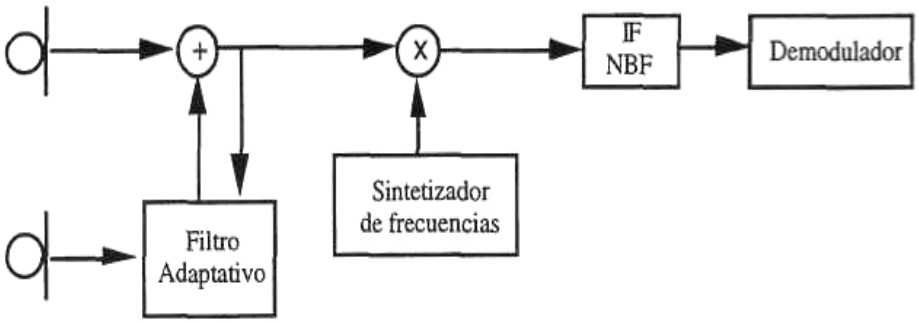


Figura 2.12 Conformador de haz con etapa de conversión a frecuencia intermedia posterior para señales con modulación FH.

Los autores desarrollaron distintos algoritmos adaptativos específicos para la implementación del cancelador de lóbulos laterales con señales moduladas FH. En el primero de ellos se utiliza un filtro adaptativo LMS en el dominio frecuencial, destinado a la minimización de la potencia a la salida del cancelador. Este filtro, equivalente a un banco de filtros, se implementa como una línea de retardos donde cada coeficiente controla la respuesta frecuencial en un rango estrecho de frecuencias. Para evitar la posible supresión de la señal deseada propia de los canceladores de lóbulos laterales, el coeficiente del filtro correspondiente a la subbanda frecuencial de la señal deseada en cada salto se mantiene fijo. El resto de los coeficientes son adaptados eliminando las posibles interferencias y minimizando el ruido. Este conformador de haz presenta diversos

problemas. En primer lugar, la cancelación de las interferencias dentro de la banda frecuencial de la señal deseada será posible si las señales interferentes presentan un espectro relativamente ancho que permita su estimación y anulación mediante la adaptación de los coeficientes correspondientes al resto de las subbandas. Sin embargo, si la interferencia es de banda estrecha, no podrá ser eliminada. Por otra parte, en entornos de modulación FH es habitual la existencia de un gran número de bandas frecuenciales distintas; en consecuencia, el filtro adaptativo en el dominio frecuencial necesitará un número muy elevado de retardos, resultando complicada su implementación. Además, el retardo asociado a este filtro puede llegar a ser insostenible dependiendo de la velocidad de salto.

Una segunda técnica adaptativa presentada por los mismos autores, consiste en la utilización de una versión modificada del algoritmo LMS "Filtered-X Filtered-ε LMS Algoritm" (Figura 2.13). La señal de entrada al filtro y el error o salida del cancelador se introducen en filtros de banda eliminada "Notch Filter" a la frecuencia de la señal deseada en cada salto. La presencia de estos filtros suprime la señal deseada a la entrada del sistema adaptativo, evitándose la posible cancelación de señal, así como los errores de inestabilidad

que pueden producirse en la adaptación de los coeficientes, debidos a niveles de señal elevado a la entrada del procesador adaptativo. La dificultad principal que presenta este conformador de haz es el diseño de los filtros de banda eliminada. Estos filtros deben modificarse con cada salto frecuencial, por tanto, es conveniente que presenten un transitorio reducido a fin de evitar errores. Por otra parte, se requiere una resolución elevada para estos filtros a fin de eliminar, lo más selectivamente posible, la banda de frecuencias de la señal deseada. Como consecuencia de estos dos requisitos para el funcionamiento correcto de los filtros de banda eliminada, aparece un compromiso entre el transitorio permitido y la resolución aceptable.

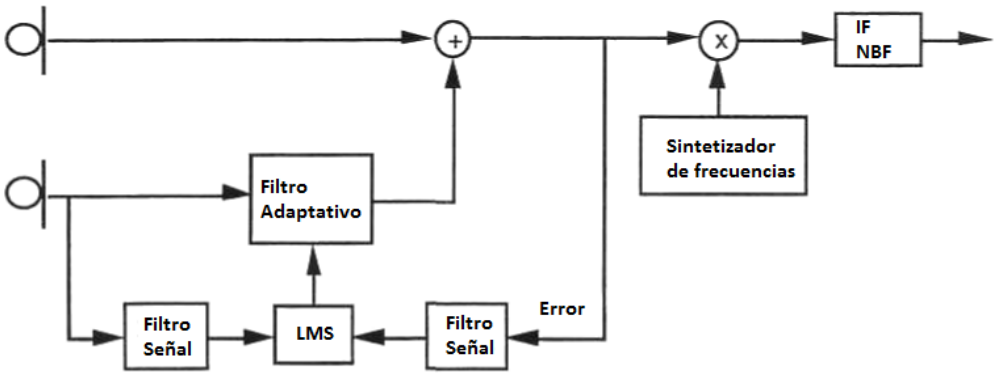


Figura 2.13 Conformador de haz con "Filtrado-X Filtrado-ε LMS Algoritmo" para señales con modulación FH.

Finalmente, Lim y Widrow propusieron otro cancelador de lóbulos laterales modificado para su utilización con modulación FH. Este nuevo conformador de haz se compone de dos procesadores en paralelo conocidos como maestro y esclavo "Master-Slave Adaptive Sidelobe Canceller" (Figura 2.14). A la entrada del procesador maestro se elimina la señal deseada mediante la aplicación de filtros de banda eliminada sintonizados a la frecuencia de salto. Igual que en el caso anterior, estos filtros evitan la posible cancelación de la señal deseada al adaptar los coeficientes del filtro adaptativo de este procesador según el criterio de mínima potencia a la salida. Una vez adaptados, estos coeficientes se transfieren al procesador esclavo, recuperándose la presencia de la señal deseada mientras que, simultáneamente, se suprimen las señales interferentes. La señal de salida del conformador de haz completo es la diferencia entre las salidas de los dos procesadores. De esta forma, se consigue minimizar las componentes residuales de señales interferentes que no hayan podido ser canceladas con el filtro adaptativo. La complejidad en la implementación de este sistema sigue siendo el diseño de los filtros de banda eliminada incorporados en el procesador maestro.

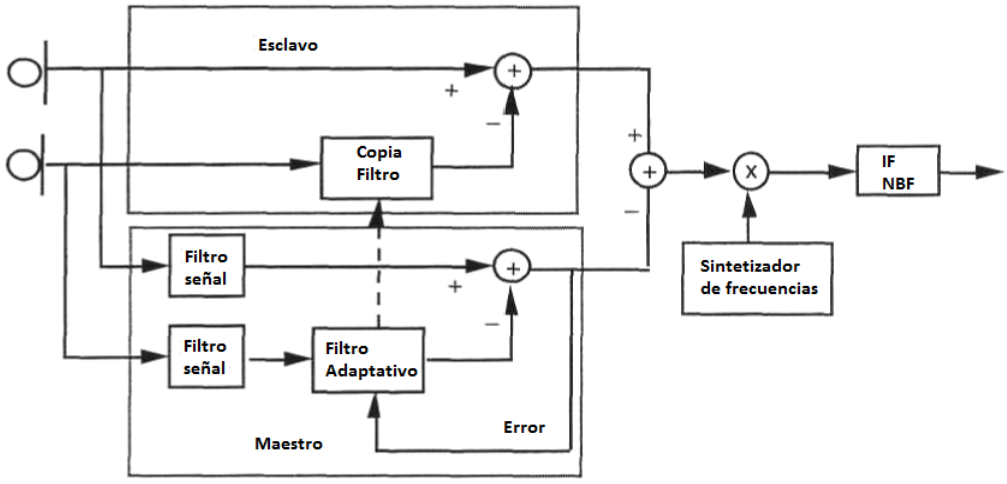


Figura 2.14 Conformador de haz "Cancelador Adaptativo Maestro-Esclavo" para señales con modulación FH.

2.4 Señal de arribo.

Las antenas inteligentes se presentan como una opción de gran interés dentro del campo de las comunicaciones móviles. Bajo el término antena inteligente se agrupan varios tipos de antenas, desde las antenas de haces conmutados hasta las antenas adaptativas, siendo en este último tipo en el que se centra esta comunicación.

En la actualidad existe un gran interés en el estudio y desarrollo de antenas adaptativas para comunicaciones móviles, en especial centradas en comunicaciones móviles UMTS, de inminente aparición en el mercado y que

consideran su uso. Los beneficios de estas antenas son la mejora en la relación señal a interferencia, y por tanto el incremento de la capacidad del sistema, aumentando con ello el número de usuarios permitidos en el sistema. La coexistencia de UMTS y GSM hace de gran interés la implementación de antenas adaptativas multi-estándar.

La presente comunicación se centra en la implementación mediante procesadores digitales de uno de los principales módulos en el procesado de señal de la antena inteligente: el módulo de conformación de la señal.

2.4.1 DESCRIPCIÓN DEL SISTEMA

Funcionamiento y esquema modular general

La idea de funcionamiento de una antena adaptativa se puede resumir como sigue. La utilización de una agrupación o *array* de elementos como antena permite, mediante la selección de los pesos adecuados, obtener un diagrama de radiación adaptado a cada situación. La señal recibida por las antenas se multiplica por los pesos de conformación.

A la salida se obtiene una señal en la que se maximiza la potencia de ciertos ángulos de llegada (los de los máximos del diagrama de radiación) y

se anulan otros (nulos en el diagrama de radiación). Idealmente se puede obtener un diagrama para cada usuario, asociando distintos pesos a cada uno. Utilizando los pesos adecuados se logra que el diagrama de radiación maximice la potencia en la dirección del usuario deseado y minimice las direcciones en que aparecen interferencias, maximizando de este modo la relación señal deseada a interferencia, C/I.

La antena adaptativa que se presenta se divide en dos módulos bien diferenciados: el módulo de radiofrecuencia, en el que se trata la señal de RF y FI, y el módulo de procesado de señal. En este último se realiza todo el procesado necesario para la conformación de la señal, partiendo de la señal de frecuencia intermedia ofrecida por el módulo de radiofrecuencia correspondiente. Se distingue entre la arquitectura modular para el enlace ascendente y la correspondiente al enlace descendente.

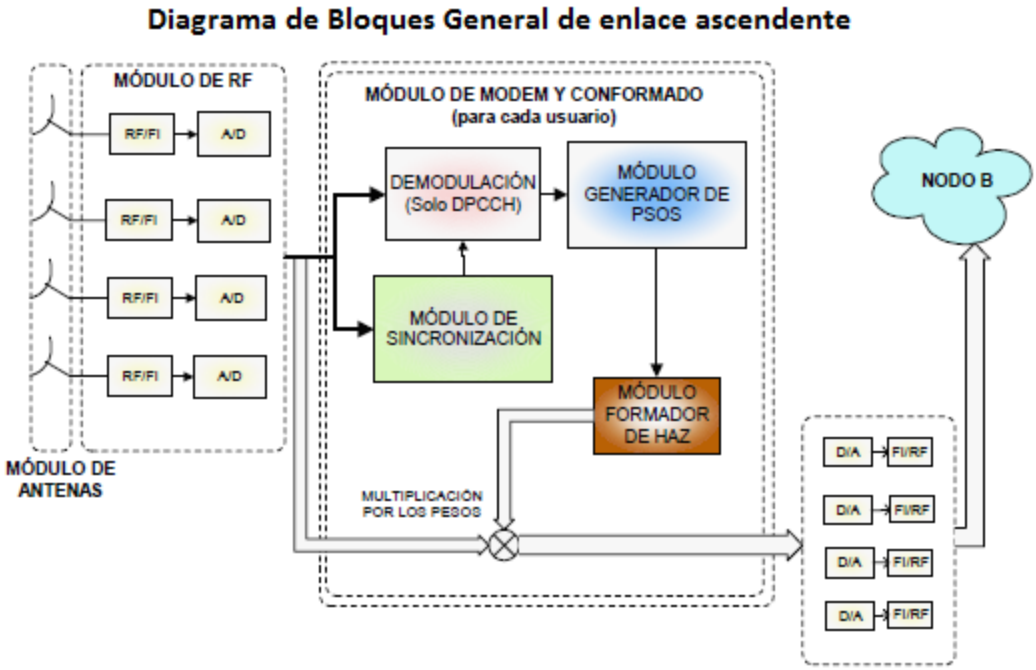


Fig. 2.15 Arquitectura modular de la antena adaptativa con la que se lleva a cabo la conformación. Enlace ascendente.

En las figuras 2.15 y 2.16 se presenta dicha arquitectura modular general. La antena adaptativa desarrollada se sitúa entre la estación base del operador y los equipos de usuarios móviles, ocupando por tanto el lugar ocupado actualmente por las antenas convencionales. El sistema propuesto consta de cuatro elementos en el arreglo.

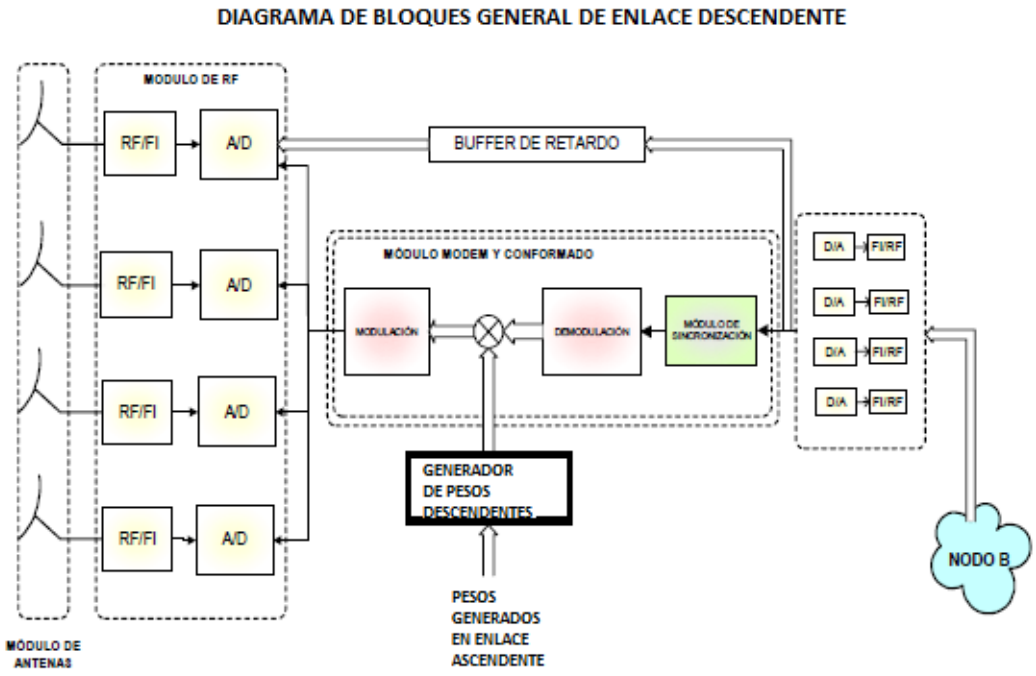


Fig. 2.16 Arquitectura modular de la antena adaptativa con la que se lleva a cabo la conformación. Enlace descendente.

El procesado de señal o *software radio* se ha dividido en varios módulos, que se pueden agrupar en dos grandes bloques: módulos de sincronismo y módulos de conformación.

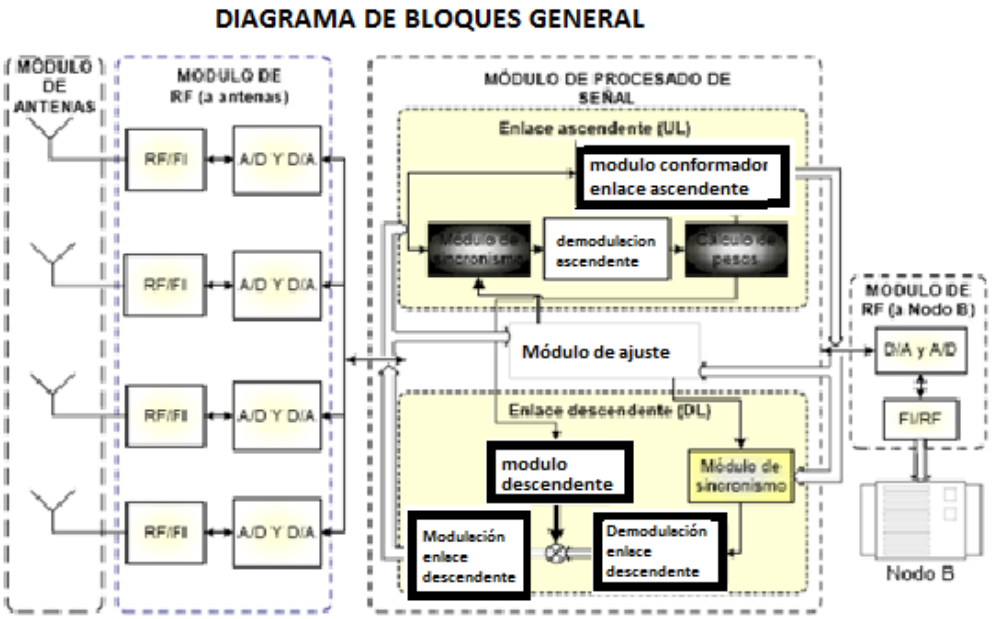


Fig 2.17 Esquema de bloques de la antena inteligente (los dos Módulos Unidos)

2.4.2 Conformación de la señal de arribo.

Módulo de conformación

El módulo de conformación de la antena adaptativa tiene como objetivo tratar la señal recibida, de modo que la señal obtenida a la salida del mismo presente una mejor relación señal a interferencia C/I. Para ello se conforma la señal multiplicándola por ciertos pesos, que maximizan la potencia de la señal en la dirección del usuario deseado y minimiza la potencia en la

dirección de las señales interferentes que puedan aparecer en el sistema. El diagrama de radiación cambia de forma adaptativa, variando los pesos para obtener los valores que optimizan la relación C/I. De este modo se logra seguir en tiempo real tanto las variaciones de los usuarios deseados como de las posibles interferencias que pueden afectar al sistema y que se desean reducir.

El módulo de conformación se divide a su vez en varios submódulos, cada uno de los cuales se encarga de una de las tareas necesarias para llevar a cabo la conformación. En las figuras 2.18 y 2.19 se presentan los esquemas de bloques del módulo de conformación, tanto para el enlace ascendente como para el descendente.

El módulo de conformación en uplink calcula los pesos de conformación a partir de los bits demodulados ofrecidos por el módulo de sincronismo y demodulación (MODEM), para cada usuario. Para ello, en primer lugar el submódulo de sincronismo de slot debe engancharse al slot correspondiente, ofreciendo a su salida los bits del canal de control que se utilizarán como referencia para realizar la adaptación continua de los pesos de conformación. Como bits de referencia se utilizan únicamente los bits piloto dentro del canal de control.

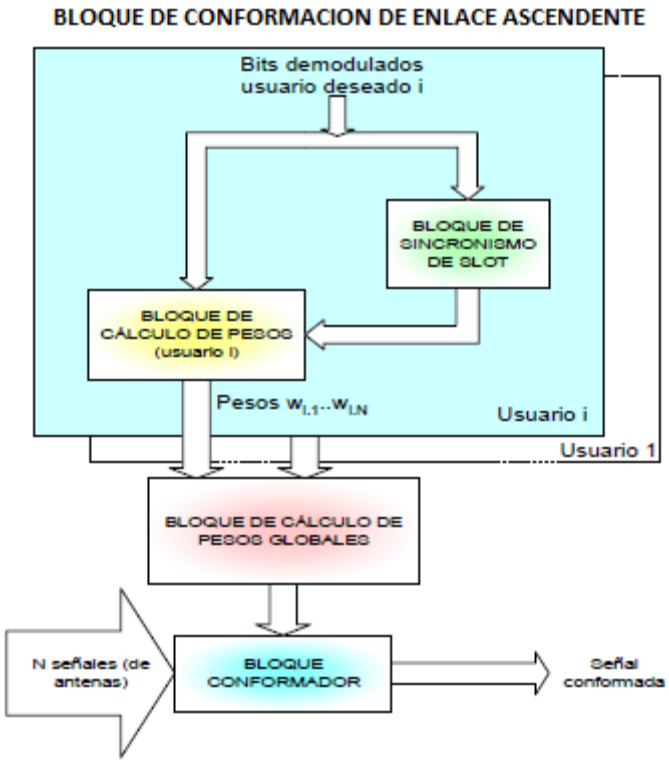


Fig. 2.18 Esquema de bloques del conformador. Enlace ascendente

El cálculo de los pesos de conformación para cada usuario se realiza de forma independiente. Se actualizan dinámicamente para cada bit recibido, mediante un algoritmo adaptativo. Como algoritmo adaptativo se ha elegido un sencillo algoritmo de mínimos cuadrados normalizado NLMS (Normalized Least Mean Square). La regla básica de actualización de los pesos de conformación, $\vec{w}(n)$, utilizando el algoritmo NLMS, es:

$$\vec{w}(n) = \vec{w}(n-1) + \frac{\mu}{\left|\vec{x}(n)\right|^2}\vec{x}(n)\cdot e^*(n) \qquad \textbf{(2.13)}$$

Donde $\vec{w}(n)$ son los pesos en el instante n, $\vec{w}(n-1)$ son los pesos en el instante anterior, $\vec{x}(n)$representa a la señal de entrada demodulada y $|\vec{x}(n)|^2$su potencia instantánea, y μes el paso de adaptación o *step size*. $e^*(n)$es el error conjugado, donde el error se calcula como la diferencia entre la señal de entrada multiplicada por los pesos y la señal de referencia. Ecuación 2.14:

$$e(n) = d(n) - y(n) = d(n) - \vec{w}^H(n-1) \cdot \vec{x}(n) \quad \textbf{(2.14)}$$

Gracias a su sencillez y bajo coste computacional este algoritmo permite actualizar los pesos de conformación a velocidad de bit, en tiempo real, si bien las características de velocidad de convergencia y error residual son peores que las de otros algoritmos más complejos.

Para reducir la carga computacional, y teniendo en cuenta que la señal ofrecida a la estación base o nodo B debe ser una única señal (y no una señal por usuario), se realiza la conformación para todos los usuarios deseados de forma simultánea, obteniéndose unos pesos globales que conforman toda la señal recibida.

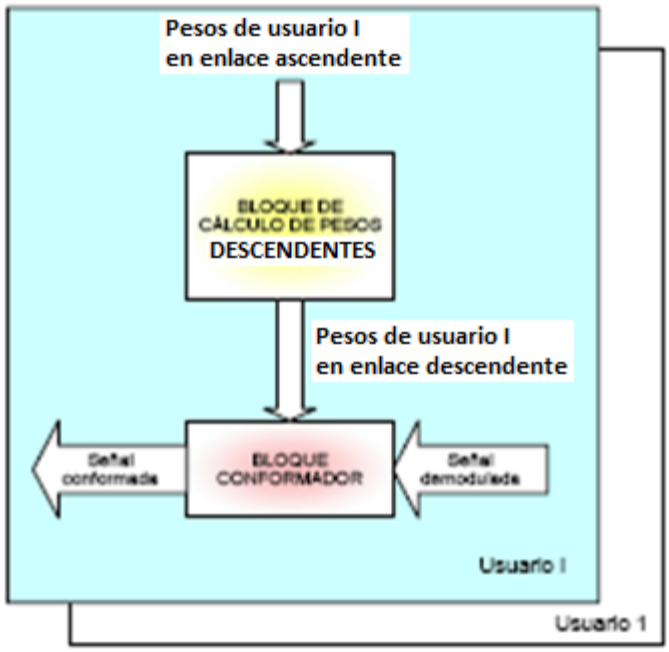


Fig. 2.19a Esquema de bloques del conformador. Enlace descendente

Esto equivale a realizar una cancelación parcial de interferencias en Uplink. El bloque de cálculo de pesos globales se encarga de obtener un conjunto único de pesos a partir de los pesos obtenidos para cada usuario. Por último el bloque conformador multiplica la señal recibida por los pesos correspondientes, obteniéndose finalmente una señal conformada, a nivel de chip.

Para el enlace descendente la conformación se simplifica, puesto que el cálculo de pesos de downlink no requiere de un nuevo cálculo mediante

algoritmo de adaptación, sino que se parte directamente de los pesos obtenidos en el enlace ascendente, y se corrigen para considerar las diferencias de frecuencias en ambos enlaces. El bloque de cálculo de pesos de downlink se encarga de realizar este cálculo. El bloque conformador multiplica la señal demodulada por los pesos conformando la señal y ofreciéndola al módulo modulador.

2.4.3 Direccionamiento de la señal de arribo

Un sistema de antenas adaptivo puede determinar la ubicación de un usuario porque:

–Recibe la señal del usuario a través sus sensores(elementos de la antena)

–La señal arriba a cada elemento de la antena en un tiempo diferente (retardo de tiempo)

–Su procesador digital de señales (DSP), es un procesador de señales especializado, calcula la dirección de arribo (DOA Direction-Of-Arrival) del usuario de los retardos de tiempo

–El DSP también suma las intensidades de las señales de cada elemento de la antena y forma el haz hacia la dirección calculada por el DOA.

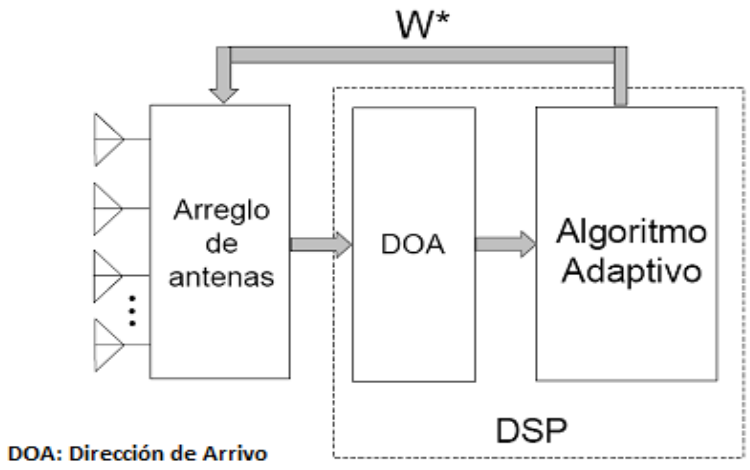


Fig. 2.19b Esquema de bloques del conformador. Enlace descendente

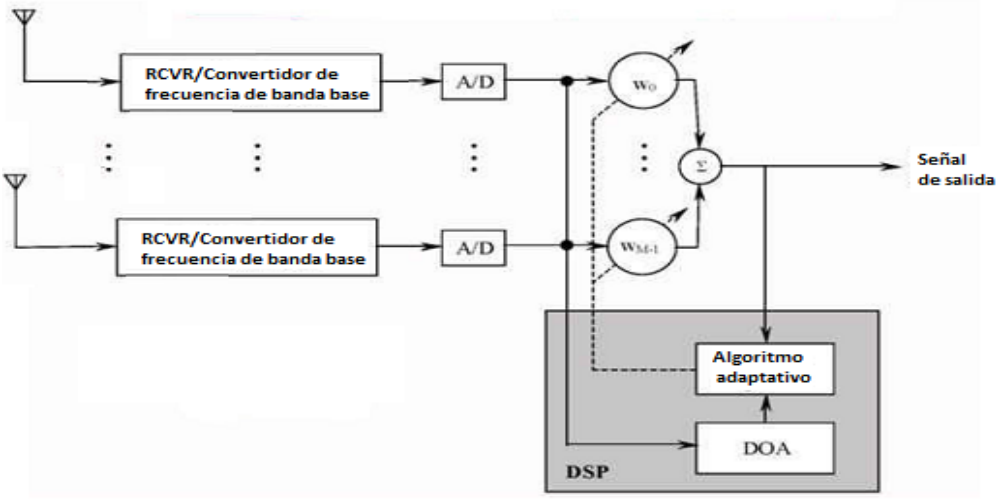


Fig. 2.19c Esquema de bloques del conformador. Enlace descendente

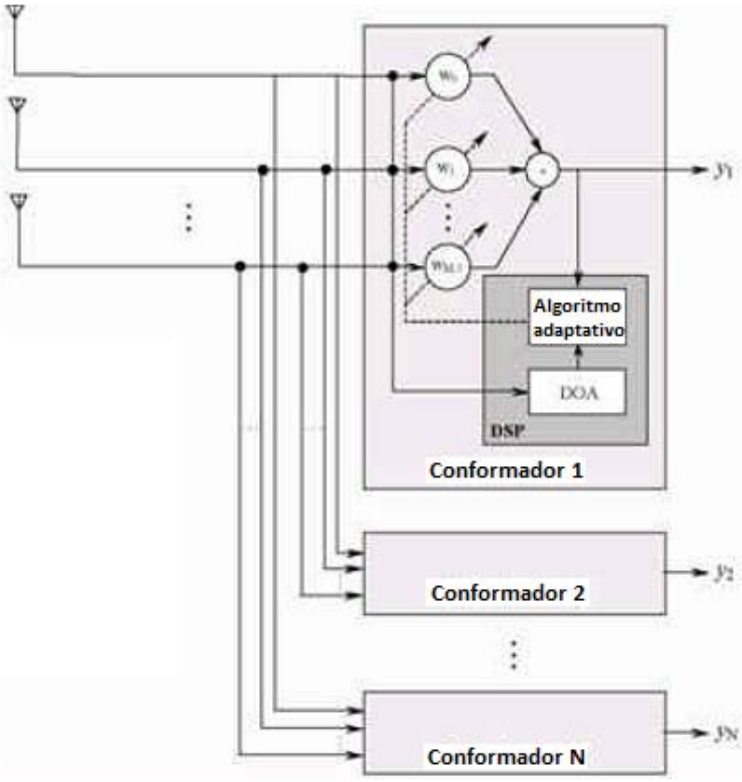


Fig. 2.19d Esquema de bloques del conformador. Enlace descendente

DIRECCION DE ARRIVO (DOA)

La determinación del arribo es muy importante para las antenas inteligentes, ya que con esta le permite a las antenas buscar la dirección de la señal, para esto se trata de determinar el punto donde se encuentra la señal en un instante dado y ver si esta en movimiento, siguiendo a la señal en su movimiento, cambiando de antena en caso de salir de su zona de influencia,

el desarrollo matemático de la dirección de arribo de la señal es ecuación (2.5)**:**

$\Delta t = (t1 - t2) = \Delta d/vo = d \cos(\Theta)/ vo$ despejando

$\text{Cos}(\Theta) = (vo/d)\,\Delta t = (vo/d)(t1-t2)$

$\Theta = \text{arcocos}((d/vo)\,\Delta t) = \arccos((d/vo)(t1-t2))$ (2.15)

Cálculos del retardo de tiempo

Cuando una onda que arriba incide con un ángulo θ sobre el arreglo de antenas, se producen retardos de tiempo relativos alrededor de los elementos de antena vecinos. Estos retardos dependen de:

La geometría de la antena y el espaciamiento entre los elementos

CAPÍTULO 3

EL ALGORITMO ADAPTATIVO MÍNIMO CUADRADO LMS

Analizaremos ampliamente al algoritmo LMS a lo largo de este capítulo, así como su relación con otros algoritmos adaptativos, los cuales garantizarán muchas aplicaciones que por sí solo no puede lograr el algoritmo LMS, el algoritmo LMS su mayor ventaja es su sencillez, tiene características lineales, lo que ayuda mucho a su comprensión se lo suele comparar con el programa Basic por su fácil entendimiento, se lo llama el Basic de los algoritmos adaptativos, por eso su gran popularidad. También tiene características aleatorias en su sistema de actualización de pesos, ya que cuando utiliza el llamado paso de adaptación utiliza una variable aleatoria y no un gradiente determinístico, el efecto es que los coeficientes del filtro pasan a ser una variable aleatoria cuya media es el filtro óptimo. También se dice que es prácticamente necesario y obligatorio aprender primero como funciona el algoritmo LMS antes de estudiar otros tipos de algoritmos adaptativos, porque es como una secuencia lógica de entendimiento, primero se debe de saber sumar y restar antes de multiplicar y dividir. Pero también presenta dificultades ya que el algoritmo LMS tiene problemas al

necesitar mayor número de iteraciones para llegar a la convergencia y a las soluciones de los problemas, tiene una convergencia muy lenta, esta lentitud suelen compensarla utilizando ordenadores más rápidos, otro problema que se presenta es que necesitan mayor numero de coeficientes o parámetros para funcionar, lo cual es un limitante, también su cálculo del error es inferior al de otros algoritmos adaptativos.

3.1 EL ALGORITMO LMS.

El algoritmo LMS fue creado por el profesor de la universidad de Stanford Bernard Widrow y su primer estudiante de doctorado Ted Hoff en 1959, este algoritmo utiliza la aproximación estocástica para el cálculo de la gradiente de la función MMSE (Minimum Mean Square Error) mínimo error cuadrático medio. El método de minimización de la función de cálculo de gradiente se conoce como descenso por gradiente (steepest descent) lo que significa que el error cuadrático medio mínimo siempre sigue la dirección tangente a la superficie, ya que de esta manera desciende más rápidamente. El LMS es un algoritmo iterativo que hace correcciones sucesivas de los pesos, los cuales permiten menores valores de error medios, el método de descenso por gradiente presenta una expresión iterativa para la actualización del vector de pesos.

El algoritmo LMS internamente utiliza el filtro de Wiener, el **filtro de Wiener** es un filtro que fue propuesto por Norbert Wiener en la década de 1940 y publicado en 1949. Su propósito es, utilizando métodos estadísticos, reducir el ruido presente en la señal de entrada de tal modo que la señal de salida del filtro se aproxime lo más posible (en el sentido cuadrático medio) a una señal deseada (sin ruido). El equivalente en tiempo discreto del trabajo de Wiener fue derivado independientemente por Kolmogorov y publicado en 1941. Por esto, la teoría es a veces referida como *teoría de filtrado de* ***Wiener-Kolmogorov***.

Entre las características que presentan el algoritmo LMS tenemos:

1.- En los algoritmos LMS se puede utilizar la solución de wiener holf para encontrar la matrix de inversión, no es requerido la matrix de autocorrelación en los filtros de entrada.

2.- En su forma simple tiene múltiples aplicaciones e implementaciones, ya que junto con otros algoritmos tiene múltiples variantes en que se presentan nuevas características hibridas, que pueden ser muy útiles en algunos casos.

3.- Tiene procedimiento iterativo, valores de entrada y de salida, utilización de filtros, generación de un valor estimado de error, ajuste de pesos.

4.- La correlación de términos necesita encontrar el valor de los coeficientes n+1, estas iteraciones contienen el producto estocástico x(n)e(n). Los algoritmos LMS presentan métodos de valores descendentes.

5.- En el algoritmo LMS cada coeficiente puede tener diferentes valores de ruido.

6.- El algoritmo LMS incluye el parámetro de paso step size, el valor de u, que se utilizan para el control y cálculo de la convergencia, estabilidad y robustez del algoritmo.

3.1.1 Base teórica del algoritmo adaptativo LMS

El algoritmo adaptativo mínimo cuadrado LMS utiliza la aproximación estocástica para el cálculo del gradiente, para la determinación de las siguientes iteraciones y del error. Se calcula el gradiente de la función de costo MMSE (mínimo error cuadrático medio)., lo que significa que el error cuadrático medio mínimo siempre sigue la dirección tangente a la superficie, ya que de esta manera desciende más rápidamente.

El algoritmo LMS utiliza el conformador de haces adaptativos el cual es una técnica que permite una máxima radiación hacia un usuario deseado y nulos en la dirección de las señales interferentes, figura 3.1 (sistema de arreglo adaptativo).

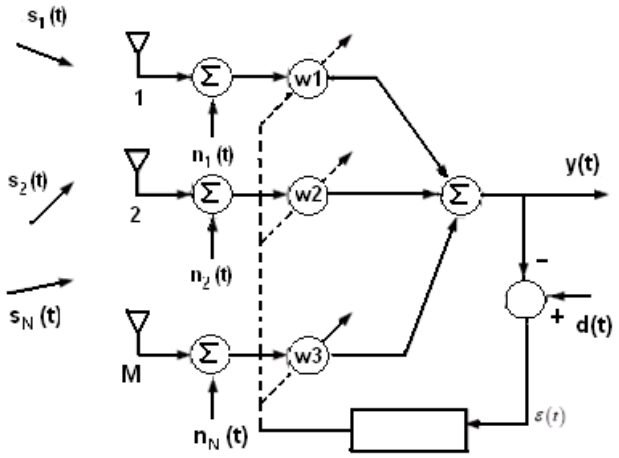


Figura 3.1 Sistema de arreglo adaptativo

Las $s_N(t)$ representan las señales incidentes en los elementos de antenas. A estas señales se suma el ruido $n_N(t)$, para después ser ambos ponderados por el procesador de señales digitales controlado por un algoritmo adaptativo. De esta manera se obtiene en la salida, la suma de las ponderaciones variables de cada elemento de antena, designada como y(*t)*. Los pesos wm son calculados iterativamente basándose en la salida del arreglo y(t), la señal de referencia *d(t)*, que es una aproximación de la señal deseada, y las ponderaciones pasadas. La señal

de referencia se aproxima a la señal deseada usando secuencias entrenadas o códigos de propagación, los cuales deben ser conocidos por el receptor. El formato de la señal de referencia varia y depende del sistema donde se implementa el conformador de haces adaptativos. La señal de referencia usualmente tiene una alta correlación con la señal deseada, el grado de correlación influye en la exactitud de la convergencia del algoritmo.

La salida del arreglo está dada por la ecuación 3.1

$$y(t) = W^{H}x(t) \tag{3.1}$$

Donde W^{H} es la transpuesta conjugada compleja del vector de ponderación. W

El vector respuesta a los datos muestreados, en la salida del arreglo, esta dado por :

$$x(t) = s(t)\, a(\theta_0) + \sum_{i=1}^{N_n} ui(t)a(\theta_i) + n(t) \tag{3.2}$$

En la cual, s(t) representa la señal deseada incidente al arreglo con un ángulo θo, ui(t); denota a las .Nu señales interferentes no deseadas que llegan al arreglo con un ángulo θi; a (θi) es el vector de propagación del arreglo de las i-ésimas señales de interferencia y *a(*θo) es el vector de propagación del arreglo de la señal deseada.

El error cuadrático medio a la salida del conformador de haz y de la señal de referencia se lo expresa con la ecuación 3.3

$$\varepsilon^2(t) = [\, d^*(t) - W^H x(t) \,]^2 \qquad (3.3)$$

Todo este análisis matemático es para llegar a la expresión iterativa para la actualización de pesos (ecuación 3.4).

Expresión iterativa para la actualización del vector de pesos.

$$w(k+1) = w(k) - \frac{1}{2}\mu\nabla\,(E\{e^2(k)\}) \qquad (3.4)$$

El algoritmo LMS es una implementación del método de gradiente, en la cual se generan diferentes valores de X(n) hasta obtener un porcentaje de error pequeño.

x(t) es el vector respuesta a los datos muestreados a la salida de un arreglo en función del tiempo.

X(n) es el vector respuesta para valores discretos.

Con estos valores obtenemos la gradiente de la función de costo

$$\nabla (E\{e^{2}(k)\}) = -2x(k)e^{*}(k) \qquad (3.5)$$

Cuando reemplazamos esta gradiente ecuación 3.5 en 3.4 se obtiene la forma generalizada del algoritmo LMS. Ecuación 3.6, esta forma generalizada del algoritmo LMS también se la llama función de control o ecuación general de los algoritmos LMS.

Ecuación general de los algoritmos LMS (3.6)

$$w(k+1) = w(k) + \mu_{LMS}\, x(k)e^*(k) \qquad (3.6)$$

μ_{LMS} es un parametro constante

Esta ecuación general de los algoritmos LMS es la ecuación más importante dentro del estudio de los algoritmos LMS. Fíjense en la semejanza con la expresión iterativa (ecuación 3.4). Todo análisis matemático posterior solo servirá para aclarar más esta ecuación, toda ecuación dependerá en algún grado de esta. Como todo procedimiento práctico de gradiente, el LMS (Least Mean Square algorithm) toma una estimación del gradiente teórico, si se desea profundizar más este análisis para valores discretos vea el anexo A.

3.1.2 Análisis matemático del algoritmo LMS

A continuación expondremos el análisis matemático del algoritmo LMS, comenzaremos hablando de la regresión lineal en forma matemática, la cual es usada por el algoritmo LMS para hallar los valores de los pesos, y de su optimización mediante mínimos cuadrados**.** Hasta llegar a la formula general del algoritmo LMS.

Que es una regresión lineal, en su forma más simple consideramos una muestra de tamaño n, i=1,2,3,4….,n. y consideraremos que hay 2 variables, una variable independiente x, y una variable dependiente y, en donde y depende de x. cuando tomamos una muestra aleatoria de valores para x y para y, podemos determinar si y es dependiente de x.

Deseamos saber si y depende de x. Con estos valores hacemos un modelo matematico deterministico y=Bo+B1x, en donde Bo es la intersección de la curva, y B1 es la pendiente de la recta. Bo y B1 son parámetros a escoger. Y obtenemos la ecuación y= Bo + B1x + e, donde e es la variable aleatoria que tiene la misma distribución de y. al sacarle el valor esperado nos debe quedar que E(y) = Bo + B1x , E(e) = 0, si esto se cumple y depende linealmente de Bo, B1 y x.

Nosotros utilizaremos la herramienta matlab que determinar los parámetros desconocidos de propagación lineal del algoritmo LMS que es un método de minimos cuadrados. Y nos queda

.y*=a+bx, en donde a y b son estimaciones de Bo y B1. El método de regresión lineal nos dice la intercepción del eje y con el modelo matemático exacto Bo+B1x nos da un valor yi, y la intercepción del eje y con la curva del modelo de regresión lineal a+bx nos da yi*, se aplica

el llamado criterio de mínimos cuadrados que es una diferencia entre yi con yi*

.ei=yi+yi* de aquí usamos la formula de SCE (ecuación 3.7) , de hay derivando dSCE/dBo =0, dSCE/dB1=0 ,

$$SCE = \sum_{i=1}^{n} ei^2 \qquad (3.7)$$

$$SCE = \sum_{i=1}^{n} (yi - y^*) \qquad (3.8)$$

$$SCE = \sum_{i=1}^{n} (yi - a - bx)^2 \qquad (3.9)$$

Y al final obtendremos 2 valores (a y b) como vemos en la tabla II

Tabla II Ajuste de una ecuacion lineal por minimos cuadrados

x_i	y_i	$x_i y_i$	$(y_i - Y)^2$	$(y_i - a - bx_i)^2$
1	0.5	0.5	8.5765	0.1687
2	2.5	5.0	0.8622	0.5625
3	2.0	6.0	2.0408	0.3473
4	4.0	16.0	0.3265	0.3265
5	3.5	17.5	0.0051	0.5896
6	6.0	36.0	6.6122	0.7972
7	5.5	38.5	4.2908	0.1993
$\sum \rightarrow 28$	24	119.5	22.7143	2.9911

Se pueden calcular las siguientes cantidades

$n = 7$ $\quad \sum x_i y_i = 119.5$ $\quad \sum x_i 2 = 140$ $\quad \sum x_i = 28$

$\bar{x} = \frac{28}{7} = 4$ $\quad \sum y_i = 24$ $\quad \bar{y} = \frac{24}{7} = 3.428571429$

usando las ecuaciones $b = \frac{n\sum xy - \sum x \sum y}{n\sum x^2 - (\sum x)^2}$ $\quad$ **y** $a = y - bx$ **se tiene**

$b = \frac{7*119.5 - 28*24}{7*140 - 28\ 2} = 0.839285714$ $\quad a = 3.428571429 - 0.82928514*4 = 0.07142857$

Por lo tanto la ecuacion lineal con ajuste por minimos cuadrados es:

$y = 0.07142857 + 0.839285714x$

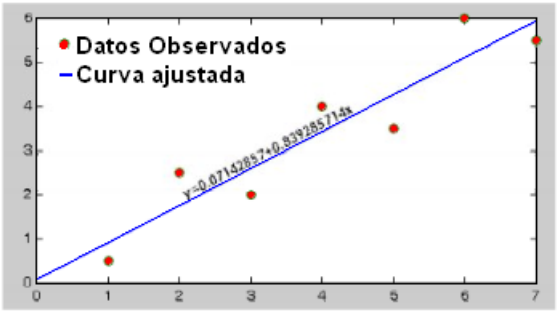


fig 3.2 ejemplo en matlab del ajuste de una curva

ejemplo con matlab:

```
x = [ 0.1, 0.4, 0.5, 0.7, 0.7, 0.9 ];
y = [ 0.61, 0.92, 0.99, 1.52, 1.47, 2.03 ];
c = polyfit(x,y,1)
c1 = x(1):0.1:x(length(x))
c2= polyval(c,c1)
plot(c1,c2);hold on
plot(x,y,'x')
axis([0,1,0,2.1])
xlabel('x')
ylabel('y')
```

En matlab hay comandos que resuelven la ecuación lineal y nos dan los valores de a y b.

Otro ejemplo

Se toma una muestra aleatoria de 8 ciudades de una región geográfica de 13 departamentos y se determina por los datos del censo el porcentaje de graduados en educación superior y la mediana del ingreso de cada ciudad, los resultados son los siguientes (Tabla III)

Tabla III Muestra aleatoria variable x vs y. variable x graduados, variable y ingresos por ciudad

CIUDAD	1	2	3	4	5	6	7	8
% de (x) Graduados	7.2	6.7	17	12.5	6.3	23.9	6	10.2
ingreso (y) Mediana	4.2	4.9	7	6.2	3.8	7.6	4.4	5.4

de las ecuaciones normales:

$$\sum y = na + b \sum x \qquad y \qquad \sum xy = a \sum x + b \sum x^2$$

$\sum y$, $\sum x$, $\sum xy$, $\sum x^2$ Por lo tanto se procede de la siguiente forma:

n	y	x	xy	x2
1	4.2	7.2	30.24	51.28
2	4.9	6.7	32.83	44.89
3	7.0	17.0	119.00	289.00
4	6.2	12.5	77.50	156.25
5	3.8	6.3	23.94	39.69
6	7.6	23.9	181.64	571.21
7	4.4	6.0	26.40	36.00
8	5.4	10.2	55.08	104.04
	43.5	89.8	546.63	1292.92

sustituyendo en las ecuaciones los resultados obtenidos se tiene:

43.50 = 8a + 89.8b (1)
546.63 = 89.8a + 1292.92b (2)

Para resolver el anterior sistema, se multiplica la primera ecuacion por 89.8 y la segunda ecuacion por 8 y de hay obtenemos:

43.50 = 8a + 89.80b * (-89.8) (1) 546.63 = 89.8a + 1292.92b * (8) (2)
3906.30 = -718.4a - 8064.04b 4373.04 = 718.4a + 10343.36b

rsolviendo el sistema de ecuaciones obtenemos

$$b = \frac{466.74}{2279.32} = 0.20477$$

este valor de b se reemplaza en cualquiera de las ecuaciones para obtener el valor de a:

Reemplazando b = 0.20477 en la primera ecuacion:

43.50 = 8a + 89.8 (0.20477), donde 43.50 = 8a + 18.3880, despejando a se obtiene:

a= 25.120 = 3.139
8

Se tiene entonces que los coeficientes de regresion son a = 3.139 y b = 0.20477. con estos valores obtenemos la ecuacion de regresion Y = 3.1390 + 0.20477 x

con esto solo reemplazamos cualquier valor de x para obtener los valores de Y (Y mediana de ingresos).

Si aplicáramos valores reales en el programa matlab, este resuelve sistemas de ecuaciones como por ejemplo :10 a + 460 b = 7600 , 460 a + 23634 = 36854 en donde resolviendo nos queda a =40.78 y b=0.76556 , sustituimos estos valores en la ecuación de regresión lineal y nos queda y=40.78 + 0.76556x con esta ecuación podemos predecir los valores de y, para x=50 nos dara y=79.28, el matlab utiliza funciones estadísticas internas para calcular los coeficientes de los pesos, también utiliza funciones para determinar la respuesta al impulso finito FIR, graficación de la función de transferencia Gs, y muchas cosas más, que serán explicadas en una lista de las funciones a utilizar en el programa de matlab.

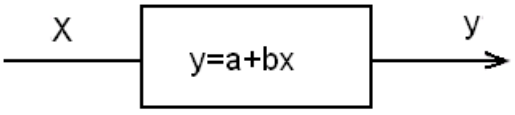
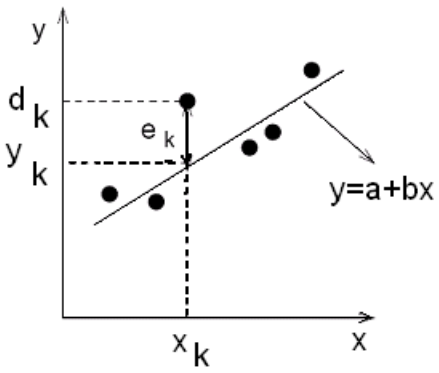


Figura 3.3 Ejemplo de regresion lineal



Medidas: $\{d_0, d_1, \ldots, d_{L-1}\}$
$\{x_0, x_1, \ldots, x_{L-1}\}$ conocidos
$y_k = y(x_k) = a + bx_k$
$e_k = d_k - y_k$

Figura 3.4 regresion lineal

objetivo: hallar parámetros a y b que minimicen el error cuadrático

A continuación mostramos los pasos que se requieren para calcular el valor del gradiente

Para un instante n, con pesos disponibles de w_n

1.- Generar un vector aleatorio con distribución uniforme en sus componentes $\underline{\delta}_n$

2.- Actualizar los pesos como $W_{n+1} \Rightarrow W_n + \frac{\mu_n}{2} \underline{\delta}_n$

3.- Calcular el nuevo error usando M vectores de datos

Si $\xi_{n+1} > \xi_n$ wn $\Rightarrow$ wn $\mu_o \Rightarrow \mu_{basico}$ n $\rightarrow$ n+1 pasar al paso 1

en caso contrario i = 2

4.-

Wn $\Rightarrow$ Wn+1 $\mu_o \Rightarrow \beta\mu_o$ con $\beta \geq 1$

Wn+1 $\Rightarrow$ Wn+1 - $\frac{\mu_o}{2}\delta_n$

si $\xi_{n+1} > \xi_n$ $\mu_o \Rightarrow \mu_{basico}$ n $\rightarrow$ n+1 pasar al paso 1 generando un vector de pertubacion ortogonal anterior

si $\xi_{n+1} < \xi_n$ i $\rightarrow$ i+1 volver

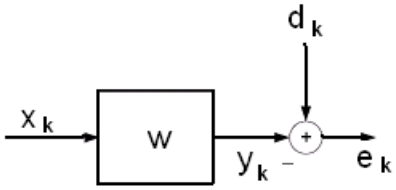
Formulación General LS

La forma más elemental del algoritmo LMS es su forma LS, su forma LS se comporta de manera completamente lineal, siguiendo el mismo orden del LMS, se suma las entradas S(n) con el ruido n(n) y se obtiene la entrada a ser evaluada en el filtro hasta obtener el valor de y(t) y , a continuación expondremos el análisis de su formulación**.**

Formulación

1.- Conjunto de observaciones: $d^t = (do, d1, \ldots, d_{L-1})$

2.- Modelo de producción de observaciones



Modelo: $\mathbf{w} = (w_0, w_1, \ldots, w_{N-1})$

Matriz de observación: $X\ (L \times N)$

Figura 3.5 determinacion de dk

3.- Criterio de optimización

$$\text{Minimizar } J = J(w) = \sum_{k=0} e_k^2 = e^t e = (d - X w)^t (d - Xw) \qquad (3.10)$$

$$\text{Solucion } w^* = (X^t X)^{-1} X^t d$$

Principio de ortogonalidad :

$$X^t e = 0 \Rightarrow y^t e = 0 \qquad (3.11)$$

Error cuadrático mínimo:

$$J^* = d^t \left[I - X (X^t X)^{-1} X^t \right] d \geq 0 \qquad \textbf{(3.12)}$$

Generalización a medidas con distinta confianza:

$$J = \sum_k \frac{e_k^2}{\sigma_{E,k}^2} \qquad \text{con } \sigma_{E,k}^2 = E[e_k^2] \qquad \textbf{(3.13)}$$

$$J = (d - X w)^t R_E^{-1} (d - Xw) = e^t R_E^{-1} e \qquad \textbf{(3.14)}$$

$$R_E = \begin{pmatrix} \sigma_{E,0}^2 & 0 & 0 & \cdots & 0 \\ 0 & \sigma_{E,1}^2 & 0 & \cdots & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & \cdots & & & \sigma_{E,L-1}^2 \end{pmatrix}$$

$$w = \left[(x^t R_E^{-1} X)^{-1} X^t R_E^{-1} \right] d \qquad \textbf{(3.15)}$$

$$J^* = d^t \left[R_E^{-1} - R_E X \left(X^t R_E X \right)^{-1} X^t R_E \right] d \qquad (3.16)$$

Principio de Ortogonalidad

$$X^t R_E^{-1} e = 0 \qquad (3.17)$$

El algoritmo LMS utiliza métodos de gradiente, los cuales son prácticos por su poca complejidad matemática en relación con otros métodos y algoritmos matemáticos. Por eso se suele hacer variaciones de los algoritmos LMS con otros algoritmos como el RLS o el DMI que posean otras características, para lograr hacer algoritmos híbridos más acorde con los requerimientos del usuario. Es de notar que se suele utilizar el gradiente de los algoritmos LMS para conformar el filtro de Wiener del mismo modo que se utilizar el algoritmo RLS para conformar el filtro de kalman. También el filtro de Wiener es utilizado ampliamente en los algoritmos DMI (algoritmo de inversión de matriz directa).

3.1.3 Variaciones con el algoritmo LMS

Es de señalar que en todos estos algoritmos se ve la influencia de filtros y en algunos de ellos de la función signo.

El algoritmo signo lms (sign - lms)

El algoritmo signo o señal es uno de los más populares algoritmos LMS modificados, ya que nos permite ciertas cualidades que son muy útiles para casos específicos, valores puntuales y ciertos rangos de valores. Este algoritmo junto con otros es denominado fino y elegante, el algoritmo signo reemplaza al algoritmo LMS y es evaluado en un solo punto, es decir es un algoritmo básico en donde mejora la conducta de la convergencia, reduce requerimientos computacionales (la cual es una cualidad del algoritmo LMS), y decrece el error mínimo cuadrático. Primero hablaremos del algoritmo signo

El error en el algoritmo signo o señal es :

$$w(n+1) = w(n) + 2\mu\ \text{sign}(e(n))x(n) \qquad (3.18)$$

donde

$$\text{sign}(n) = \begin{cases} 1 & n>0 \\ 0 & n=0 \\ -1 & n<0 \end{cases} \qquad (3.19)$$

El algoritmo LMS signo normalizado

Es el algoritmo signo LMS al que se le saca la norma de los vectores, esta normalización le permite tener cualidades que no presenta el LMS signo por si solo, le permite agregar el valor del error a la ecuación normalizada, solo hay que tener cuidado de que no se presente la división para cero al sumar el valor normalizado con el valor del error.

En algunos casos se suele utilizar valores de error muy pequeños y constantes para evitar la división para cero. Esto lo vemos en la ecuación 3.20

$$w(n+1) = w(n) + 2\mu\ \frac{\text{sign}(e(n))x(n)}{\varepsilon + \|x(n)\|^2} \qquad (3.20)$$

El algoritmo signo regresor lms

El signo regresor o algoritmo dato signo se caracteriza por dar datos x(n) a la función signo sign (x(n)) , la cual ha su vez es parte de la ecuación del algoritmo signo regresor, de hay su nombre, porque permite regresar los datos a ser evaluados. ecuación 3.21

$$w(n+1) = w(n) + 2\mu\, e(n)\, \text{sign}(x(n)) \qquad \textbf{(3.21)}$$

El algoritmo signo signo lms

En este algoritmo nosotros aplicamos la función signo para x(n). se puede aplicar una función signo dentro de otra función signo, esta es conocida como la función signo signo lms . ecuación 3.22

$$w(n+1) = w(n) + 2\mu\, \text{sign}(e(n))\text{sign}(x(n)) \qquad \textbf{(3.22)}$$

El algoritmo LMS normalizado (NLMS).

Es la normalización de el algoritmo LMS, en el cual se utiliza la recursión, para esto necesitaremos un algoritmo LMS donde se utilice parámetros variables de tiempo esto lo tenemos en la ecuación 3.23 en esta se presenta características de estabilidad y de convergencia.

$$w(n+1) = w(n) + 2\mu(n)e(n)x(n) \quad \textbf{(3.23)}$$

normalizamos el algoritmo LMS en u(n), esto lo vemos en la ecuación 3.24

$$\mu(n) = \frac{1}{2x^{T}(n)x(n)} = \frac{1}{2\|x(n)\|^{2}} \quad \textbf{(3.24)}$$

También podemos hacer que esta ecuación se comporte de manera recursiva aplicando la ecuación 3.24 en 3.23 y obtenemos la ecuación 3.25

$$w(n+1) = w(n) + \frac{1}{x^{T}(n)x(n)}\ e(n)x(n) \qquad (3.25)$$

Con esta ecuación podemos evaluar el valor del error cuadrático. Todos estos son procedimientos de optimización, al agregar el valor de error a la ecuación obtenemos la ecuación 3.26

$$w(n+1) = w(n) + \frac{\bar{\mu}}{\varepsilon + x^{T}(n)x(n)}\ e(n)x(n) \qquad (3.26)$$

Donde obtenemos los valores de $\bar{\mu}$ y $\mathcal{E}$. El valor de $\mathcal{E}$ es pequeño y constante previene la división para valores de datos normalizados muy pequeños.

En la tabla IV encontraremos la ecuación del algoritmo LMS normalizado (NLMS).

Tabla IV el algoritmo NLMS

funciones de valores reales		funciones complementarias
entrada:		
inicializacion del vector	**W(n)=0**	
vector de entrada	**x(n)**	
señal referencia	**d(n)**	
(es colocada a la salida)		
parametro de paso	$\bar{\mu}$	
Constante	ε	
Largo del filtro	**M**	
salida:		
señal filtrada a la salida	**y(n)**	
vector de coeficientes	**W(n+1)**	
Procedimiento:		
1) $y(n) = w^T(n)x = w(n)x^T(n)$		1) $y(n) = w^H(n)x(n)$
2) $e(n) = d(n) - y(n)$		2) $e(n) = d(n) - w^H(n)x(n)$
3) $w(n+1) = w(n) + \frac{\bar{\mu}}{xT(n)x(n)} e(n)x(n)$		3) $w(n+1) = w(n) + \frac{\bar{\mu}}{\varepsilon + \mathbf{x}^H(n)\mathbf{x}(n)} \mathbf{x}(n)e^*(n)$

La H suele representar la transpuesta conjugada en algunas nomenclaturas

El algoritmo Variable LMS step size (VSLMS).

El algoritmo VSLMS fue introducido en 1986 para facilitar requerimientos que no se pueden obtener en otros algoritmos, para esto hacemos valores paramétricos largos y es necesaria la rápida convergencia y valores para métricos muy pequeños, con estos obtenemos errores y valores de desajuste más pequeños, para esto

empezamos adaptando w(n) para valores óptimos, con estos valores formamos un filtro de coeficientes w(n) que son aproximaciones de la solución. Estos valores paramétricos decrecen en el orden del exceso de MSE. La variación de los parámetros de los coeficientes se expresa como un algoritmo de recursión LMS (ecuación 3.27)

$$wi(n+1) = wi(n) + 2\mu_i(n)e(n)x(n-1) \quad i=0,1,.....,M-1 \quad (3.27)$$

Donde wi(n) es el coeficiente i esimo de w(n), en la iteración n y $\mu_i(n)$ es la asociación para determinar los valores determinados. Basado en un monitoriado de la función signo, que nos da un valor estimado del gradiente $e(n)x(n-i)$ Este valor estimado cambia sucesivamente. Este valor de decrecimiento tiene un factor c1, que se basa en un numero m1, con valores sucesivos. $e(n)x(n-i)$

Tabla V El algoritmo VSLMS

Entrada: vector de coeficiente inicial: w(0)

vector datos de entrada: $x(n) = [x(n)\ x(n-1)....x(n-M+1)]^T$

terminos del gradiente: $go(n-1)=e(n-1)x(n-1), g1(n-1)=e(n-1)x(n-1),.....,$ $g_{M-1}(n-1)=e(n-1)x(n-M)$

parametros de paso: : $\mu_0(n-1), \mu_1(n-1),....., \mu_{M-1}(n-1)$

a = constante positiva pequeña

μ_{max} = constante positiva del algoritmo

Salidas: señal de referencia

tambien llamada señal deseada: d(n)

Señal obtenida a la salida del filtro: y(n)

filtro actualizado de pesos: w(n+1)

termimos del gradiente: go(n),g1(n),.....,gM-1(n)

parametros actualizados de paso : $\mu_0(n), \mu_1(n),....., \mu_{M-1}(n)$

procedimiento: 1) $y(n) = w^T(n)x(n)$

2) e(n) = d(n) - y(n)

3) pesos y parametros de paso de adaptacion

For i= 0,1,2.....,M-1

gi(n)=e(n)x(n-1)

$\mu_i(n) = \mu_i$ (n-1)+ assign(gi(n))sign(gi(n))

if μ_i (n)> μ_{max}, $\mu_i(n) = \mu_{max}$

if μ_i (n)< μ_{min}, $\mu_i = \mu_{min}$

wi(n+1)=wi(n)+2μ_i(n)gi(n)

end

Incrementan los valores de los parámetros en un factor c2. Que se basa en un numero m2, y asi sucesivamente hasta llegar a un valor optimo. Tabla IV

Con el valor de la ecuación 3.27 podemos escribir esta ecuación en forma de matrices de la forma w(n+1) ecuación 3.28

$$w(n+1) = w(n) + 2\mu(n)e(n)x(n) \quad (3.28)$$

Donde $u(n)$ es la matriz diagonal con elementos u0. El algoritmo VSLMS se encuentra en la tabla IV

Existen muchas más variantes con el algoritmo LMS, no podemos hablar de todas porque haría el presente proyecto demasiado extenso, para lo cual mostraremos una tabla con los principales algoritmos que son una variante del algoritmo LMS. Tabla VI

Tabla VI Variantes del algoritmo LMS

$x(n)=[x(n)x(n-1)....x(n-M)]^T$, $w(n)=[wo(n)w1(n).....wM(n)]^T$, $e(n)=d(n)-y(n)$

Algoritmo	Recursivo
1. LMS	$w(n+1)=w(n)+2\mu e(n)x(n)$
2. LMS con datos complejos	$w(n+1)=w(n)+2\mu e^*(n)x(n)$ $y(n)=w^H(n)x(n)$ (H= transpuesta conjugada)
3. Signo LMS	$w(n+1)=w(n)+2\mu sign(e(n))x(n)$
4. Signo regresor LMS	$w(n+1)=w(n)+2\mu e(n)sign(x(n))$
5. signo signo LMS	$w(n+1)=w(n)+2\mu sign(e(n))sign(x(n))$
6. LM normalizado	$w(n+1)=w(n)+\frac{1}{x^T(n)x(n)}e(n)x(n)$ with $\mu(n) = 1/[2x^T(n)x(n)]$
9 a ε- Normalizado LMS	$w(n+1)=w(n)+\frac{\bar{\mu}}{\varepsilon + x^T(n)x(n)}e(n)x(n)$ $\bar{\mu}$ = parametros de paso ε = prevenir división para numeros muy pequeños
9b ε - Normalizado LMS con datos complejos	$w(n+1)=w(n)+\frac{\mu}{\varepsilon + x^H(n)x(n)}e^*(n)x(n)$ H = transpuesta conjugada
10. algoritmo signo LMS Normalizado	$w(n+1)=w(n)+2\mu\frac{sign(e(n))x(n)}{\varepsilon+\|x(n)\|^2}$
11. Leaky LMS	$w(n+1)=(1-2\mu\gamma)w(n)+2\mu e(n)x(n)$ 0<<<1
12. Compacto LMS	$w(n+1)=w'(n)+\frac{a-c^T w'(n)\ c}{c^T c}$ $w'(n)=w(n)+2\mu e(n)x(n)$ c= vector constante a = constante

Algunos autores consideran el encontrar los valores de error y los métodos de gradiente como una variante más del algoritmo LMS, sin embargo otros autores solo lo consideran como una aplicación del

algoritmo LMS, en la cual se implementa el filtro de wiener. Por su importancia dentro del algoritmo LMS lo colocaremos en la presente tesis como si fuera una variante más del algoritmo LMS.

El algoritmo RLS

Es una variante del algoritmo LMS, el RLS en si sugiere múltiples variantes como controlar la distribución del vector de perturbación, usar diferentes distribuciones para cada uno de los pesos del filtro, tomar el modulo en lugar del cuadrado del error para ahorrar operaciones, etc. Todo lo contenido en este capítulo se desarrollo en los años 70 en el contexto de procesado de señal para reconocimiento de formas y ha servido de inspiración a lo que hoy se denomina de manera más precisa algoritmos genéticos o aprendizaje en redes neuronales. Se recomienda que antes de utilizar algoritmos genéticos o neuronales, se comprenda y se familiarice con los algoritmos de gradiente que todavía hoy son materia de investigación y publicaciones prestigiosas. De otro modo, se podría caer en el defecto de tratar de solucionar problemas con herramientas mucho más complicadas e incomprensibles que un LMS, DSD o RLS, que en el 90% de las situaciones son más que suficiente para una aplicación

concreta. Sobre este algoritmo hay un capitulo en la presente tesis, con lo cual se puede ver la importancia de él.

El algoritmo GRS (Guided random search)

Básicamente el algoritmo, denominado GRS (Guided Random Search), consiste en aprovecharse del RLS, aprovecha el hecho de que el error disminuya (y esto implica que la dirección escogida en la antena es correcta) se caracteriza por mantener e incluso en aumentar el desplazamiento en la dirección encontrada. La idea es permanecer en la dirección correcta hasta que se es tangente a una curva de error. Llegado ese momento o cercano a el, la dirección correcta será ortogonal (dependiendo de lo próximo que se esté al punto de tangencia con una superficie de error constante) comenzándose de nuevo el algoritmo. De hecho el GRS usa, en parte, una propiedad del algoritmo AG (Gradiente Acelerado). (el algoritmo AG tiene la característica de dar rápidos valores de gradiente. Esta propiedad establece que para criterios de error cuadrático, la recta que une el punto de comienzo con el segundo punto de tangencial es la dirección que apunta directamente al mínimo. Esta propiedad del AG se esquematiza en las figura 3.6 y 3.7. El GRS usa de hecho esta propiedad ya que cuando se agota la dirección correcta, encontrada por

búsqueda aleatoria, procede a realizar una búsqueda en la dirección ortogonal, aleatoria aun, a la usada.

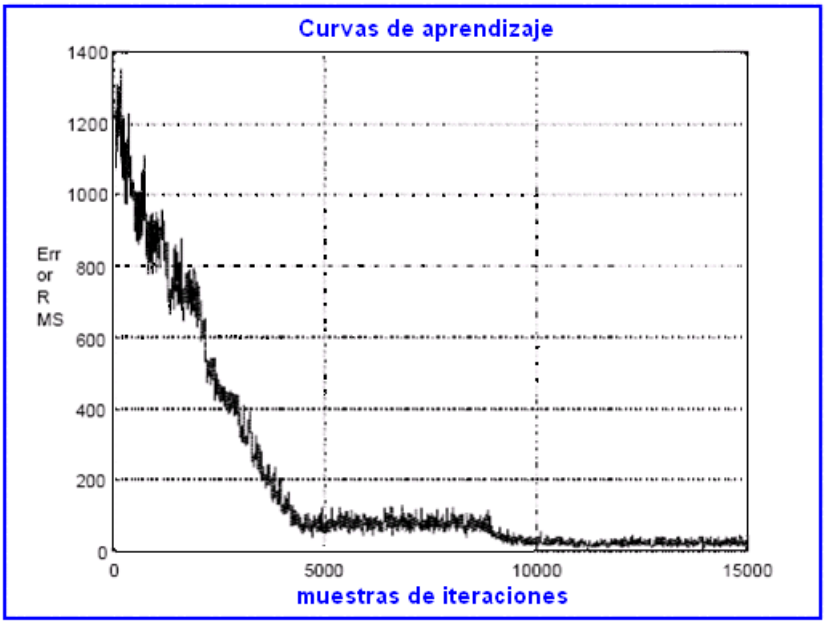


Figura 3.6 Curva de aprendizaje para el algoritmo lms

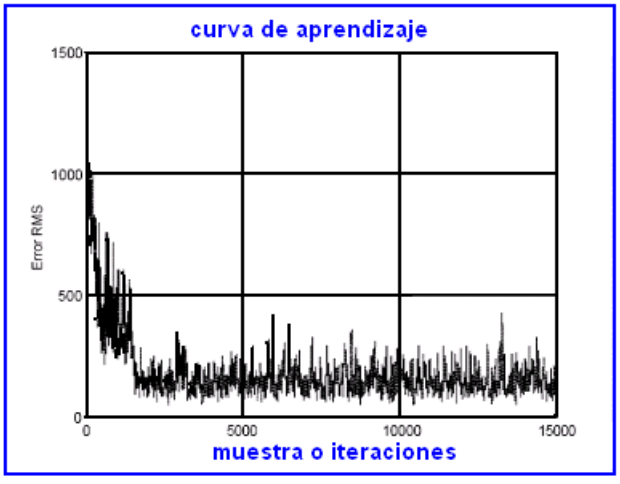


Figura 3.7 Curva de aprendizaje para el algoritmo LRS. La distribución de la perturbación es uniforme de varianza 0.3 con un paso de adaptación igual a 0.1 (en la parte superior) y 0.5 (en la inferior).

3.2 FILTRO DE WIENER

El uso de los filtros es de especial importancia ya que con estos se permite el paso de la señal deseada, eliminando ruido y depurando las señales. Es de tener presente que según el algoritmo adaptativo que se use , también será el filtro que se suele utilizar, por ejemplo tenemos que él algoritmo LMS suele utilizar el filtro de wiener, en este filtro se utiliza el algoritmo mínimo cuadrado, mientras que con el algoritmo adaptativo RLS mínimo cuadrado recursivo se suele utilizar el filtro de kalman. En la década de 1940 a 1950 Wiener estuvo investigando la teoría del caos, en esa época todos los sistemas eran hechos en función del tiempo, y se requería una ecuación específica para cada caso distinto, se podía crear otras ecuaciones en función de la frecuencia, el ángulo, etc. Wiener propuso que se podría crear una sola función que con pequeños cambios en ella podría utilizarse en función del tiempo, del ángulo, de la frecuencia, etc, para esto se requeriría funciones que tuvieran ciertas características en función de la primera, segunda y mas derivadas, también deberían tener ciertos limite, el propuso en esa época que podrían crearse estas funciones a las que denomino invariantes en el tiempo, en la década de 1940 y 1950 escogió para probar su teoría a las funciones gaussianas que cumplían las condiciones que el necesitaba. Eligio a las funciones gaussianas por que en esa época la capacidad computacional no era muy grande. También pronostico que en el futuro se podrían utilizar funciones senosoidales como señal de

entrada y referencia y que esto ocurriría cuando las computadoras tuvieran una mayor capacidad de cálculo.

También hay un amplio uso del filtro de wiener con los algoritmos dmi (Inversión de Matriz Directa) en estos se los usa para tener ciertas características, como el de calcular los valores de error cuadrático, el valor del mínimo error cuadrático medio y la determinación de muestras directa de covarianza de algoritmo inverso, cuando el filtro de wiener es utilizado con el algoritmo dmi tiene la característica de que ya no presenta la cualidad de ser lineal, esto da ciertas ventajas y desventajas, pero eso se analizara en otro capítulo de esta tesis.

Se le suele llamar al filtro de wiener el filtro de filtrado optimo, el cual utiliza el criterio de mínimo error cuadrático, el principio de ortogonalidad, el filtro de wiener es utilizado en igualación, habiendo extensiones de su uso como el filtro de wiener complejo, hay un desarrollo del filtro de wiener IIR que es llamado causal y no causal. El filtro de wiener por su utilización de los algoritmos LMS se lo suele llamar de filtrado optimo o filtrado lineal optimo con restricciones. Para su utilización se requiere una solución iterariva de las ecuaciones normales y se suele hacer un análisis de su convergencia.

El filtro de coeficientes de wiener esta denotado por w, para valores reales estacionarios x(n), que produce valores estimados de d(n). estos valores producirán la determinación de valores de error e(n).

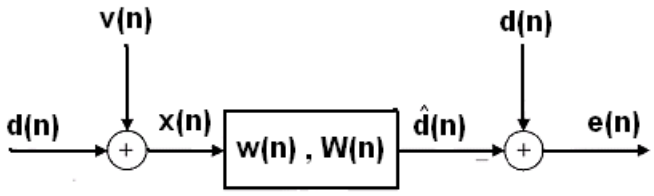


Figura 3.8 Diagrama de bloques de el filtrado optimo

El filtro de wiener es el estimador lineal óptimo en el sentido del mínimo error cuadrático medio MMSE, usado en entornos estacionario.

Estima el canal h conociendo los datos transmitidos (pesos iniciales o valores iniciales), los valores conocidos son rk ecuación 3.29 y xk ecuación 3.30

$$\mathbf{r}_k = [r_k, r_{k-1}, \dots, r_0], \tag{3.29}$$

$$\mathbf{x}_k = [x_k, x_{k-1}, \dots, x_0] \tag{3.30}$$

En la figura 3.9 vemos el diagrama de bloques del filtro de wiener de la función rk

fig 3.9 diagrama del filtro de wiener

El filtro de wiener nos dará el error ek en función de rk y yk, ecuación 3.31 y del error cuadrático medio ecuación 3.32.

$$\text{el error sera: } e_k = r_k - y_k = r_k - h^T x(k) = r_k - \sum_{i=0}^{N-1} h(i)x(k-i) \qquad (3.31)$$

$$\text{el error cuadratico sera: } e_k^2 = r_k^2 - 2r_k x(k)^T h + h^T x(k)x(k)^T h \qquad (3.32)$$

El error cuadrático medio MMSE se obtiene tomando esperanzas matemáticas y considerando a x y r como estacionarios, de donde obtenemos la función de costo ecuación 3.33

$$J = E\{ e_k^2 \} \tag{3.33}$$

$$J = E\{r_k^2\} - 2E\{r_k x(k)^T h\} + E\{h^T x(K) x(k)^T h\}$$

$$J = \sigma^2 - 2P^T h + h^T R h$$

En donde P es el vector de correlación cruzada y R la matriz de autocorrelación de la señal deseada. Ecuación 3.34.y 3.35.

$$P = E\{r_k \, x(k)\} \tag{3.34}$$

$$R = E\{x(k) \, x(k)^T\} \tag{3.35}$$

Calculando el gradiente en función de h, realizamos una derivada dJ/dh. Ecuación 3.36

$$\frac{dJ}{dh} = -2P + 2Rh \tag{3.36}$$

Con este valor igualando a cero se obtiene el mínimo MSE, este es el h optimo o ecuación de wiener-Hopf ecuación 3.37.

$$h_{opt} = R^{-1} P \qquad \text{Ecuacion de Wiener Hopf} \qquad (3.37)$$

3.3 La Solucion de Wiener

El filtro de Wiener utiliza la regresión lineal para encontrar los coeficientes de la ecuación de pesos, conocidos estos coeficientes solo hay que resolver un sistema de ecuaciones que nos da los pesos para cada iteración.

Formaremos 2 ecuaciones diferenciales en donde se aplica la primera derivada, con respecto a los valores de los pesos (wo, w1) ecuación 3.29 resolviendo estas ecuaciones del filtro de wiener obtendremos las ecuaciones para obtener los pesos ecuación 3.30

Nosotros observamos que existen planos que presentan forma parabólica y tienen un punto mínimo y partes que son paralelas al plano w, nosotros observamos que aplicando la primera derivada de los valores de error con respecto a wo y w1 e igualándolos para cero encontramos los puntos mínimos, y aplicando la segunda derivada obtenemos valores positivos.

$$\frac{\partial J(wo,w_1)}{\partial wo} = 0 \qquad \frac{\partial J(wo,w_1)}{\partial w_1} = 0 \qquad (a)$$

$$\frac{\partial^2(wo,w_1)}{\partial^2 wo} > 0 \qquad \frac{\partial^2 J(wo,w_1)}{\partial^2(w_1)} > 0 \qquad (b) \qquad (3.38)$$

Para los 2 coeficientes en el filtro obtenemos la ecuación 3.39

$$J(wo,w_1)=w_0^2 r_x(0)+2wow_1 r_x(1)+w_1^2 r_x(0)-2wo\, r_{dx}(0)-2w_1 r_{dx}(1)+\sigma_d^2 \qquad (3.39)$$

Introduciendo la ecuación 3.38 en la parte a de 3.39 producimos un sistema de ecuaciones. ecuaciones 3.40

$$2w_o^0 r_x(0) + 2w_1^0 r_x(1) - 2r_{dx}(0) = 0 \qquad (a)$$

$$2w_1^0 r_x(0) + 2w_0^0 r_x(1) - 2r_{dx}(1) = 0 \qquad (b) \qquad (3.40)$$

Resolviendo el sistema de ecuaciones encontramos que pueden formar una matrix, que puede ser evaluada en la ecuación de wiener ecuación B.1

Esto lo explicaremos más a fondo en el anexo B y en la parte 3.4 del presente capitulo, donde se ve como aplicando matlab se entiende mejor todas estas ecuaciones.

3.4 Descripción de las diferentes partes del programa en matlab

Toda la parte teórica anteriormente explicada, se verá lo fácil que se vuelve cuando se utiliza el programa matlab, el cual utilizando una serie de funciones que resuelven el algoritmo rápidamente. A continuación explicaremos los diferentes partes que tiene el programa en matlab y como se llega a estos valores.

En el programa primero ingresaremos el orden del filtro, luego el numero de antenas, se ingresa los valores de las antenas, con estos valores se los almacena en una variable h, estos son los valores iniciales, se nos preguntara si deseamos ingresar un ruido adicional en el sistema, si lo deseamos agregar a la entrada, sumándose este ruido adicional a la entrada o a la salida sumándose este ruido a la señal de referencia, utilizaremos una función de entrada, esta función de entrada puede ser del tipo senosoidal o del tipo gaussiano, el lazo que almacena estos valores puede ser de diferentes tamaños, según el numero de muestras que se deseen.

El lazo de valores de la señal de entrada es:

```
for k=1:400
entrada(k)=sin((2*pi*k)/M);
% senal_referencia(k)=cos((2*pi*k)/M); % señal de referencia en algunos
libros se la llama señal deseada
End
```

Tambien existirá una señal de referencia en donde se evaluara los valores de la Señal para compararla con los valores de los pesos que obtengamos en el programa.

El lazo de valores de la señal de referencia es:

```
for k=1:400
%entrada(k)=sin((2*pi*k)/M);
senal_referencia(k)=cos((2*pi*k)/M); % señal de referencia en algunos libros
End   % se la llama señal deseada
```

en el programa, ingresamos una opción señal para indicar si se desea utilizar la misma ecuación de entrada como señal de referencia o utilizar otra curva como señal de referencia.

Utilizamos una función de matlab llamada butter con el cual resolveremos los valores de a y b que son valores de una regresión lineal u otra, se ingresa una función de transferencia Gz=tf(b,a,-1) tf es otra función de matlab 7, muchas de la funciones que se usaran en estos algoritmos adaptativos son de matlab7 y se lo necesita para hacer correr el programa.

En algunas versiones del algoritmo lms se determina los valores de h con la función h=ldiv(b,a,orden) que en el presente programa no utilizaremos, a la salida de la función de transferencia la llamaremos y.

'y' se lo evaluara con la función lsim con los valores de entrada serán Gz y entrada, las funciones en matlab se escribirán:

```
[b,a] = butter(2,0.25) % usaremos otra forma para calcular a y b
Gz = tf(b,a,-1) % la funcion de transferencia tf funciona en matlab 7
% esta funcion permite resolver la transformada z,
% el primer orden evalua valores de pesos con la funcion ldiv
%h=ldiv(b,a,orden)';
% if you use ldiv this will give h :filter weights to be
% ingreso de los voltajes en la señal de entrada (estos voltajes serán
% los valores de pesos iniciales que serviran para hallar los pesos W
```

```
y = lsim(Gz,entrada) % simula respuestas arbitrarias de entrada
```

La potencia en el programa lo determinaremos con la siguiente línea de código:

```
potencia=mean(senal_referencia.^2)
if x==1
% tambien calcularemos la potencia para cada una de las antenas
% antena1
entrada1=D(1)+n+E(1);
potencia1=mean(entrada1.^2)
end
```

El cálculo de la cota nos servirá para obtener un valor de mu (constante de ajuste en el algoritmo lms) y su fórmula es:

```
% calculo de la cota de la contante de ajuste
cota=1/((orden+1)*potencia);
```

El lazo de actualización de los pesos es el siguiente

```
%para 60 puntos  por corrida determinamos los pesos w
N=60
%begin of algorithm
w = zeros ( orden  , 1 )
for n = orden : N
```

```
 u = entrada(n:-1:n-orden+1);
y(n)= w' * u;  % y(n) señal a la salida
    % e(n) = senal_referencia(n) - y(n);
    e(n)=senal_referencia(n)-y(n);
    j=j+1 % nos da el numero de iteraciones hasta que se cumpla el error
     if e(n)<error
        fprintf('el numero de iteraciones hasta que se completo el error ');
         j
  end

% se puede lograr mayor exactitud en la curva variando el valor de mu
% si se coloca valores de mu muy altos se pierde la señal de ajuste
% con valores de mu más pequeños se hace más lenta la convergencia
    if n < 20
        mu=cota/2
    else
        mu=cota/4
    end
    w = w + mu * u * e(n);
    % algunos autores usan el valor de w = w + 2*mu * u * e(n); para
    % evaluar los pesos, y hay personas que están tratando de mejorar
%esta ecuacion
end
```

El lazo de chequeo de los resultados del error es:

```
% chequeo de resultados de error
for n =  N+1 : totallength
   u = entrada(n:-1:n-orden+1) ;
   y(n) = w' * u ;   % y(n) señal a la salida
   % e(n) = senal_referencia(n) - y(n) ;
   e(n)=senal_referencia(n)-y(n);
end
```

Esta estimación tiene ruido y puede considerarse como el gradiente real contaminado por un ruido aditivo. Los algoritmos LMS tienen la desventaja que internamente producen un ruido aproximado al 10%, sin contar otros ruidos externos.

El algoritmo LMS emplea una estimación especial del gradiente que resulta válida para el filtro FIR adaptativo y se puede extender a filtros IIR adaptativos. Como se verá a continuación, es un algoritmo muy importante debido a su simplicidad y facilidad de cálculo.

En cada iteración, el error se calcula como:

$$\varepsilon_k = d_k - X_k^T W_k = d_k - W_k^T X_k \qquad (3.41)$$

Y la regla de actualización de los coeficientes viene dada por 3.42:

$$W_{k+1} = W_k - \alpha \nabla \xi(Wk) \qquad (3.42)$$

Por tanto, para obtener una estimación del gradiente es necesario estimar el error cuadrático medio. Este algoritmo utiliza un método instantáneo, de modo que la estimación del error cuadrático medio en cada iteración es exactamente el cuadrado del valor del error medido:

$$\hat{\xi}_k = \hat{E}_k\left[\varepsilon^2\right] = \varepsilon_k^2 \qquad (3.43)$$

El gradiente estimado tendrá la forma:

$$\hat{\nabla}_{\xi_k}(W_k) = \left[\frac{\partial \hat{\xi}_k(W_k)}{\partial W_0}, \frac{\partial \hat{\xi}_k(W_k)}{\partial W_1}, \ldots, \frac{\partial \hat{\xi}_k(W_k)}{\partial W_L}\right]^T$$

$$= \left[\frac{\partial \varepsilon_k^2(W_k)}{\partial W_0}, \frac{\partial \varepsilon_k^2(W_k)}{\partial W_1}, \ldots, \frac{\partial \varepsilon_k^2(W_k)}{\partial W_L}\right]^T = 2\varepsilon_k\left[\frac{\partial \varepsilon_k(W_k)}{\partial W_0}, \frac{\partial \varepsilon_k(W_k)}{\partial W_1}, \ldots, \frac{\partial \varepsilon_k(W_k)}{\partial W_L}\right]^T \qquad (3.44)$$

$$\frac{\partial \varepsilon_k(W_k)}{\partial W_i} = \frac{\partial\left[d_k - \sum_{m=0}^{L} X_{mk} W_{mk}\right]}{\partial W_i} = -X_{ik} \quad i=0,1,\ldots,L \Rightarrow \hat{\nabla}_{\xi_k}(W_k) = -2\varepsilon_k X_k \qquad (3.45)$$

Y el algoritmo LMS queda:

$$\mathbf{W}_{k+1} = \mathbf{W}_k + 2\alpha\varepsilon_k\mathbf{X}_k \qquad (3.46)$$

El programa en matlab del algoritmo LMS. Deberá recibir los siguientes argumentos de entrada:

- la señal aplicada a la entrada del filtro, Xn=2*sin((2*pi*k)/N)

- el numero de antenas n

- la señal de referencia, d(k)=2*cos((2*pi*k)/N);

- el ruido de las diferentes antenas, (hay que tener presente que el algoritmo LMS en su estructura internamente produce un ruido del 10%), aparte de este ruido puede haber otros. n(f)

- el orden del filtro FIR cuyos coeficientes se van a ajustar, orden

- el número de iteraciones en el aprendizaje, N (este número se hará hasta que cumpla el error)

- la constante de ajuste mu

- el vector de pesos inicial. Los voltajes de las antenas V(f)

- el error deseado (puede ser ingresado por teclado) error

Si se conociera la frecuencia se podría usar como entrada Xn=2*cos(2*pi*f*t) y tener una ecuación en función del tiempo y la frecuencia.

A su salida devolverá:

- la salida en cada iteración durante el proceso de ajuste, y(t)

- la curva de aprendizaje (grafico de la curva)

- los vectores de pesos obtenidos durante el proceso. W(t)

- la Potencia de la antena P(k)

· el numero de iteraciones en que converge la curva y se cumple el valor del error. z

El programa determinara los coeficientes de la función de transferencia, los cuales los llamaremos a y b, siendo b los coeficientes del numerador y a los coeficientes del denominador, el programa matlab suele utilizar la función butter para hallar los coeficientes, esta función es del matlab 7 .ejemplo [b,a] = butter (2,0.25) según los valores que hay dentro de butter se pueden formar los valores de a y b, en caso que no se pueda usar la función butter se puede usar una función de transferencia en función de z.

.b = 0.09763 z^2 + 0.1953 z + 0.09763, .a = z^2 - 0.9428 z + 0.3333

Gz = b / a

Para expresar la función de transferencia en matlab usamos tf

```
>> w2 = 62.83^2
h = tf(w2,[1 2 w2])
w2 =
  3.9476e+003
```

Transfer function:

$$\frac{3948}{s^2 + 2s + 3948}$$

3.5 Pantallas de visualización del algoritmo adaptativo mínimo cuadrado lms

Al hacer correr cualquiera de las aplicaciones en matlab del algoritmo adaptativo mínimo cuadrado lms que hemos realizado, se presentaran algunas pantallas de visualización, algunas de estas pantallas son para información de los usuarios a usar el programa, otras son para ingresar valores de entrada, y las ultimas son las generadas por el programa matlab son pantallas de salida, con estos datos de entrada y salida formaremos tablas de datos y resultados obtenidos. Todas estas pantallas servirán tanto para señales senosoidales como para señales gausianas. También podrían servir para otros tipos de señales que cumplan las características señaladas por wiener para filtros adaptativos.

Las pantallas a utilizar en el programa del algoritmo adaptativo lms son:

Pantalla de presentación:

En esta pantalla se hace una explicación al usuario de las características de la aplicación y lo que resuelve, está el nombre de la universidad, los nombres de los programadores del programa, el profesor guía, el tipo de señal a usar en el programa, , una pequeña explicación de los datos de entrada y los de salida.

Pantalla de ingreso de datos:

En esta pantalla ingresamos el numero de antenas a utilizar (en el algoritmo lms de 1 a 5)., los valores iniciales de las antenas, los valores de ruido de las antenas.

En esta pantalla aparecerá una opción llamada opción ruido que si es igual a 1 el ruido se ingresara a la entrada del sistema (antes del filtro), si se escoge 2 el ruido se agregara a la salida del sistema. También se ingresara un valor de error del sistema

Pantalla de potencias obtenidas:

Esta pantalla nos dará la potencia del sistema y el valor de la potencia de cada una de las antenas.

Pantalla de comparación de pesos estimados:

En esta pantalla presentaremos los datos iniciales ingresados por teclado (los llamaremos también valores actuales) los representaremos por medio de una cruz negra, y los compararemos con los valores obtenidos por el algoritmo lms, este valor obtenido lo llamaremos valores estimados, y lo representaremos con un color rojo.

Pantalla de valor de la curva de error:

En esta pantalla se muestra el comportamiento de la ecuación de error del algoritmo lms, esta nos servirá para compararla con el comportamiento de otros algoritmos adaptativos.

Pantalla de comparación entre la salida del sistema con la señal de referencia: Esta pantalla es una de las más importantes, aparecerá la señal que obtenemos a la salida del sistema (color rojo) , y se la comparara con la señal de referencia (color azul).

En el anexo F se muestran las pantallas de visualización que obtendremos en nuestras implementaciones al utilizar diferentes valores de entrada y diferentes casos en cada uno de los algoritmos utilizados.

3.6 Programa en matlab del algoritmo adaptativo mínimo cuadrado LMS

Al hacer el programa se presento la idea de hacerlo de corrido, o utilizando funciones los cuales se llamarían desde un programa principal, decidimos colocar el algoritmo LMS dentro de una función en matlab y de esta manera poder llamarlo desde el editor, esto permite ciertas ventajas ya que todo el programa está en un solo archivo, para evitar que se haga muy complejo el programa decidimos colocar la mayor cantidad posible de comentarios, de lo que se hace en cada parte del programa, y algunas partes están por bloques, que ayudan a su comprensión y aplicación, permitiendo a futuro poder agregarle mas partes con un mínimo cambio en otras partes del programa.

Programa Principal y función principal

```
function[caratula]=problema3_lms;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Algoritmo LMS
% Autores: Juan Alvarez , Maribel Chuez
%
%Escuela Superior Politecnica del Litoral
%Facultad de Electricidad y Computacion
%Guayaquil, Ecuador
% email :juan_alvarez8@latinmail.com
% email : mchuez@ceibo.espol.edu.ec
% Webpage : proximamente
% Date    : Viernes 30 de abril del 2009
% Version : 1.0.3
% Referencias : S. Haykin, Adaptive Filter Theory. 3rd edition,
Upper Saddle River, NJ: Prentice-Hall, 1996.
%
%clear all % esta función borra los valores anteriores que estén
memoria %
 %close all % esta función borra cualquier ventana anterior activa %
 %hold off   % esta función encera cualquier grafico que estuviera
activo %

fprintf('*******************************************************.\n')
fprintf('*                                                     *.\n')
fprintf('*      ESCUELA SUPERIOR POLITECNICA DEL LITORAL       *.\n')
fprintf('*                                                     *.\n')
fprintf('* PROBLEMA3_lms.- Determinar los pesos, el numero de *.\n')
fprintf('* iteraciones y la potencia de un grupo de antenas   *.\n')
fprintf('* inteligentes utilizando el algoritmo LMS (algoritmo*.\n')
fprintf('* mínimo cuadrado, las antenas están distribuidas en *.\n')
fprintf('* forma de celdas para lo cual requeriremos los      *.\n')
fprintf('* siguientes datos:                                  *.\n')
fprintf('*                                                    *.\n')
fprintf('*  • la señal aplicada a la entrada del filtro,      *.\n')
fprintf('*         ej  Xn=2* sin((2*pi*k)/N)                  *.\n')
fprintf('*  • el numero de antenas  x                         *.\n')
fprintf('*  • la señal de referencia, d(k)=2*cos((2*pi*k)/N); *.\n')
fprintf('*  • el ruido de las diferentes antenas, (hay que    *.\n')
fprintf('*  tener presente que el algoritmo LMS en su         *.\n')
fprintf('*  extructura internamente produce un ruido del 10   *.\n')
fprintf('*  por ciento,aparte de este ruido puede haber otros.*.\n')
fprintf('*  ruidos adicionales que ingresaremos por teclado   *.\n')
fprintf('*  • el orden del filtro FIR cuyos coeficientes se   *.\n')
fprintf('*  van a ajustar, orden                              *.\n')
fprintf('*  • el número de iteraciones en el aprendizaje, N   *.\n')
fprintf('*  (este número se hará hasta que cumpla el error)   *.\n')
fprintf('*  • la constante de ajuste   mu                     *.\n')
fprintf('*  • el vector de pesos inicial. Los voltajes de las *.\n')
```

```
fprintf('*  antenas V(f)                                          *.\n')
fprintf('*  • el error deseado (puede ser ingresado error        *.\n')
fprintf('*   por teclado)                                        *.\n')
fprintf('*                                                       *.\n')
fprintf('*  A su salida devolverá:                               *.\n')
fprintf('*  • la salida en cada iteración durante el proceso     *.\n')
fprintf('*  de ajuste, y(t)                                      *.\n')
fprintf('*  • la curva de aprendizaje (grafico de la curva)      *.\n')
fprintf('*  • los vectores de pesos obtenidos durante el         *.\n')
fprintf('*  proceso. W(t)                                        *.\n')
fprintf('*  • la Potencia de la antena P(k)                      *.\n')
fprintf('*  • el numero de iteraciones en que converge la        *.\n')
fprintf('*  curva y se cumple el valor                           *.\n')
fprintf('*       del error.  j                                   *.\n')
fprintf('*    alumno: Juan Francisco Alvarez Alvarado            *.\n')
fprintf('*    alumno: Maribel del Rosario Chuez Gonzales         *.\n')
fprintf('*                                                       *.\n')
fprintf('*    Profesor :Ing. Pedro Vargas                        *.\n')
fprintf('*                                                       *.\n')
fprintf('*******************************************************.\n')
t=input('teclee enter para proseguir');
 clc
% ingresamos el orden del filtro, cuyo coeficiente se va ha ajustar%
orden=input(' ingrese el orden del filtro, cuyo coeficiente se va ha
ajustar....')
% los valores de pesos que recomendamos para en estas dos curvas
son:
% h=0.5; 0.2; -0.2; -0.4; -0.7
% valores recomendado por Juan Alvarez y Maribel Chuez
% se tiene que ingresar valores que estén dentro del rango de la
curva
% se puede ingresar valores cercanos a estos, pero el error
aumentara
% mientras más alejados sean de estos valores.
% ingresamos el número de antenas del sistema
disp(' ingrese el numero de antenas entre 1 y 5');
x=input(' ingrese el numero de antenas entre 1,2,3,4 o 5....');
if x==1
    D(1)=input('ingrese el voltaje1 (en V)..... voltaje1=');
end

if x==2
    D(1)=input('ingrese el voltaje1 (en V)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
end

if x==3
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
end

if x==4
```

```
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
end
if x==5
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
end
% ingresamos el ruido que presenta cada antena del sistema
% Si existe ruido adicional al ocasionado por el sistema
if x==1
    E(1)=input(' ingrese el ruido en la antena1.....ruido1=');
    ruido=E(1);
end

if x==2
    E(1)=input('ingrese el ruido en la antena1.....ruido1==');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    ruido=(E(1)+E(2))/2;
end

if x==3
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    ruido=(E(1)+E(2)+E(3))/3;
end

if x==4
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    ruido=(E(1)+E(2)+E(3)+E(4))/4;
end
if x==5
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    ruido=(E(1)+E(2)+E(3)+E(4)+E(5))/5;
end
opcion_ruido=2;
disp('si desea que el ruido adicional vaya a la señal a la entrada
digite 1');
disp(' de lo contrario se considerara que el ruido adicional se
agregara a la señal de referencia');
opcion_ruido=input('opcion_ruido=');
```

```
error=input('ingrese el error del sistema=');
% numero de iteraciones del proceso hasta que se cumpla el error
opcion_senal=2;
disp('si desea que la señal de entrada se use como señal de
referencia digite 1');
disp(' de lo contrario se considerara otra señal diferente como
señal de referencia ');
opcion_senal=input('opcion_ruido=');
% numero de iteraciones del proceso hasta que se cumpla el error
N=2000;
j=0;
% vamos a considerar que todas las antenas tienen la misma señal a
la
% entrada del filtro por ejemplo  entrada=2*sin((2*pi*k)/N)
if x==1
    h=[D(1)]
    p=[D(1)]
end
if x==2
    h=[D(1);D(2)]
    p=[(D(1)+D(2))/2]+[0.8*(D(1)+D(2))/2];
end
if x==3
    h=[D(1);D(2);D(3)]
    p=[(D(1)+D(2)+D(3))/3]+[0.8*(D(1)+D(2)+D(3))/3];
end
if x==4
    h=[D(1);D(2);D(3);D(4)]
    p=[(D(1)+D(2)+D(3)+D(4))/4]+[0.8*(D(1)+D(2)+D(3)+D(4))/4];
end
if x==5
    h=[D(1);D(2);D(3);D(4);D(5)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5))/5]+[0.8*(D(1)+D(2)+D(3)+D(4)+D(5))/5];
end
M=30;
for k=1:400
entrada(k)=sin((2*pi*k)/M);
% senal_referencia(k)=cos((2*pi*k)/M); % señal de referencia en
algunos libros se la llama señal deseada
end
for k=1:400
%entrada(k)=sin((2*pi*k)/M);
senal_referencia(k)=cos((2*pi*k)/M); % señal de referencia en
algunos libros se la llama señal deseada
end
% señal a la entrada del filtro, genera 2000 valores aleatorios de
muestra
entrada';
entrada=entrada'
if opcion_ruido==1
    entrada=entrada+ruido
end
```

```
n = sqrt(0.05)*randn(400,1); % ruido a la entrada del filtro, genera
2000 valores aleatorios de muestra
[b,a] = butter(2,0.25) % usaremos otra forma para calcular a y b
Gz = tf(b,a,-1) % la funcion de transferencia tf funciona en matlab
7
% esta funcion permite resolver la transformada z,
% el primer orden evalua valores de pesos con la funcion ldiv
%h=ldiv(b,a,orden)';
% if you use ldiv this will give h :filter weights to be
% ingreso de los voltajes en la señal de entrada (estos voltajes
serán
% los valores de pesos iniciales que serviran para hallar los pesos
W
y = lsim(Gz,entrada) % simula respuestas arbitrarias de entrada,
relaciona la señal Gz con la entrada
% para agregar el ruido
% este es el ruido que se genera internamente por el metodo LMS
% por lo general el valor de ruido que genera el algoritmo LMS puede
llegar al 10 % de la señal
n = n * std(y)/(10*std(n)); % aqui se genera el ruido del sistema
% el valor a la entrada del filtro es la suma de la señal entrante
mas el ruido
if opcion_senal==1
senal_referencia = y + n;
end
if opcion_senal>1
    senal_referencia'
    senal_referencia=senal_referencia'
    senal_referencia=senal_referencia+n
end
if opcion_ruido>1
    senal_referencia=senal_referencia+ruido
end
totallength=size(senal_referencia,1) % da el largo de la función de
entrada

% conociendo los voltajes de las antenas podemos determinar la
potencia en ellas
% potencia de la señal a la salida (potencia del sistema
potencia=mean(senal_referencia.^2)
if x==1
% tambien calcularemos la potencia para cada una de las antenas
% antena1
entrada1=D(1)+n+E(1);
potencia1=mean(entrada1.^2)
end
if x==2
% antena1
entrada1=D(1)+n+E(1);
potencia1=mean(entrada1.^2)
% antena2
entrada2=D(2)+n+E(2);
potencia1=mean(entrada2.^2)
end
```

```
if x==3
% antena1
entrada1=D(1)+n+E(1);
potencia1=mean(entrada1.^2)
% antena2
entrada2=D(2)+n+E(2);
potencia1=mean(entrada2.^2)
% antena3
entrada3=D(3)+n+E(3);
potencia3=mean(entrada3.^2)
end
if x==4
% antena1
entrada1=D(1)+n+E(1);
potencia1=mean(entrada1.^2)
% antena2
entrada2=D(2)+n+E(2);
potencia1=mean(entrada2.^2)
% antena3
entrada3=D(3)+n+E(3);
potencia3=mean(entrada3.^2)
% antena4
entrada4=D(4)+n+E(4);
potencia4=mean(entrada4.^2)
end
if x==5
% antena1
entrada1=D(1)+n+E(1);
potencia1=mean(entrada1.^2)
% antena2
entrada2=D(2)+n+E(2);
potencia2=mean(entrada2.^2)
% antena3
entrada3=D(3)+n+E(3);
potencia3=mean(entrada3.^2)
% antena4
entrada4=D(4)+n+E(4);
potencia4=mean(entrada4.^2)
% antena5
entrada5=D(5)+n+E(5);
potencia5=mean(entrada5.^2)
end

% calculo de la cota de la contante de ajuste
cota=1/((orden+1)*potencia);
%para 60 puntos  por corrida determinamos los pesos w
N=60
%begin of algorithm
w = zeros ( orden  , 1 )
for n = orden : N
    u = entrada(n:-1:n-orden+1);
    y(n)= w' * u;  % y(n) señal a la salida
    % e(n) = senal_referencia(n) - y(n);
    e(n)=senal_referencia(n)-y(n);
```

```
    j=j+1 % nos da el numero de iteraciones hasta que se cumpla el
error
     if e(n)<error
        fprintf('el numero de iteraciones hasta que se completo el
error ');
        j
     end

    % aquí determinamos el error del sistema
% mientras mayor sea el valor de constante de ajuste
% obtenemos una convergencia rápida pero menor exactitud
% en la respuesta de los pesos, mientras que si mu es
% pequeño la convergencia es más lenta pero calcula mejor los pesos
% Start with big mu for speeding the convergence then slow down to
reach the correct weights
% se puede lograr mayor exactitud en la curva variando el valor de
mu
% si se coloca valores de mu muy altos se pierde la señal de ajuste
% con valores de mu más pequeños se hace más lenta la convergencia
    if n < 20
        mu=cota/2
    else
        mu=cota/4
    end
    w = w + mu * u * e(n);
    % algunos autores usan el valor de w = w + 2*mu * u * e(n); para
    % evaluar los pesos, y hay personas que están tratando de
mejorar         %esta ecuacion
end
% chequeo de resultados de error
for n =  N+1 : totallength
    u = entrada(n:-1:n-orden+1) ;
    y(n) = w' * u ;   % y(n) señal a la salida
    % e(n) = senal_referencia(n) - y(n) ;
    e(n)=senal_referencia(n)-y(n);
end
hold on
plot(senal_referencia)
plot(y,'r');
title('salida del sistema') ;
xlabel('numero de muestras ')
ylabel('valor estimado a la salida ')
figure
semilogy((abs(e))) ;
title('curva de error') ;
xlabel('numero de muestras ')
ylabel('valor de error')
figure
plot(h, 'k+')
hold on
plot(w, 'r*')
legend('pesos actuales','pesos estimados')
title('comparacion de los pesos actuales con los pesos estimados') ;
```

```
% axis([coordenada eje x- coordenada eje x+  coordenada y-
coordenada y+])
axis([0 6 -1 1])
```

3.7 Descripción del ejemplo a correr en el programa.

Ingresaremos 2 curvas, la una será la señal de entrada:
entrada(k) = sin((2*pi*k)/M) y la segunda será la señal de referencia: señal_referencia(k) = cos((2*pi*k)/M) ingresaremos el orden del filtro, y el numero de valores reales (numero de antenas), también ingresaremos la señal de transferencia del filtro, también ingresaremos el error deseado, el algoritmo lms y rls suelen tener un error menor de 1 millonésima 0.0000000001, se ingresara un ruido adicional a las antenas, escogiéndose si se la quiere agregar el ruido a la entrada o a la salida del programa.

3.8 Pruebas y resultados de las implementaciones LMS.

Primero explicaremos como están desarrolladas las aplicaciones en matlab, que elementos usan, sus valores de entrada y sus valores de salida, estos valores los pondremos en tablas, utilizaremos estas aplicaciones con señales senosoidales y con señales gausianas.

En este primer ejemplo, utilizaremos el programa problema3_lms con los siguientes valores: entrada(k)=sin((2*pi*k)/M); senal_referencia(k) = cos((2*pi*k)/M); con mu= cota/4 y mu=cota/2, para N=400 muestras. Consideraremos que las señales vienen de diferentes direcciones, y el tipo

de señal es la misma. Consideraremos valores puntuales ingresados por teclado como los valores de las antenas. Los valores de mu son el valor de la constante de ajuste del algoritmo lms.

Para todos los ejemplos que utilicen el programa problema3_lms son validos los siguientes objetivos: demostrar que se pueden utilizar funciones senosoidales en algoritmos lms y evaluar los pesos estimados de antenas con el algoritmo lms. Comparar los valores obtenidos por estas funciones senosoidales con valores reales ingresados por el usuario. Hacer un grafico de esta comparación, hacer un grafico del error y un grafico de comparación entre la señal de referencia y la señal obtenida por la simulación. También determinar que las señales senosoidales pueden aplicarse en los algoritmos lms al igual como se aplican las señales gausianas. tal como lo habían propuesto Bernard Widrow y Ted Hoff , y anteriormente Wiener el creador del filtro de Wiener. Otros objetivos de la presente corrida es ver la correlación del número de muestras con la convergencia de la señal, también con cuantas muestras se obtiene la mejor comparación de las señales de salida del filtro con la señal de referencia de la señal senosoidal, también se consideraría como otro objetivo del presente programa el ver si este tipo de datos obtenidos en forma experimental podrían darnos valores aproximados de antenas reales y evitar construir un sistema costoso de antenas para obtener este tipo de valores de prueba, esto

podría bajar costos a alguna empresa que trabaje con señales senosoidales en condiciones parecidas al presente programa.

Desarrollo: En la corrida ingresaremos un filtro de orden = 5, a continuación el numero de antenas (n=5), con los siguientes valores de voltaje v1=0.5 V, v2=0.2 V, v3=02 V, v4=-05 V, v5=-0.8 V. ingresaremos el ruido de las antenas con valores de ruido1=0.01 V, ruido2=0.01 V, ruido3=0.01V, ruido4=0.01 V y ruido5= 0.01V. se pueden ingresar un diferente numero de orden y de antenas, pero al hacer esto tiene que aumentar el error (teoría de wiener)

A continuación hay una opción para elegir colocar este ruido adicional a la entrada del filtro (opción_ruido=1) o a la salida del filtro (opción_ruido=2) En caso que no lo escojamos automaticamente agregara este ruido adicional a la salida del filtro. Es decir el ruido adicional se suma a la señal de referencia. En el presente ejemplo (problema3_lms_a) escogeremos la opción 1 para colocar el ruido adicional a la entrada del filtro.

A continuación ingresaremos el error del sistema = 0.0000001. después de esto nos saldrá el mensaje que debemos ingresar la opción_señal (la opción_señal si es igual a 1 escogera como señal de referencia a la misma señal de entrada) ,(en caso de escoger opción_señal=2 se escogerá una señal

diferente como señal de referencia, esta señal debe de tener una alta correlación o similitud con la señal de entrada, cada una de estas opciones tiene ciertas ventajas, según el tipo de programa y modelo que se esté utilizando.) Los valores de el filtro serán dados por el usuario y otros calculados internamente en el programa, como son los valores a y b (coeficientes de la función), y el valor de la señal de transferencia Gz. Utilizaremos la misma señal de referencia dada por el ejemplo del demo de la central matlab en los programas de la presente tesis. Pero se puede utilizar otras funciones de referencia si es requerido, cambiando la formula.

.b = 0.09763 z^2 + 0.1953 z + 0.09763, .a = z^2 - 0.9428 z + 0.3333

Gz = b / a.

El programa al correr determinara la potencia del sistema y luego la potencia de cada una de las antenas. Luego evaluara los gráficos de comparación de pesos (comparara los valores reales, ingresados por teclado (los valores de las antenas (con negro)) con los valores evaluados por el programa (color rojo). Sera un grafico amplitud señal (V) vs B, donde B es un valor constante, Es decir en el eje de las y estará el valor de la amplitud (en V) vs un valor de B=6 en el

eje de las x, en este valor de 6 estarán repartidas las muestras evaluadas por el programa, también se hará una grafica de el error en función de la amplitud de la señal vs el número de muestras. Y al final un grafico de comparación de la señal de referencia con la señal del sistema a la salida. En amplitud de la señal vs el numero de muestras (la señal de referencia es la señal azul, y la señal a la salida del programa es la roja).

TABLA VII Datos de entrada para el Problema3_lms_a

Tipo de señal utilizada: senosoidal

Algoritmo utilizado: lms

Aplicación en matlab utilizado: problema3_lms

Nombre dado al Ejemplo: problema3_lms_a

Orden del filtro: 5

Numero de antenas: 5

Valores iniciales de entrada para las antenas:

V1= 0.5 V, V2=0.2 V, V3=-0.2 V, V4=-0.5 V, V5=-0.8 V

Valores de ruido ingresados para las antenas:

Ruido1= 0.01 V, ruido2=0.01 V, ruido3= 0.01, ruido4=0.01, ruido5=0.01

Opción_ruido=1, opción_señal=1

numero de muestras: 400

constantes de ajuste: mu=0.32 y mu=0.15

Error del sistema: 0.0000001

TABLA VIII Datos obtenidos por el ejemplo Problema3_lms_a

Pesos reales:	0.5 V	0.2 V	-0.2 V	-0.5 V	-0.8 V
Pesos obtenidos lms:	0.49V	0.19V	-0.1V	-0.4 V	-0.7V

Para ver las pantallas de visualización de las potencias del sistema, señal obtenida a la salida, comparación de señales y curvas de error del presente ejemplo ir al anexo F ejemplo problema3_lms_a

Ejemplo problema2_lms (programa2_lms_a)

Nota1.- Los programas problema1_lms y problema3_lms utilizan señales senosoidales como señales de entrada y de referencia, el programa2_lms utiliza señales gausianas, tanto como señales de entrada, como de referencia.

Nota2.- Solamente el programa problema3_lms tiene la opción senal_referencia para cambiar la señal de referencia.

En este segundo ejemplo, utilizaremos el problema2_lms con los siguientes valores: senal_de_entrada=rand(N,1); señal de ruido n=n*std(y)/(10*std(n)) , donde y=lsim(Gz,entrada) , la señal de ruido debe de ser del mismo tipo que la señal de entrada. Gz es una función de transferencia.

Utilizaremos la senal_referencia(k)= y + n; con mu= 0.32 y mu=0.15, para N=2000 muestras. Cuando fue creado el algoritmo lms en 1959 fue creado

para trabajar con señales gausianas, porque por sus características cumplían los requisitos para funcionar con el filtro de wiener.

Para todos los ejemplos que utilicen el programa problema2_lms son validos los siguientes objetivos: demostrar que los valores obtenidos con funciones gausianas son mas exactas que los obtenidos con las funciones senosoidales, tal como lo había propuesto Bernard Widrow, Ted Hoff y anteriormente Wiener el creador del filtro de Wiener. Comparar estas funciones gausianas con valores reales ingresados por el usuario. Hacer un grafico del error y un grafico de comparación entre la señal de referencia y la señal obtenida por la simulación. Otros objetivos de la presente corrida es ver la correlación del número de muestras con la convergencia de la señal es decir con cuantas muestras se obtiene la mejor comparación de las señales de salida del filtro con la señal de referencia de la señal gausiana, también se consideraría como otro objetivo del presente programa que este tipo de datos obtenidos en forma experimental podrían darnos valores aproximados de antenas reales y evitar construir un sistema costoso de antenas para obtener este tipo de valores de prueba, esto podría bajar costos a alguna empresa que trabaje con señales gausianas en condiciones parecidas al presente programa.

Desarrollo: En la corrida ingresaremos un orden de filtro = 5, a continuación el numero de antenas (n=5), con los siguientes valores de voltaje v1=0.0976 V, v2=0.2873 V, v3=0.3360 V, v4=0.2210 V, v5=0.0964 V. ingresaremos el ruido de las antenas con valores de ruido1=0.01 V, ruido2=0.01 V, ruido3=0.01V, ruido4=0.01 V y ruido5= 0.01V.

A continuación hay una opción para elegir colocar este ruido adicional a la entrada del filtro (opción_ruido=1) o a la salida del filtro (opción_ruido=2) En caso que no lo escojamos automaticamente agrega este ruido adicional a la salida del filtro. Es decir el ruido adicional se suma a la señal de referencia. En el presente ejemplo (problema2_lms_a) escogeremos la opción 1 para colocar el ruido adicional a la entrada del filtro.

A continuación ingresaremos el error del sistema = 0.0000001. los programas problema1 y problema2 no tienen la opción_señal para cambiar la señal de referencia. Los valores de el filtro serán dados por el usuario y otros calculados internamente en el programa, como son los valores a y b (coeficientes de la función), y el valor de la señal de transferencia Gz. Utilizaremos la misma señal de referencia dada por la central matlab como ejemplo. Pero se puede utilizar otras funciones de Tranferencia en caso de ser requerido. Tal como hicimos con el ejemplo problema3_lms_a

.b = 0.09763 z^2 + 0.1953 z + 0.09763, .a = z^2 - 0.9428 z + 0.3333

Gz = b / a.

El programa al correr determinara la potencia del sistema y luego la potencia de cada una de las antenas. Luego evaluara los gráficos de comparación de pesos (comparara los valores reales, ingresados por teclado (los valores de las antenas (con negro)) con los valores evaluados por el programa (color rojo). Sera un grafico amplitud señal (V) vs B, donde B es un valor constante, Es decir en el eje de las y estará el valor de la amplitud (en V) vs un valor de B=6 en el eje de las x, en este valor de 6 estarán repartidas las muestras evaluadas por el programa, también se hará una grafica de el error en función de la amplitud de la señal vs el número de muestras.

Y al final un grafico de comparación de la señal de referencia con la señal del sistema a la salida. En amplitud de la señal vs el numero de muestras (la señal de referencia es la señal azul, y la señal a la salida del programa es la roja).

En este segundo ejemplo utilizaremos los valores recomendados por la biblioteca matlab. Numero de muestras de N= 2000, mu=0.32 y mu=0.15. y utilizando una funcion aleatoria gaussiana como señal de entrada , con un ruido del 10%

TABLA IX Datos de entrada para el Problema2_lms_a

Tipo de señal utilizada: gaussiana

Algoritmo utilizado: lms

Aplicación en matlab utilizado: problema2_lms

Nombre dado al Ejemplo: problema2_lms_a

Orden de la señal: 5

Numero de antenas: 5

Valores iniciales de entrada para las antenas

V1= 0.0976 V, V2=0.2873 V, V3=0.3360 V, V4=0.2210 V, V5=0.0964 V

Valores de ruido ingresados para las antenas:

Ruido1= 0.01 V, ruido2=0.01 V, ruido3= 0.02, ruido4=0.03, ruido5=0.01

Opción_ruido=1, numero de muestras =2000

Constante de ajuste: mu=0.32 y mu=0.15

Error del sistema = 0.0000001

TABLA X Datos obtenidos por el ejemplo Problema2_lms_a

Pesos reales:	0.0979V	0.2873V	0.3360V	0.2210V	0.0964V
Pesos obtenidos lms:	0.03V	0.3V	0.32V	0.24V	0.11V

Para ver las pantallas de visualización de las potencias del sistema, señal obtenida a la salida, comparación de señales y curvas de error del presente ejemplo ir al anexo F ejemplo problema2_lms_a

TABLA XI Datos de entrada para el Problema2_lms_b

Tipo de señal utilizada: gaussiana

Algoritmo utilizado: lms

Aplicación en matlab utilizado: problema2_lms

Nombre dado al ejemplo: problema2_lms_b

Orden de la señal=5

Numero de antenas=5

Valores iniciales de entrada para las antenas:

V1= 0.0976 V, V2=0.2873 V, V3=0.3360 V, V4=0.2210 V, V5=0.0964 V

Valores de ruido ingresados para las antenas:

Ruido1= 0.01 V, ruido2=0.01 V, ruido3= 0.02, ruido4=0.03, ruido5=0.01

Opción_ruido=1, numero de muestras =2000

Constante de ajuste: mu=0.32 y mu=0.15

Error del sistema = 0.0000001.

TABLA XII Datos obtenidos por el ejemplo Problema2_lms_b

Pesos reales:	0.0979V	0.2873V	0.3360V	0.2210V	0.0964V
Pesos obtenidos lms:	0.18V	0.25V	0.35V	0.22V	0.025V

Para ver las pantallas de visualización de las potencias del sistema, señal obtenida a la salida, comparación de señales y curvas de error del presente ejemplo ir al anexo F ejemplo problema2_lms_b

Ahora compararemos la evolución del error si el número de orden del filtro es diferente al número de antenas (que también es el número de pesos ,valores reales utilizados en el programa) por teoría el error debe de aumentar mientras mayor sea la diferencia de orden con el numero de valores ingresados de valores reales. Utilizaremos un filtro de orden = 8 y 5 como numero de antenas. (problema2_lms_c)

TABLA XIII Datos de entrada para el problema2_lms_c

Tipo de señal utilizada: gaussiana

Algoritmo utilizado: lms

Aplicación en matlab utilizado: problema2_lms

Nombre dado al ejemplo: problema2_lms_c

Orden de la señal=8

Numero de antenas=5

Valores iniciales de entrada para las antenas:

V1= 0.0976 V, V2=0.2873 V, V3=0.3360 V, V4=0.2210 V, V5=0.0964 V

Valores de ruido ingresados para las antenas:

Ruido1= 0.01 V, ruido2=0.01 V, ruido3= 0.01, ruido4=0.01, ruido5=0.01

Opción_ruido=1, numero de muestras =2000

Constante de ajuste mu=0.32 y mu=0.15

Error del sistema = 0.0000001.

TABLA XIV Datos obtenidos por el ejemplo Problema2_lms_c

Pesos reales:	0.0976V	0.2873V	0.3360V	0.2210V	0.0964V
Pesos obtenidos lms:	< 0.06V	0.15V	> 0.35V	0.32V	0.04V

Para ver las pantallas de visualización de las potencias del sistema, señal obtenida a la salida, comparación de señales y curvas de error del presente ejemplo ir al anexo F ejemplo problema2_lms_c

Ahora probaremos el programa problema2 con un valor de filtro de orden 5 y 4 valores reales de entrada. (problema2_lms_d)

TABLA XV Datos de entrada para el Problema2_lms_d

Tipo de Señal utilizada: gaussiana

Algoritmo utilizada: lms

Aplicación en matlab utilizado: problema2_lms

Nombre dado al ejemplo: problema2_lms_d

Orden del filtro: 5

Numero de antenas: 4

Valores iniciales de entrada para las antenas:

V1= 0.0976 V, V2=0.2873 V, V3=0.3360 V, V4=0.2210 V

Valores de ruido ingresados para las antenas:

Ruido1= 0.01 V, ruido2=0.01 V, ruido3= 0.02, ruido4=0.03

Opción_ruido=1, numero de muestras =2000

Constante de ajuste: mu=0.32 y mu=0.15

Error del sistema = 0.0000001.

TABLA XVI Datos obtenidos por el ejemplo Problema2_lms_d

Pesos reales:	0.0976V	0.2873V	0.3360V	0.2210V
Pesos obtenidos lms:	0.1V	0.33V	0.35V	0.21V

Para ver las pantallas de visualización de las potencias del sistema, señal obtenida a la salida, comparación de señales y curvas de error del presente ejemplo ir al anexo F ejemplo problema2_lms_d

Ahora evaluaremos el mismo problema para solamente 200 muestras para ver si también aumenta el error. Por teoría al haber menos muestras (número de iteracciones) el error debe de aumentar. (problema2_lms_e)

TABLA XVII Datos de entrada para el Problema2_lms_e

Tipo de señal utilizada: gaussiana

Algoritmo utilizado: lms

Aplicación en matlab utilizado: problema2_lms

Nombre dado al ejemplo: problema2_lms_e

Orden del filtro: 5

Numero de antenas: 5

Valores iniciales de entrada para las antenas:

V1= 0.0976 V, V2=0.2873 V, V3=0.3360 V, V4=0.2210 V, V5=0.0964 V

Valores de ruido ingresados por las antenas:

Ruido1= 0.01 V, ruido2=0.01 V, ruido3= 0.02, ruido4=0.03, ruido5=0.01

Opción_ruido=1, numero de muestras =200

Constante de ajuste: mu=0.32 y mu=0.15

Error del sistema = 0.0000001.

TABLA XVIII Datos obtenidos por el ejemplo Problema2_lms_e

Pesos reales:	0.0976V	0.2873V	0.3360V	0.2210V	0.0964V
Pesos obtenidos lms:	0.05V	0.26V	0.32V	0.25V	0.12V

Para ver las pantallas de visualización de las potencias del sistema, señal obtenida a la salida, comparación de señales y curvas de error del presente ejemplo ir al anexo F ejemplo problema2_lms_e

A continuación realizaremos otro ejemplo variando el valor de mu (valor de la constante de ajuste del algoritmo lms), según la teoría del algoritmo LMS al variar el valor de la constante de ajuste mu se debe de cambiar más rápidamente la señal y esto puede causar que aumente considerablemente el error. Usaremos los valores de mu=0.50 y mu =0.30 para 200 muestras. (problema2_lms_f). Con los siguientes datos:

TABLA XIX Datos de entrada para el Problema2_lms_f

Tipo de señal utilizada: gaussiana

Algoritmo utilizado: lms

Aplicación en matlab utilizado: problema2_lms

Nombre dado al ejemplo: problema2_lms_f

Orden del filtro=5

Numero de antenas=4

Valores iniciales de entrada para las antenas:

V1= 0.0976 V, V2=0.2873 V, V3=0.3360 V, V4=0.2210 V

Valores de ruido ingresados para las antenas:

Ruido1= 0.01 V, ruido2=0.01 V, ruido3= 0.02, ruido4=0.03

Opción_ruido=1, numero de muestras =200

Constante de ajuste: mu=0.50 y mu=0.30

Error del sistema = 0.0000001.

TABLA XX Datos obtenidos por el ejemplo Problema2_lms_f

Pesos reales:	0.0976V	0.2873V	0.3360V	0.2210V
Pesos obtenidos lms:	0.27V	0.32V	0.05V	no sale en grafico

En este ejemplo problema2_lms_f vemos como aumenta considerablemente el error al variar el valor de la constante de ajuste mu.

Para ver las pantallas de visualización de las potencias del sistema, señal obtenida a la salida, comparación de señales y curvas de error del presente ejemplo ir al anexo F ejemplo problema2_lms_f

Ahora realizaremos el mismo problema, pero utilizando la formula de cota, la cual evaluara internamente el valor de mu, para N=2000 muestras mu=cota/2 y mu=cota/4. (problema2_lms_g). Con los siguientes datos:

TABLA XXI Datos de entrada para el Problema2_lms_g

Tipo de señal utilizada: gaussiana

Algoritmo utilizado: lms

Aplicación en matlab utilizado: problema2_lms

Nombre dado al ejemplo problema2_lms_g

Orden del filtro: 5

Numero de antenas: 4

Valores iniciales de entrada para las antenas:

V1= 0.0976 V, V2=0.2873 V, V3=0.3360 V, V4=0.2210 V

Valores de ruido ingresados para las antenas:

Ruido1= 0.01 V, ruido2=0.01 V, ruido3= 0.02, ruido4=0.03

Opción_ruido=1, numero de muestras =2000

Constante de ajuste: mu=cota/2 y mu=cota/4

Error del sistema = 0.0000001.

TABLA XXII Datos obtenidos por el ejemplo Problema2_lms_g

Pesos reales:	0.0976V	0.2873V	0.3360V	0.2210V
Pesos obtenidos lms:	0.08V	0.31V	0.32V	0.23V

Para ver las pantallas de visualización de las potencias del sistema, señal obtenida a la salida, comparación de señales y curvas de error del presente ejemplo ir al anexo F ejemplo problema2_lms_g

Ahora evaluaremos que sucede si los ruidos adicionales los agregamos a la salida del sistema es decir junto a la señal de referencia. (opción_ruido=2) en el problema2. (problema2_lms_h)

TABLA XXIII Datos de entrada para el Problema2_lms_h

Tipo de señal utilizada: gaussiana

Algoritmo utilizado: lms

Aplicación en matlab utilizado: problema2_lms

Nombre dado al ejemplo: problema2_lms_h

Orden del filtro=5

Numero de antenas=5

Valores iniciales de entrada para las antenas:

V1= 0.0976 V, V2=0.2873 V, V3=0.3360 V, V4=0.2210 V, V5= 0.0964 V

Valores de ruido ingresados para las antenas:

Ruido1= 0.01 V, ruido2=0.01 V, ruido3= 0.02, ruido4=0.03, ruido5=0.01

Opción_ruido=2, numero de muestras =2000

Constante de ajuste: mu=cota/2 y mu=cota/4

Error del sistema = 0.0000001

TABLA XXIV Datos obtenidos por el ejemplo Problema2_lms_h

Pesos reales:	0.0976V	0.2873V	0.3360V	0.2210V	0.0964V
Pesos obtenidos lms:	0.1V	0.3V	0.332V	0.22V	0.1V

Para ver las pantallas de visualización de las potencias del sistema, señal obtenida a la salida, comparación de señales y curvas de error del presente ejemplo ir al anexo F ejemplo problema2_lms_h

Ahora utilizaremos el programa problema1_lms

Utlizaremos como señal de entrada `entrada(k)=sin((2*pi*k)/M);` Para N=200 muestras, y valores de mu (valor de la contante de ajuste) de mu=cota/2 y mu=cota/4 , (Valores de mu recomendados por la central matlab son 0.32 y 0.15) después probaremos con otros valores de mu..

La señal de referencia debe de ser una curva que sea muy cercana o correlacionada a la señal de entrada por ejemplo senal_referencia(k) =1.1sin((2*pi*k)/M); o senal_referencia(k)=cos((2*pi*k)/M);etc. En este ejemplo usaremos la misma señal de entrada con un pequeño ruido adicional y luego con las otras curva que mostramos. Utilizaremos la opción 1 que el ruido adicional sea agregado a la entrada antes del filtro.

Utilizaremos valores de pesos que estén cercanos a los que da la señal real. Mientras los valores de los pesos sean más lejanos de lo que evalue la curva a la entrada el error debe de ser mayor. (según la teoría del algoritmo LMS)

TABLA XXV Datos de entrada para el Problema1_lms_a

Tipo de señal utilizada: senosoidal

Algoritmo utilizado: lms

Aplicación en matlab utilizado: problema1_lms

Nombre dado al ejemplo: problema1_lms_a

Orden del filtro: 5

Numero de antenas: 5

Valores iniciales de entrada para las antenas:

V1= 0.20 V, V2=0.25 V, V3=0.21 V, V4=0.18 V, V5=0.15 V

Valores de ruido ingresados para las antenas:

Ruido1= 0.01 V, ruido2=0.01 V, ruido3= 0.01, ruido4=0.01, ruido5=0.01

Opción_ruido=1, numero de muestras =200

Constante de ajuste: mu=cota/2 y mu=cota/4

Error del sistema = 0.0000001.

TABLA XXVI Datos obtenidos por el ejemplo Problema1_lms_a

Pesos reales:	0.20V	0.25V	0.21V	0.18V	0.15V
Pesos obtenidos lms:	0.26V	0.24V	0.23V	0.19V	0.145V

Para ver las pantallas de visualización de las potencias del sistema, señal obtenida a la salida, comparación de señales y curvas de error del presente ejemplo ir al anexo F ejemplo problema1_lms_a

TABLA XXVII Datos de entrada para el Problema1_lms_b

Tipo de señal utilizada: senosoidal

Algoritmo utilizado: lms

Aplicación en matlab utilizado: problema1_lms

Nombre dado al ejemplo: problema1_lms_b

Orden del filtro: 5

Numero de antenas: 5

Valores iniciales de entrada para las antenas:

V1= 0.20 V, V2=0.25 V, V3=0.21 V, V4=0.18 V, V5=0.15 V

Valores de ruido ingresados para las antenas:

Ruido1= 0.01 V, ruido2=0.01 V, ruido3= 0.01, ruido4=0.01, ruido5=0.01

Opción_ruido=1, numero de muestras =400

Constante de ajuste: mu=cota/2 y mu=cota/4

Error del sistema = 0.0000001.

TABLA XXVIII Datos obtenidos por el ejemplo Problema1_lms_b

Pesos reales:	0.20V	0.25V	0.21V	0.18V	0.15V
Pesos obtenidos lms:	0.27V	0.251V	0.23V	0.19V	0.145V

Para ver las pantallas de visualización de las potencias del sistema, señal obtenida a la salida, comparación de señales y curvas de error del presente ejemplo ir al anexo F ejemplo problema1_lms_b

Cambiando de N=200 muestras a N=400 muestras el error parece no haber cambiado, Ahora lo evaluaremos con N=2000 muestras para ver si disminuye el error, aunque por teoría del algoritmo LMS el error debe de aumentar si aumenta la frecuencia.

TABLA XXIX Datos de entrada para el Problema1_lms_c

Tipo de señal utilizada: senosoidal

Algoritmo utilizado: lms

Aplicación en matlab utilizado: problema1_lms

Nombre dado al ejemplo: problema1_lms_c

Orden del filtro: 5

Numero de antenas: 5

Valores iniciales de entrada para las antenas:

V1= 0.20 V, V2=0.25 V, V3=0.21 V, V4=0.18 V, V5=0.15 V

Valores de ruido ingresados para las antenas:

Ruido1= 0.01 V, ruido2=0.01 V, ruido3= 0.01, ruido4=0.01, ruido5=0.01

Opción_ruido=1, numero de muestras =2000

Constante de ajuste: mu=cota/2 y mu=cota/4

Error del sistema = 0.0000001

TABLA XXX Datos obtenidos por el ejemplo Problema1_lms_c

Pesos reales:	0.20V	0.25V	0.21V	0.18V	0.15V
Pesos obtenidos lms:	0.25V	0.23V	0.20V	0.17V	0.13V

Para ver las pantallas de visualización de las potencias del sistema, señal obtenida a la salida, comparación de señales y curvas de error del presente ejemplo ir al anexo F ejemplo problema1_lms_c

Ahora evaluaremos la misma ecuación de entrada, para N=400, con valores de constante de ajuste de mu=cota/2 y mu=cota/4 para ver la evolución del error si el numero de orden del filtro es diferente al número de antenas (que también es el numero de pesos ,valores reales utilizados en el programa) por teoría el error debe de aumentar mientras mayor sea la diferencia de orden con el numero de valores ingresados de valores reales.

TABLA XXXI Datos de entrada para el Problema1_lms_d

Tipo de señal utilizada: senosoidal

Algoritmo utilizado: lms

Aplicación en matlab utilizado: problema1_lms

Nombre dado al ejemplo problema1_lms_d

Orden del filtro: 8

Numero de antenas: 5

Valores iniciales de entrada para las antenas:

V1= 0.20 V, V2=0.25 V, V3=0.21 V, V4=0.18 V, V5=0.15 V

Valores de ruido ingresados para las antenas:

Ruido1= 0.01 V, ruido2=0.01 V, ruido3= 0.01, ruido4=0.01, ruido5=0.01

Opción_ruido=1, numero de muestras =400

Constante de ajuste: mu=cota/2 y mu=cota/4

Error del sistema = 0.0000001

TABLA XXXII Datos obtenidos por el ejemplo Problema1_lms_d

Pesos reales:	0.20V	0.25V	0.21V	0.18V	0.15V
Pesos obtenidos lms:	0.25V	0.23V	0.19V	0.165V	0.125V

Para ver las pantallas de visualización de las potencias del sistema, señal obtenida a la salida, comparación de señales y curvas de error del presente ejemplo ir al anexo F ejemplo problema1_lms_d

Para un filtro de nivel 8 y 5 antenas se ve un pequeño aumento del error en todos los pesos, lo cual prueba que se cumple la teoría del algoritmo LMS .

Ahora lo evaluaremos disminuyendo el numero de antenas, para N=400 muestras, mu=cota /2 y mu=cota/4, orden=10 y numero de antenas =3

TABLA XXXIII Datos de entrada para el Problema1_lms_e

Tipo de señal utilizada: senosoidal

Algoritmo utilizado: lms

Aplicación en matlab utilizado: problema1_lms

Nombre dado al ejemplo: problema1_lms_e

Orden del filtro=10

Numero de antenas=3

Valores iníciales de entrada para las antenas:

V1= 0.20 V, V2=0.25 V, V3=0.21 V

Valores de ruido ingresados para las antenas:

Ruido1= 0.01 V, ruido2=0.01 V, ruido3= 0.01

Opción_ruido=1, numero de muestras =400

Constante de ajuste: mu=cota/2 y mu=cota/4

Error del sistema = 0.0000001

TABLA XXXIV Datos obtenidos por el ejemplo Problema1_lms_e

Pesos reales:	0.20V	0.25V	0.21V
Pesos obtenidos lms:	0.22V	0.21V	0.20V

Para ver las pantallas de visualización de las potencias del sistema, señal obtenida a la salida, comparación de señales y curvas de error del presente ejemplo ir al anexo F ejemplo problema1_lms_e

Ahora evaluaremos la misma ecuación de entrada, para N=400, con valores de constante de ajuste de mu=cota/2 y mu=cota/4, cambiaremos el valor del error que se agrega a las antenas por teoría el error debe de aumentar. según la teoría del algoritmo LMS. .n1=0.01, .n2=0.02 .n3=0.02 .n4=0.03 .n5=0.04

TABLA XXXV Datos de entrada para el Problema1_lms_f

Tipo de señal utilizada: senosoidal

Algoritmo utilizado: lms

Aplicación en matlab utilizado: problema1_lms

Nombre dado al ejemplo: problema1_lms_f

Orden del filtro: 5

Numero de antena: 5

Valores iniciales de entrada para las antenas:

V1= 0.20 V, V2=0.25 V, V3=0.21 V, V4=0.18 V, V5=0.15 V

Valores de ruido ingresados para las antenas:

Ruido1= 0.01 V, ruido2=0.02 V, ruido3= 0.02,ruido4=0.03,ruido5=0.04

Opción_ruido=1, numero de muestras =400

Constante de ajuste: mu=cota/2 y mu=cota/4

Error del sistema = 0.0000001

TABLA XXXVI Datos obtenidos por el ejemplo Problema1_lms_f

Pesos reales:	0.20V	0.25V	0.21V	0.18V	0.15V
Pesos obtenidos lms:	0.261V	0.24V	0.225V	0.195V	0.14V

Para ver las pantallas de visualización de las potencias del sistema, señal obtenida a la salida, comparación de señales y curvas de error del presente ejemplo ir al anexo F ejemplo problema1_lms_f

CAPÍTULO 4

EL ALGORITMO ADAPTATIVO MÍNIMO CUADRADO RECURSIVO RLS

El algoritmo adaptativo mínimo cuadrado recursivo es una extensión de el algoritmo mínimo cuadrado, presenta varias ventajas como el disminuir el número de interacciones para llegar a su convergencia, además permite hacer operaciones matemáticas más complejas que el mínimo cuadrado, su principal característica es que permite utilizar los valores obtenidos anteriormente en la convergencia para hallar el próximo valor y no solamente la diferencia de error como hace el mínimo cuadrado. En la utilización de los algoritmos RLS se presenta la oportunidad de utilizar la estimación, esto se aparta de los métodos de gradiente que utilizan los algoritmos LMS. Los algoritmos LMS utilizan el filtro de wiener para hallar su solución en cambio el algoritmo RLS utiliza el filtro de Kalman para hallar su solución, El filtro de Kalman está basado en la matriz de autocorrelación de datos y del vector P. Dichas estimaciones se realizan mediante el promedio de un numero M de muestras más recientes del vector de datos Xn y de la referencia d(n). (Habitualmente y a fin de simplificar su coste computacional, en los algoritmos RLS se suele utilizar el

promedio), esta utilización del promedio se llama IIR, (filtro de respuesta infinita) en la cual utilizan un coeficiente tao y que realiza un promediado exponencial, en base a un parámetro, esto se expresa como $1/(1-\tau)$. Tanto la matriz como el vector se actualizan con las ecuaciones R(n+1) y P(n+1).

Estas expresiones también se analizan en los algoritmos RLS , así como sus propiedades, estas ecuaciones presentan la cualidad de presentarse como un estimador de las funciones a estimar.

El algoritmo RLS es sin duda el mejor algoritmo adaptativo para la minimización del MSE. Sus prestaciones no dependen de la dispersión de auto valores. Tiene menos interacciones que el mínimo cuadrado (LMS) y permite hacer operaciones más avanzadas que el mínimo cuadrado, solo presentando un aumento pequeño en la complejidad matemática. la convergencia es del orden de la longitud del filtro, es decir, para un filtro de Q coeficientes tarda Q iteraciones o vectores de datos en converger y su desajuste se minimiza con valores de λ próximos a la unidad. Por ello, y siempre que su mayor complejidad lo permita, no tiene rival en la minimización del MSE superando el diseño del filtro de Wiener. Su éxito relativo es debido a que, dentro del área de teoría de control, Kalman mejoro el filtro de wiener y proporcionó una forma más formal de presentarlo. además el filtro de Kalman proporciona más

versatilidad a su empleo y por tanto el ámbito de su aplicación es mayor. En este mismo capítulo 4 trataremos más a fondo sobre el filtro de Kalman en el cual se verá las ventajas que tiene para el tratamiento de convergencia y desajuste.

4.1 Algoritmo RLS

La solución mínimo cuadrado (capitulo 3) no es muy práctico para la implementación de filtros adaptativos. Ya que se necesitan muchas iteraciones para llegar a su convergencia, por eso es que se ideo hacer un nuevo algoritmo que sea recursivo, que posea nuevas características y que sea implementado en señales puntuales. Una de las mayores ventajas del algoritmo RLS es su capacidad recursiva iterativa. Con la cual con muy pocas iteraciones disminuye rápidamente el error cuadrático. Una desventaja que posee el algoritmo RLS en comparación con el LMS, es que al no ser lineal siempre no posee la robustez que posee el LMS para encontrar siempre la convergencia que tiene el LMS, por eso es importante colocar los valores correctos del filtro para garantizar una buena convergencia del algoritmo RLS. El algoritmo RLS (mínimo cuadrado recursivo) está basado en la estimación del algoritmo LS (mínimo cuadrado) de un filtro de coeficiente w(n-1) una interacion n-1 puede ser rápidamente estimados con herramientas computacionales, usando el arribo de datos que se obtiene con los

valores recursivos. Este tipo de algoritmo podría ser un caso especial del filtro de kalman. La implementación del algoritmo utilizando el método recursivo requiere de valores de actualización utilizando herramientas matemáticas, esta ecuación se la puede ver en (4.1) en donde J(n) es la función costo donde n es la variable de longitud de los datos obtenidos. El algoritmo RLS presenta una mayor complejidad matemática que el algoritmo LMS, esto se verá más sencillo cuando se realice el programa, ya que muchas de estas operaciones matemáticas son las mismas del algoritmo lms y otras son realizadas por alguna función en matlab en forma interna, lo que permite que ingresemos los valores importantes en el programa y ver su desarrollo para diferentes valores.

$$J \equiv E = \sum_{n=1}^{N} g(n)e^2(n) \tag{4.1}$$

$$J(n) = \sum_{k=1}^{n} n_n(k)e^2(k), \quad n_n(k) \equiv \text{weighting factor} \tag{4.2}$$

Donde

$$e(k) = d(k) - y(k) = d(k) - w^T(k)x(k) \quad (4.3)$$

$$x(k) = [x(k) \quad x(k-1) \ldots\ldots x(k-M+1)]^T \quad (4.4)$$

$$w(n) = [w1(n) \quad w2(n) \ldots\ldots w_M(n)]^T \quad (4.5)$$

Este desarrollo matemático es igual al del algoritmo LMS, la única diferencia nueva es el valor de landa, que es un coeficiente del algoritmo RLS, un coeficiente del filtro de kalman. Esto se ve en la ecuación 4.6

$$n_n(k) = \lambda^{n-k} \qquad k=1,2,\ldots,n \quad (4.6)$$

Donde el valor de λ es menor que 1, y de $\eta_n(k)$, (este valor de landa es un coeficiente del filtro) el cual es confinado en el rango de:

$$0 < \eta_n(k) \leq 1 \text{ for } k = 1, 2, \ldots, n.$$

EL factor de peso de λ es también conocido como el factor olvidado (porque muchas veces no se piensa mucho en el, pero ha veces puede ser importante para lograr una buena convergencia) subsecuentemente el peso de datos recientes

tienden a olvidarse con el paso del tiempo. Esta propiedad es producida por los algoritmos adaptativos con alguna capacidad de rastreo. De hay que nosotros podemos minimizar la función de J(n) (4.7)

Algunas veces si la convergencia no es buena en el algoritmo RLS, lo que se hace es normalizar el algoritmo, y con eso se logra una mejor convergencia, pero aun así no es tan segura como en el algoritmo LMS.

$$J(n) = \sum_{k=1}^{n} \lambda^{n-k} e^2(k) \qquad (4.7)$$

El valor mínimo de j(n) es logrado con la ecuación normal 4.8, una característica del algoritmo rls es que al normalizar la ecuación se puede disminuir el error.

$$R_\lambda(n)\, w = P_\lambda(n) \quad (w = R_\lambda^{-1} P_\lambda(n)) \qquad (4.8)$$

Donde la matriz de correlación de MxM de la matriz $\mathbf{R}_\lambda(n)$ Nos produce valores más satisfactorios ecuaciones 4.9 y 4.10.

$$R_{\lambda}(n) = \sum_{k=1}^{n} \lambda^{n-k} x(k)x^{T}(k) = X^{T} \Lambda X \quad (4.9)$$

$$P_{\lambda}(n) = \sum_{k=1}^{n} \lambda^{n-k} x(k)d(k) = X^{T} \Lambda d \; ; \Lambda = \text{diag}[\lambda^{n-1} \; \lambda^{n-2} \; .... 1] \quad (4.10)$$

Para hacer un análisis más a fondo del algoritmo RLS para valores discretos vea el anexo F

4.1.1 Base teórica del algoritmo adaptativo RLS

El estudio del algoritmo adaptativo RLS está muy relacionado con el desarrollo del algoritmo de Wiener y el algoritmo de kalman, este tiene una amplia utilización en filtros, y se lo utiliza para hallar el valor de convergencia. Los filtros de wiener y de kalman tienen una gran importancia para las señales estadísticas, y de probabilidad. Muchos las llaman señales de estadística, este método es muy útil para resolver problemas que utilizan algoritmos con mínimo cuadrado recursivo, hasta de segundo orden, es muy práctico para determinados fragmentos, como es de suponer al aumentar el orden aumenta el error, pero si se tiene buenos valores del filtro disminuye considerablemente.

El filtro de kalman y el algoritmo adaptativo RLS pertenecen al armazón estadístico, la cual las utiliza para determinar valores, porcentajes, convergencias, etc. Para poder utilizarlas en algunas ocasiones se requiere de un conocimiento a priori de los segundos momentos de orden. Por otro lado, el método de mínimos cuadrados pertenece al marco de valores deterministicos. además, este método de mínimos cuadrados nos sirven para llegar a la convergencia del los valores deseados. una ventaja de utilizar valores con algoritmos RLS es que nos permitirá utilizar modelos lineales (ya que el algoritmo rls posee una parte lineal, aunque no por completo como el lms), esto es muy útil en telecomunicaciones ya que permite predecir la señal hasta cierto punto. A continuación se realizaremos un pequeño análisis matemático de los algoritmos RLS, primero se hablara un poco de los algoritmos LS y como luego al hacerse recursivo se llega al RLS.

4.1.2 Análisis matemático del algoritmo RLS

Antes de analizar al algoritmo RLS primero analizaremos el algoritmo mínimo cuadrado LS, Consideramos un filtro adaptable lineal a veces con

coeficientes n y un vector de la entrada con un valor real moderado (señal de entrada) y un valor de d(n) que es su señal de referencia.

Nótese que no se ha especificado para el x(n) (vector de entrada), si tiene o no ruido, y por consiguiente también puede agregársele ruido sin que afecte a la ecuación (al método de resolución), estos valores de entrada se las llamara numero de muestras M, este numero de muestras M pueden ser valores instantáneos (para valores discretos, y para valores puntuales), esto los vemos en la figura 4.1. De aquí se ve que el problema es estimar con el filtro valores de pesos W que me den valores cercanos a la señal de referencia d(n) que es la solución deseada usando la combinación lineal, de aquí obtenemos el valor de y(n) a la salida. ecuación 4.11

$$y(n) = w^{T}(n)x(n) = \sum_{k=1}^{M} w_k(n)\,x_k(n) \quad n=1,2,\ldots,N \tag{4.11}$$

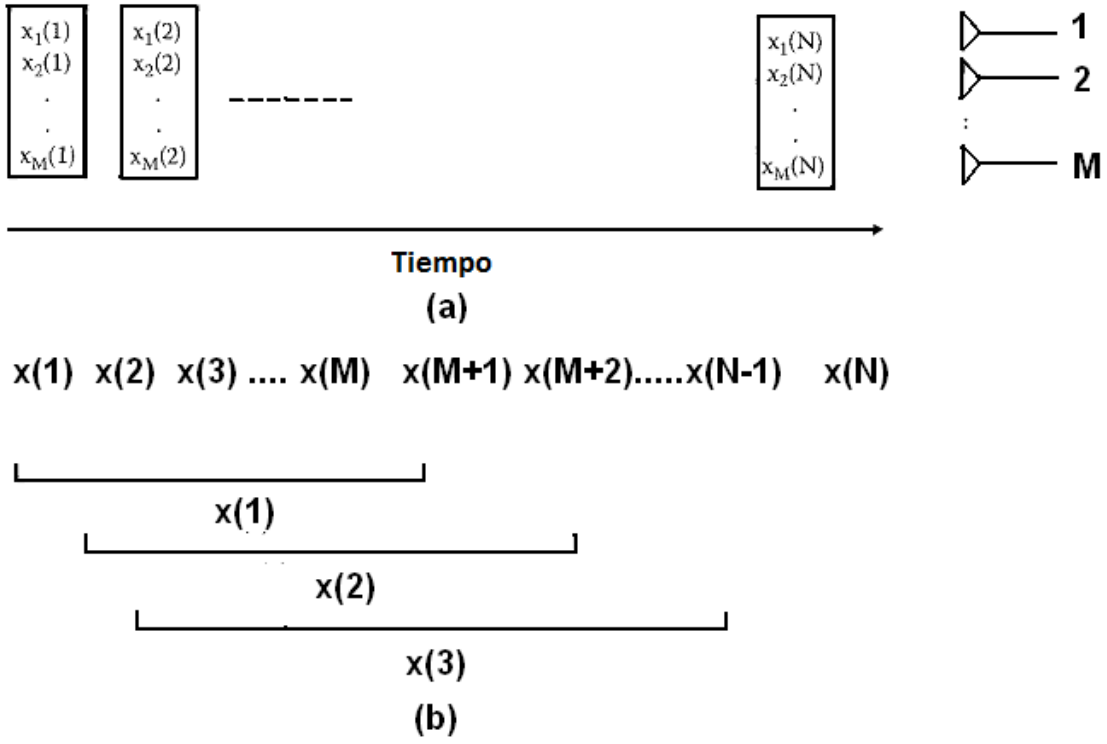


Figura 4.1 (a) Aplicación Multisensor (b) Sensor de aplicación unitario

Para calcular el error se utiliza una combinación lineal. La estimación del error está definida por la relación 4.12

$$e(n) = d(n) - y(n) = d(n) - w^T(n)x(n) \qquad (4.12)$$

Los coeficientes de los filtros adaptativos pueden ser minimizados para hallar el error cuadrado (para el mínimo cuadrado) ecuación 4.13 y la figura 4.2.

$$J \equiv E = \sum_{n=1}^{N} g(n)e^2(n) \qquad (4.13)$$

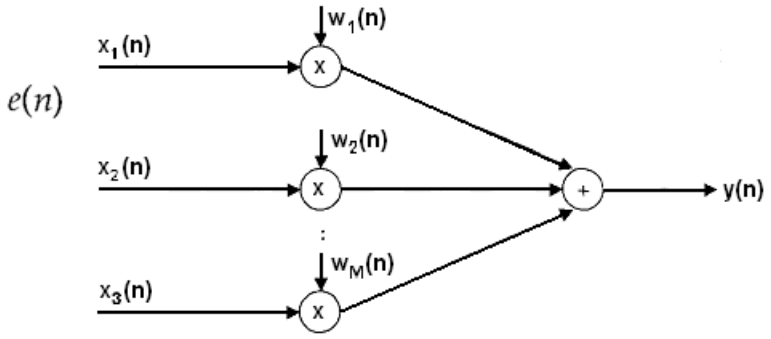


Figura 4.2 estimador lineal para un sistema de parámetro M

Si tenemos una función g(n) que es la función de pesos, utilizando el método de mínimos cuadrados en los coeficientes del filtro y optimizándolo y creamos una matriz de M x N. este le permitirá presentar tiempos y minimizar los valores de error de la señal deseada. Esto lo veremos a la

salida del sistema. La minimización es válida cuando el coeficiente vector w(n) es constante, donde w(n) es el vector de valores que serán implementados en el filtro. Los valores de n se los sitúa en un intervalo de 1 menor igual a n y n mayor igual a N. En estadística la estimación del mínimo cuadrado es conocido como regresión, e(n) es el valor del error evaluado en n. el cual es conocido como una señal, y w es el vector de regresión.

$$X^T = \begin{bmatrix} x_1(1) & x_1(2) \ldots . & x_1(N) \\ x_2(1) & x_2(2) \ldots . & x_2(N) \\ \vdots & \vdots & \vdots \\ x_M(1) & x_M(2) & x_M(N) \end{bmatrix} \quad \begin{matrix} \downarrow \\ \text{valores instantaneos} \\ \text{complementos} \end{matrix} \quad \rightarrow \text{Registro de datos } (M \times N) \qquad (4.14)$$

Donde nosotros asumimos que el valor de N tiene que ser mayor a M. esta es otra de las características del algoritmo mínimo cuadrado recursivo.

Para el caso de los algoritmos mínimo cuadrado recursivo y otros casos solemos utilizar valores de x(M) x(M+1), x(m-1) esto se lo puede ver más claramente en la ecuación 4.15.

$$X^T = \begin{bmatrix} x(M) & x(M+1) & \ldots\ldots & x(N) \\ x(M-1) & x(M) & \ldots\ldots & x(N-1) \\ x(M-2) & x(M-1) & \ldots\ldots & x(N-2) \\ \vdots & & & \\ x(1) & x(2) & \ldots\ldots & x(N-M+1) \end{bmatrix} \quad (4.15)$$

Los valores de salida de y, el error e, y los valores de dato del vector xk se lo puede expresar como :

$$y = Xw \quad (4.16)$$

$$e = d - y \quad (4.17)$$

A continuación veremos el desarrollo matemático de Y y de d, los que nos permitirá obtener las formulas para evaluar W y el función de gradiente J. Como podemos observar el desarrollo matemático del algoritmo RLS es mayor que el del algoritmo LMS, por eso algunas personas prefieren trabajar solamente con el algoritmo LMS, o a lo sumo con el RLS que tiene

muchas partes comunes con el LMS, y tratan de evitar otros algoritmos adaptativos como el algoritmo music, el algoritmo cma o el algoritmo dmi.

En el programa RLS al igual que en el LMS ingresaremos el numero de orden, el numero de antenas a utilizar y sus valores, y los ruidos de cada señal, en un lazo colocaremos los valores de la señal de entrada, y en otro la de la señal de referencia.

A continuación se hará un lazo en donde se evaluaran los pesos, que son los valores óptimos que queremos hallar, este lazo va desde el valor de orden (numero de canal del filtro) hasta un valor de N, que es el numero de valores en los que se buscara encontrar los pesos estimados, se utilizara los valores de la señal de entrada, u, este valor de u se lo multiplicara por P que es un conjunto de valores de un vector de datos, obtendremos el valor de phi con la multiplicación de u'*P, la función K es igual a phi/(landa*phi*u). su programación en matlab es la siguiente:

```
%evalua 70 puntos por corrida ( N - orden 70 = 80 - 10 )
N=80;   % N=80
%begin of the algorithm
```

```
%forgetting factor valores recomendados por la biblioteca matlab
lamda = 0.999994115 ;% lamda= 0.9995
%initial P matrix
delta=100000000000000 % delta = 1e10 ;
P = delta * eye (orden);
w = zeros ( orden  , 1 ) ;
for n = orden : N
    u = entrada(n:-1:n-orden+1);
    phi = u' * P ;
    k = phi'/(lamda + phi * u );
    y(n)=w' * u;
    e(n) = senal_referencia(n) - y(n) ;
    w = (w) + k * e(n) ;
    % w=w/10;
    P = ( P - k * phi ) / lamda;
    % ajuste del ploteo, pesos
    Recordedw(1:orden,n)=w;
end
```

.u son los valores de la señal de entrada

P conjunto de valores de un vector de datos

Phi es el producto de u'*P

K es la ganancia de kalman K=phi'/(landa+phi*u)

y(n) es la salida del filtro que sera igual a w'*u

El cálculo del error se realiza por medio de e(n)=d(n)-y(n) y con estos valores obtenemos los valores de los pesos, w=w+k*e(n) , evaluamos P, P=(P-K*phi)/landa

Y aquí termina el lazo de evaluación de pesos.

El algoritmo RLS tiene un segundo lazo para chequear la curva y ver que sea lo más exacta posible.

Las líneas de programación en matlab del lazo de control es:

```
%check of results
for n =  N+1 : totallength
   u = entrada(n:-1:n-orden+1) ;
   y(n) = w' * u ;
   e(n) = senal_referencia(n) - y(n);
end
```

La única diferencia entre el algoritmo mínimo cuadrado recursivo con el algoritmo mínimo cuadrado, es la propiedad que tiene el mínimo cuadrado recursivo de utilizar los valores anteriores para retroalimentar el sistema y minimizar el error y el número de interacciones que nos acerque a la solución buscada. Además de que el algoritmo RLS necesita de los valores de landa y de delta para evaluar los valores de los pesos.

La ecuación para métrica de y en función de x y w, puede tomar valores para N mayor igual a M. si N=M nos podría dar la ecuación de peso $\hat{\mathbf{w}}$

$$\hat{w} = X^{-1}y \tag{4.18}$$

Con estos valores evaluamos si x inversa existe nos permite encontrar $\hat{w}$ que es un estimado de w . utilizando el error mínimo cuadrado recursivo para determinar w y para valores de N mayor de M y utilizando el vector de error e=(e1 e2 e3,….eN)T. con este vector podremos hallar una nueva función que relacione w y J con el error ecuación 4.17 y 4.18 de estas obtenemos la ecuación 4.19

$$\mathbf{e} = [e_1 \;\; e_2 \cdots e_N]^{\mathrm{T}}$$

$$J = \sum_{i=1}^{N} e_i^2 = e^T e \tag{4.19}$$

Al minimizar estos valores obtenemos la ecuación 4.20

$$J = (y-Xw)^T(y-Xw) \tag{4.20}$$

$$J = y^Ty - w^TX^Ty - yXw + w^TX^TXw \tag{4.21}$$

En el algoritmo rls diferenciando J con respecto a w y la ecuación que es igualada a cero nos permitirá encontrar las condiciones adecuadas para estimar el valor de w. ecuación 4.22 y 4.23

$$\frac{\partial J}{\partial w}\Big|_{w=\hat{w}} = -2X^T y + 2X^T X\hat{w} = 0 \qquad (4.22)$$

$$X^T X\hat{w} = X^T y \qquad (4.23)$$

De aquí obtenemos $\hat{w}$ ecuación 4.24

$$\hat{w} = (X^T X)^{-1} X^T y \qquad (4.24)$$

Y este valor de los pesos W es el necesitamos para saber la evolución de la curva en el algoritmo rls.

El sistema de modelado lineal que tienen los algoritmos LS y RLS son muy útiles por su sencillez de aplicación y por dar una respuesta aceptable bajo ciertas condiciones, es importante tener en cuenta que esto solo sirve hallar la solución de problemas no tan complejos, los cuales requerirán otro tipo

de algoritmos más complejos que nos permitan llegar a la respuesta con menos iteraciones.. También hay que tener presente que sería conveniente hacer simulaciones de LS y RLS en presencia del ruido gaussiano, y de otros valores que se presentan en la vida real en las antenas y comunicaciones. Estos valores de estimación de LSE y RLS podrían verse afectados por el ruido gaussiano y otras señales.

El algoritmo RLS se suele utilizar para conformar el filtro de kalman del mismo modo que el algoritmo LMS se utiliza para conformar el filtro de wiener. Existen variaciones del algoritmo RLS que se desarrollan en forma hibrida con otros algoritmos adaptativos. Esto se realiza con el objetivo de formar algoritmos híbridos que posean características más deseables según el tipo de problema que se tenga que resolver.

Las principales formulas que se utilizan en el algoritmo RLS son:

$$\hat{w}(n) = \hat{w}(n-1) + q(n)\left[d^*(n) - \hat{w}^H(n-1)x(n)\right] \qquad (4.25)$$

$$q(n) = \frac{\gamma^{-1} R_{xx}^{-1}(n-1)x(n)}{1 + \gamma^{-1} x^H(n) R_{xx}^{-1}(n-1)x(n)} \qquad (4.26)$$

$$R_{xx}^{-1}(n) = \gamma^{-1}\left[R_{xx}^{-1}(n-1) - q(n)x(n)R_{xx}^{-1}(n-1)\right] \qquad (4.27)$$

Ahora explicaremos como funciona un combinador lineal adaptativo con un filtro FIR adaptativo (filtro de respuesta finita) cuyo esquema lo presentamos a continuación:

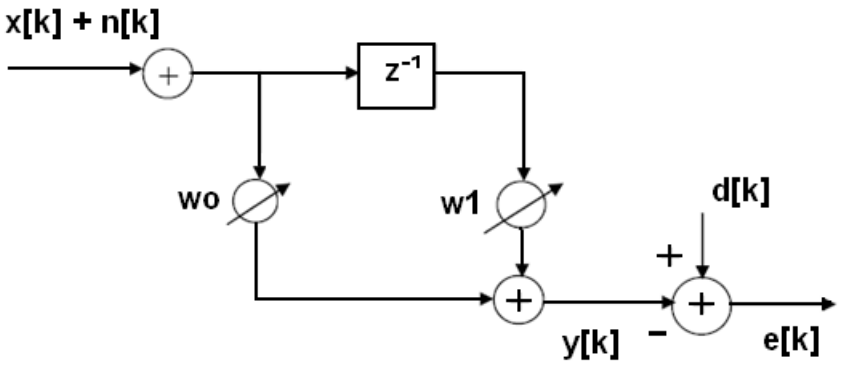


Figura 4.3 Combinador lineal adaptativo

Donde:

$$x[k] = \operatorname{sen}\left[\frac{2\pi k}{N}\right] \qquad (4.28)$$

n[k] es un ruido aditivo, blanco y gaussiano de media nula y varianza igual a 0,05 **(4.29)**

$$d[k] = 2\cos\left[\frac{2\pi k}{N}\right] \qquad (4.30)$$

Matriz de autocorrlacion de la señal de entrada.Rxx:

$$Rxx = \begin{bmatrix} 0.5 + E\left[n^2[k]\right] & 0.5 * \cos\left[\frac{2\pi}{N}\right] \\ 0.5 * \cos\left[\frac{2\pi}{N}\right] & 0.5 + E\left[n^2[k]\right] \end{bmatrix} \quad (4.31)$$

Matriz de correlacion cruzada entre la entrada y la salida deseada, Rxd:

$$R_{xd} = \begin{bmatrix} 0 \\ -\operatorname{sen}\left[\frac{2\pi}{N}\right] \end{bmatrix} \quad (4.32)$$

El objetivo es encontrar el vector de pesos óptimo que minimice el error cuadrático medio entre la salida del filtro y la salida deseada.

En esta primera parte explicaremos cómo funciona el combinador lineal (utilizando la matriz de auto correlación y la matriz cruzada)

1.- Para N = 40, realizaremos una representación gráfica de una superficie de error para la cual daremos su expresión matemática.

La expresión de la superficie de error es la siguiente:

$$\xi = E[\varepsilon_k^2] = E[d^2] + W_k^T R_{xx} W_k - 2R_{xd}^T W_k^T \qquad (4.33)$$

Sustituyendo los valores indicados:

$$\xi = E[\varepsilon_k^2] = (0.5 + E[n^2(k^2)]) \cdot (w_{0,k}^2 + w_{1,k}^2) + w_{0k} w_{1k} \cos\left[\frac{2\pi}{N}\right] + 2w_{1k}\, sen\left[\frac{2\pi}{N}\right] + 2 \qquad (4.34)$$

En la figura 4.4 vemos la representación de la superficie de error.

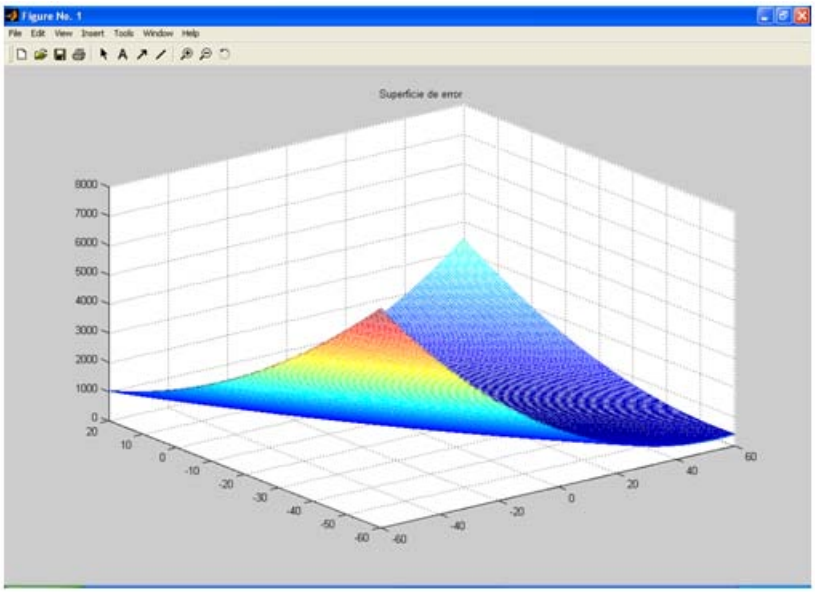


Figura 4.4 Representacion de la superficie de error en matlab

También se puede obtener una representación en curvas de nivel mediante .la función **contour(·)** de Matlab:

figure(2); contour(w0k,w1k,sup_error)

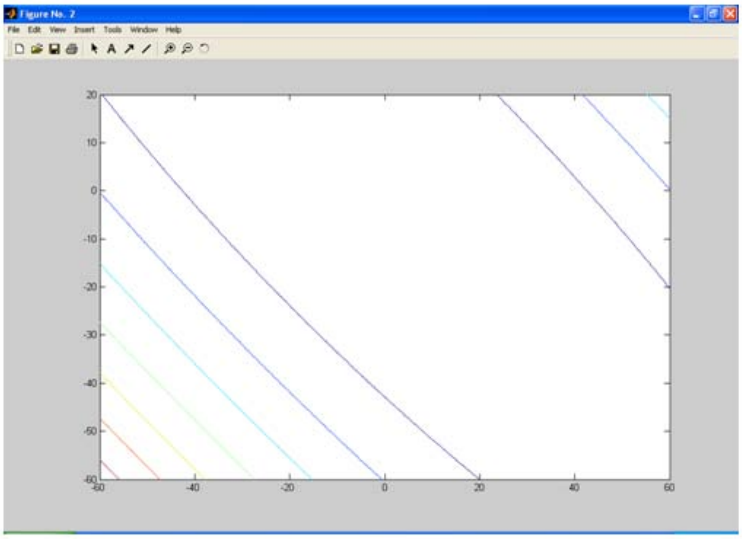


Figura 4.5 Representacion de las curvas de nivel en matlab

2 Calcularemos el vector de pesos óptimos y el error cuadrático medio mínimo para los valores planteados al inicio. Debemos de recordar que:

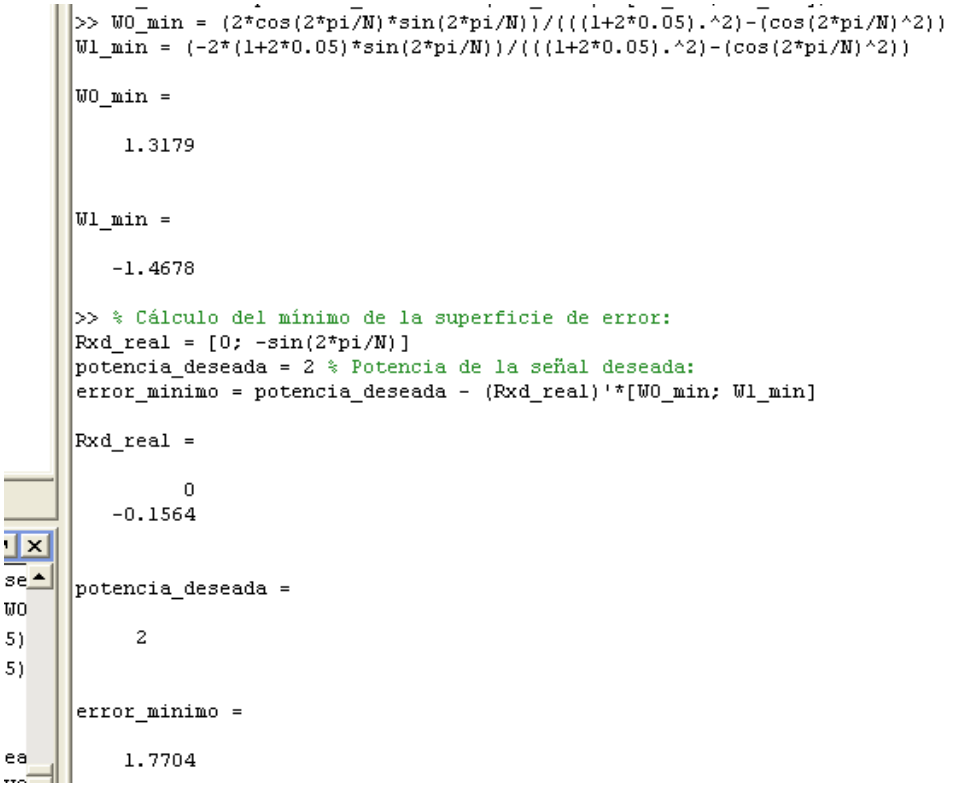
$$w^* = R_{XX}^{-1} \cdot R_{xd} = \begin{bmatrix} \dfrac{2 \cdot \cos\left[\frac{2\pi}{N}\right] \cdot \operatorname{sen}\left[\frac{2\pi}{N}\right]}{\left(1 + E\left[(n(k^2))\right]^2 - \cos^2\left[\frac{2\pi}{N}\right]\right.} \\ \dfrac{-2 \cdot \left(1 + 2.E\left[(n(k^2))\right]\right) \cdot \operatorname{sen}\left[\frac{2\pi}{N}\right]}{\left(1 + E\left[(n(k^2))\right]^2 - \cos^2\left[\frac{2\pi}{N}\right]\right.} \end{bmatrix} \qquad (4.35)$$

Y que el error cuadrático medio mínimo viene dado por 4.36:

$$\xi_{min} = \xi_{min} = E[d_k^2] - W^{*T} R_{xx} W^* - 2R_{xd}^T W^* = E[d[k]^2] - R_{xd}^T \cdot W^* \qquad (4.36)$$

```
% Cálculo del vector de pesos óptimo.
W0_min = (2*cos(2*pi/N)*sin(2*pi/N))/(((1+2*0.05).^2)-(cos(2*pi/N)^2));
W1_min = (-2*(1+2*0.05)*sin(2*pi/N))/(((1+2*0.05).^2)-(cos(2*pi/N)^2));
% Cálculo del mínimo de la superficie de error:
Rxd_real = [0; -sin(2*pi/N)];
potencia_deseada = 2; % Potencia de la señal deseada:
error_minimo = potencia_deseada - (Rxd_real)'*[W0_min; W1_min];
```

Pantalla 4.a Obtención de los pesos optimos wo_min y w1_min



```
>> W0_min = (2*cos(2*pi/N)*sin(2*pi/N))/(((1+2*0.05).^2)-(cos(2*pi/N)^2))
Wl_min = (-2*(1+2*0.05)*sin(2*pi/N))/(((1+2*0.05).^2)-(cos(2*pi/N)^2))

W0_min =

    1.3179


Wl_min =

   -1.4678

>> % Cálculo del mínimo de la superficie de error:
Rxd_real = [0; -sin(2*pi/N)]
potencia_deseada = 2 % Potencia de la señal deseada:
error_minimo = potencia_deseada - (Rxd_real)'*[W0_min; Wl_min]

Rxd_real =

         0
   -0.1564


potencia_deseada =

     2


error_minimo =

    1.7704
```

Representaremos el vector de pesos óptimos con la función contour.

figure(2);contour(W0_min, W1_min,error_minimo)

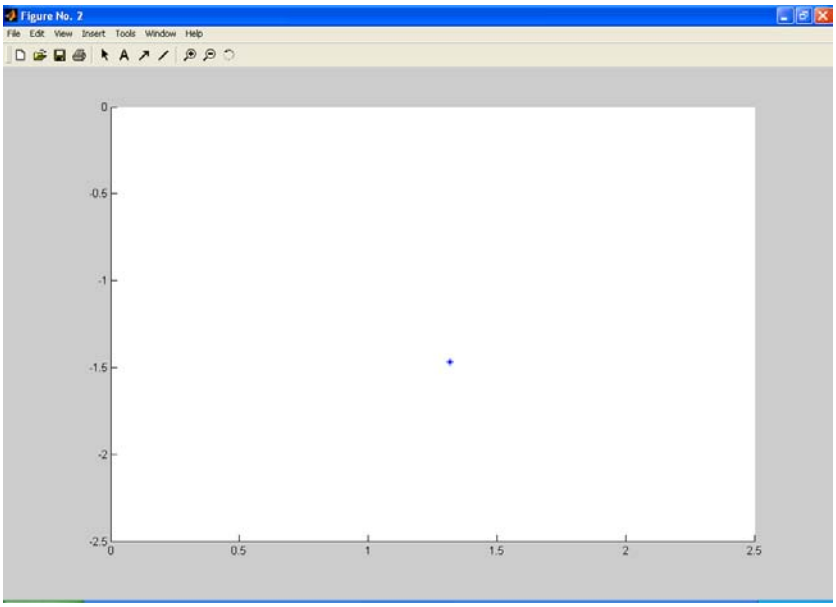


Figura 4.6 Peso optimo con la funcion contour

Como sabemos por teoría por lo anteriormente estudiado el método de máxima pendiente cuya regla de actualización de los coeficientes del filtro es la siguiente:

$$\mathbf{W}_{k+1} = \mathbf{W}_k - \alpha \nabla \xi(\mathbf{W}_k) = \mathbf{W}_k - \alpha(2\mathbf{R}_{XX}\mathbf{W}_k - 2\mathbf{R}_{Xd}) \quad (4.37)$$

En el estudio del método se asumió que se disponía de una medida exacta del vector gradiente en cada iteración. En la mayoría de las aplicaciones

no se dispone de una medida exacta y debe utilizarse una estimación basada en una serie de observaciones.

4.1.3 Variaciones con el algoritmo RLS

Analizaremos varias variaciones con el algoritmo RLS. Una de ellas es la aproximación al algoritmo mínimo cuadrado, otra de ellas es la Variación utilizando el principio de ortogonalidad y el filtro de Kalman.

Aproximación al algoritmo mínimo cuadrado.

Utilizando la aproximación al mínimo cuadrado (LS) nosotros podemos minimizar la diferencia cuadrática entre los datos d(n) y la señal de salida (output signal) de un sistema lineal LTI. La señal y (n) es generada sobre el sistema, dependiendo de los parámetros desconocidos w"s. el LSE error lineal del algoritmo mínimo cuadrado son conocidas como valores de dato. Para definir el valor de LSE en función de J(w). w valor de los pesos. Nos queda el modelo de coeficientes ecuación 4.38

$$J(w) = \sum_{n=1}^{N} (d(n)-y(n))^2 \qquad (4.38)$$

Y la dependencia de J en w es a través de la ecuación y(n). Este toma valores de w minimizados de la función de costo J(w). esta tiene un valor de error, que fácilmente al hacerse recursivo la relaciona con los algoritmos RLS, hay que tener presente si se presentan señales de ruido que afecten la señal, para lo cual necesitaremos simular el sistema de error. A continuación veremos un ejemplo de determinación del error. Decidimos colocar este ejemplo para que se vea de donde se origina el error para esta función.

Ejemplo 4.1 Sea la señal y(n)=a * (cos(wo(n))) donde wo es conocido como la amplitud de determinación. Halle el error de la función de costo J(a)

$$J(a) = \sum_{n=1}^{N} (d(n) - a \cdot \cos w_0 n)^n \qquad (4.39)$$

De hay nosotros obtenemos la derivada con respecto a a

$$\frac{\partial J(a)}{\partial a} = \sum_{n=1}^{N} (-)2\cos w_o n(d(n) - a\cos w_o n) = 0 \qquad (4.40)$$

$$\hat{a} = \frac{\sum_{n=1}^{N} d(n)\cdot\cos w_o n}{\sum_{n=1}^{N} \cos w_o n} \qquad (4.41)$$

Luego asumimos valores que los valores a la salida son del tipo lineal, los cuales tienen una relación de y(n)= x(n)w, donde x(n) es conocido como los valores de secuencia. Con estos valores podemos obtener el valor del error tanto en sistemas lineales LSE como en sistemas lineales recursivos RLS. Este criterio está representado en J(w), ecuación 4.42

$$J(w) = \sum_{n=1}^{N} (d(n) - x(n)w)^2 \qquad (4.42)$$

Este valor estimado de w es:

$$w = \frac{\sum_{n=1}^{N} d(n)x(n)}{\sum_{n=1}^{N} x^2(n)} \tag{4.43}$$

De aquí podemos determinar el momento mínimo de error LS el cual lo determinaremos en el problema 4.1, la fórmula del momento de error J es la 4.44

$$Jmin = J(\hat{w}) = \sum_{n=1}^{N} d^2(n) - \hat{w} \sum_{n=1}^{N} d(n)x(n) = \sum_{n=1}^{N} d^2(n) - \frac{\left[\sum_{n=1}^{N} d(n)x(n)\right]^2}{\sum_{n=1}^{N} x^2(n)} \tag{4.44}$$

A continuación veremos el principio de ortogonalidad (es decir que están a 90 grados) de los algoritmos LS y RLS.

Variación utilizando el principio de ortogonalidad y el filtro de wiener.

El principio de ortogonalidad es un principio que poseen los algoritmos mínimo cuadrado y mínimo cuadrado recursivo, en el se suele utilizar el filtro de wiener para resolver los problemas de LS y se suele utilizar el filtro de kalman para resolver los problemas en RLS. Para la determinación de los pesos se utilizan derivadas de los pesos, w1,w2,w3,w4,…. Para dwk. De aquí determinamos el error.

$$\frac{\partial J(w1,w2,.....,w_M)}{\partial Wk} = \frac{\partial}{\partial_{Wk}}\left[\sum_{m=1}^{N} e(m)e(m)\right] = 2\sum_{m=1}^{N} e(m)\frac{\partial e(m)}{\partial_{wk}} \quad k= 1,2,....,M \qquad (4.45)$$

Utilizando la ecuación 4.45 determinaremos el valor de w y M de los coeficientes. En e(m) ecuación 4.46

$$e(m) = d(m) - \sum_{k=1}^{M} w_k x_k(m) \qquad (4.46)$$

Sacando la derivada de e(m) con respecto a wk y introduciendo los resultados a la ecuación 4.47 nosotros obtenemos:

$$\frac{\partial J}{\partial wk} = -2 \sum_{m=1}^{N} e(m) x_k(m) \qquad (4.47)$$

Nosotros observamos que cuando w= $\hat{w}$(el valor optimo) nosotros tenemos una relación de derivada

$\frac{\partial J}{\partial w_k} = 0$ for $k = 1, 2, \ldots, M$, y evaluado en 4.47 obtenemos la ecuación 4.48

$$\sum_{m=1}^{N} \hat{e}(m) x_k(m) = \hat{e}\, x_k \qquad k=1,2,3,.....,M \qquad (4.48)$$

Donde

$$\hat{e} = [\,\hat{e}(1) \quad \hat{e}(2) \quad \hat{e}(3) \ldots\ldots \hat{e}(N)]^T = d - \hat{y} \tag{4.49}$$

$$x = [x_k(1) \quad x_k(2) \quad \ldots\ldots\ldots\ldots \quad x_k(N)]^T \qquad k=1,2,3,\ldots,M \tag{4.50}$$

Podemos notar que cuando $\mathbf{w} = \hat{\mathbf{w}}$ (el valor optimo) nosotros tenemos la relación $\frac{\partial J}{\partial w_k} = 0$ for $k = 1, 2, \ldots, M$,

El error estimado $\hat{e}(m)$ es un valor optimo en el mínimo cuadrado, todo esto resulta de el uso del principio de ortogonalidad.

El presente análisis indica que los coeficientes de los filtros son óptimos al utilizar mínimos cuadrados, cuando a la salida del filtro podemos determinar el error y la ortogonalidad en otras palabras el error es ortogonal.

Muchos consideran al algoritmo RLS como un caso especial del filtro de kalman.

4.2 FILTRO DE KALMAN

El filtro de Kalman es uno de los principales filtros utilizados sobre todo para depuración de señales, permitir el paso de señales deseadas y eliminación del ruido, es ampliamente utilizado en los algoritmos RLS.

La deducción del filtro de Kalman puede, o debe, hacerse en términos del estimador óptimo de media condicional basado en el modelo de Gauss-Markov de un proceso.

El filtro de kalman utiliza cambios con respecto al filtro de wiener, el primer cambio que tiene lugar es que se debe de establece un modelo de señal sobre el que después se derivara el algoritmo. Este modelo, denominado de Gauss-Markov es válido para cualquier proceso gaussiano.

El filtro de Kalman, es un observador de estados que es capaz de filtrar las señales de ruido presentes en un sistema, esto lo hace una herramienta muy poderosa cuando se desea controlar sistemas con un alto contenido de ruido.

A continuación haremos un análisis matemático de las ecuaciones que intervienen en el filtro de kalman.

En primer lugar el filtro de kalman minimiza el error cuadrático, y se basa en el algoritmo de mínimos cuadrado LS. Ecuación 4.51

$$\sum_{k=1}^{M} e_k^2 = \sum_{k=1}^{M} (r_k - h^T x(k))^2 \qquad (4.51)$$

Las estimaciones optimas hp en función de LS son (ecuación 4.52)

$$hp = [\, Xp(k)^T Xp(k)]^{-1} Xp(k)^T rp \qquad (4.52)$$

De donde obtenemos las ecuaciones 4.53, 4.54 y 4.55

$$rp = (r_0, r_1, \ldots\ldots, r_{p-1})^T \qquad (4.53)$$

$$Xp(k) = (X^T(0), X^T(1), \ldots\ldots, X^T(p-1))^T \qquad (4.54)$$

$$hp = (h[0], h[1], \ldots\ldots, h[N-1])^T \qquad (4.55)$$

Y con estos valores obtenemos x transpuesta, ecuación 4.56

$$X^T(k) = (x[k], x[k-1], \ldots\ldots, x[k-(N-1)]) \ , \quad k=0,1,\ldots\ldots,p-1 \qquad (4.56)$$

Este desarrollo LS es muy costoso computacionalmente, el algoritmo rls con el filtro de kalman actualiza el valor de hp para cada muestra que llega al filtro, para ello pondera exponencialmente los datos para ir eliminando de forma gradual el efecto que tienen sobre h los datos más antiguos, dejando solamente las ultimas componentes de las ecuaciones 4.53 y la 4.54, y las ultimas columnas de 4.55, esto le permitirá seguir pequeñas variaciones de la señal, los cuales nos permitirán formar una etapa de predicción (ecuaciones 4.57 y 4.58) y una de corrección (ecuaciones 4.59, 4.60 y 4.61).

$$h(k) = h(k-1) + g(k) \qquad (4.57)$$

Predicción

$$P(k) = \frac{1}{\gamma}[P(k-1) - g(k)x^T(k)P(k-1)] \qquad (4.58)$$

$$g(k) = \frac{P(k-1)x(k)}{\alpha_k} \quad (4.59)$$

$$e_k = r_k - X^T(k)h(k-1) \quad \text{Corrección} \quad (4.60)$$

$$\alpha_k = \gamma + x^T(k)P(k-1)x(k) \quad (4.61)$$

En la siguiente figura 4.7 se ve el nombre de cada uno de los parámetros del filtro de kalman.

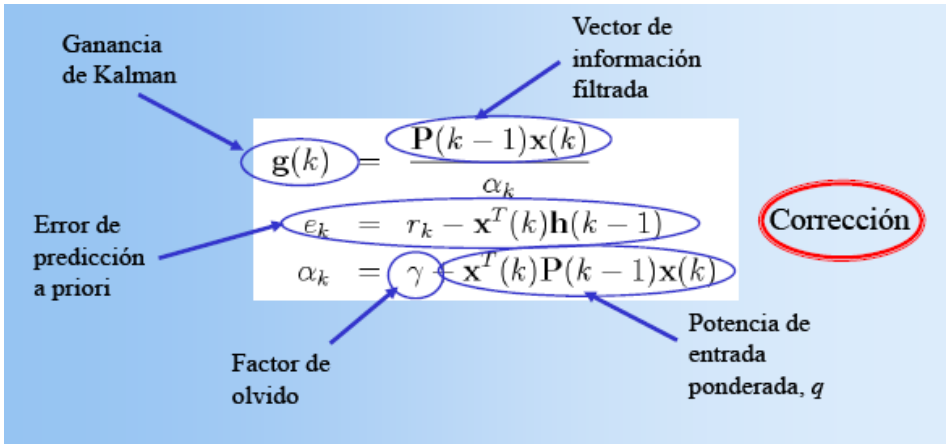


Fig 4.7 Filtro de Kalman Parametros

Los pasos para resolver el filtro de kalman lo vemos en la tabla XXXVII

PASOS:

1. Calcular la salida a priori: $\mathbf{x}^T(k)\mathbf{h}(k-1)$, y el error a priori.
2. Calcular el vector de información filtrada: $\mathbf{P}(k-1)\mathbf{x}(k)$
3. Calcular la potencia de error normalizada: $q = \mathbf{x}(k)^T\mathbf{P}(k-1)\mathbf{x}(k)$
4. Calcular la ganancia: $1/(\gamma + q)$
5. Calcular el vector de información filtrada normalizado: $\mathbf{g}(k)$
6. Actualizar el vector de pesos del filtro: **h**.
7. Actualizar la matriz de covarianzas: **P**.

Tabla XXXVII Pasos para resolver el filtro de kalman

Si se desea profundizar más sobre el filtro de Kalman y como se desarrolla una aplicación revise el anexo G

4.3 Pantallas de visualización del algoritmo adaptativo mínimo cuadrado recursivo RLS

Al hacer correr cualquiera de las aplicaciones en matlab del algoritmo adaptativo mínimo cuadrado recursivo rls que hemos realizado, se presentaran algunas pantallas de visualización, algunas de estas pantallas son para información de los usuarios a usar el programa, otras son para ingresar valores de entrada, y las

ultimas son las generadas por el programa matlab son pantallas de salida, con estos datos de entrada y salida formaremos tablas de datos y resultados obtenidos. Todas estas pantallas servirán tanto para señales senosoidales como para señales gausianas. También podrían servir para otros tipos de señales que cumplan las características señaladas por kalman para filtros adaptativos.

Las pantallas a utilizar en el programa del algoritmo adaptativo rls son:

Pantalla de presentación:

En esta pantalla se hace una explicación al usuario de las características de la aplicación y lo que resuelve, está el nombre de la universidad, los nombres de los programadores del programa, el profesor guía, el tipo de señal a usar en el programa, , una pequeña explicación de los datos de entrada y los de salida.

Pantalla de ingreso de datos:

En esta pantalla ingresamos el numero de antenas a utilizar (en el algoritmo rls de 1 a 15). los valores iníciales de las antenas, los valores de ruido de las antenas. En esta pantalla aparecerá una opción llamada opción ruido que si es igual a 1 el ruido se ingresara a la entrada del sistema (antes del filtro), si se escoge 2 el ruido se agregara a la salida del sistema. También se ingresara un valor de error del sistema

Pantalla de potencias obtenidas:

Esta pantalla nos dará la potencia del sistema y el valor de la potencia de cada una de las antenas.

Pantalla de comparación de pesos estimados:

En esta pantalla presentaremos los datos iníciales ingresados por teclado (los llamaremos también valores actuales) los representaremos por medio de una cruz negra, y los compararemos con los valores obtenidos por el algoritmo rls, este valor obtenido lo llamaremos valores estimados, y lo representaremos con un color rojo.

Pantalla de valor de la curva de error:

En esta pantalla se muestra el comportamiento de la ecuación de error del algoritmo rls, esta nos servirá para compararla con el comportamiento de otros algoritmos adaptativos.

Pantalla de comparación entre la salida del sistema con la señal de referencia:

Esta pantalla es una de las más importantes, aparecerá la señal que obtenemos a la salida del sistema (color rojo) , y se la comparara con la señal de referencia (color azul).

En el anexo F se muestran las pantallas de visualización que obtendremos en nuestras implementaciones al utilizar diferentes valores de entrada y diferentes casos en cada uno de los algoritmos utilizados.

4.4 Programa en matlab del algoritmo adaptativo mínimo cuadrado Recursivo RLS

Al hacer el programa en matlab del algoritmo RLS decidimos utilizar el mismo procedimiento que utilizamos en el algoritmo LMS para lo cual utilizamos una función principal que sería llamada desde el programa matlab, esto permite ciertas ventajas ya que todo el programa está en una sola función, para evitar que se haga muy complejo el programa decidimos colocar la mayor cantidad posible de comentarios, de lo que se hace en cada parte del programa.

El programa en matlab del algoritmo RLS. Debera recibir los siguientes argumentos de entrada:

- la señal aplicada a la entrada del filtro, Xn=2*sin((2*pi*k)/N)

- el numero de antenas n

- la señal de referencia, d(k)=2*cos((2*pi*k)/N);

- el ruido de las diferentes antenas, (hay que tener presente que el algoritmo LMS en su estructura internamente produce un ruido del 10%), aparte de este ruido puede haber otros. n(f)

- el orden del filtro FIR cuyos coeficientes se van a ajustar, orden

- el número de iteraciones en el aprendizaje, N (este número se hará hasta que cumpla el error)

- la constante de ajuste mu

- el vector de pesos inicial. Los voltajes de las antenas V(f)

- el error deseado (puede ser ingresado por teclado) error

Si se conociera la frecuencia se podría usar como entrada Xn=2*cos(2*pi*f*t) y tener una ecuación en función del tiempo y la frecuencia.

A su salida devolverá:

- la salida en cada iteración durante el proceso de ajuste, y(t)

- la curva de aprendizaje (grafico de la curva)

- los vectores de pesos obtenidos durante el proceso. W(t)

- la Potencia de la antena P(k)

- el numero de iteraciones en que converge la curva y se cumple el valor del error. z

Programa Principal y función principal

```
function[caratula]=problema3_rls;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
% Algoritmo RLS
% Autores: Juan Alvarez , Maribel Chuez
%
%Escuela Superior Politecnica del Litoral
%Facultad de Electricidad y Computacion
%Guayaquil, Ecuador
% email :juan_alvarez8@latinmail.com
% email : mchuez@ceibo.espol.edu.ec
% Webpage : proximamente
% Date    : Viernes 30 de abril del 2009
% Version : 1.0.1
% Referencias : S. Haykin, Adaptive Filter Theory. 3rd edition, Upper
Saddle River, NJ: Prentice-Hall, 1996.
%
%clear all % esta funcion borra los valores anteriores que esten
memoria %
%close all % esta funcion borra cualquier ventana anterior activa %
%hold off   % esta funcion encera cualquier grafico que estuviera
activo %
 fprintf('*****************************************************.\n')
fprintf('*    ESCUELA SUPERIOR POLITECNICA DEL LITORAL        *.\n')
fprintf('*                                                    *.\n')
fprintf('* PROBLEMA3_rls.- Determinar los pesos, el numero de *.\n')
fprintf('* iteraciones y la potencia de un grupo de antenas   *.\n')
fprintf('* inteligentes utilizando el algoritmo RLS (algoritmo*.\n')
fprintf('* minimo cuadrado recursivo, las antenas estan distri*.\n')
fprintf('* buidas en forma de celdas                          *.\n')
fprintf('* para lo cual requeriremos de los siguientes datos: *.\n')
fprintf('*                                                    *.\n')
fprintf('*  • la señal aplicada a la entrada del filtro,      *.\n')
fprintf('*   ej  Xn=sin((2*pi*k)/N)                           *.\n')
fprintf('*  • el numero de antenas  x                         *.\n')
fprintf('*  • la señal de referencia, d(k)=cos((2*pi*k)/N);   *.\n')
fprintf('*  • el ruido de las diferentes antenas, (hay que    *.\n')
fprintf('*  tener presente que el algoritmo RLS en su         *.\n')
fprintf('*  estructura internamente produce un ruido que puede*.\n')
fprintf('*  ser de 2, 3 y hasta 10 %  aparte de este ruido    *.\n')
fprintf('*  generado por el mismo programa RLS se puede       *.\n')
fprintf('*  ingresar ruidos adicionales en el programa a la   *.\n')
fprintf('*  entrada o a la salida del programa.  n(f)         *.\n')
fprintf('*  • el orden del filtro FIR cuyos coeficientes se   *.\n')
```

```
fprintf('*  van a ajustar, orden                                    *.\n')
fprintf('*  • el número de iteraciones en el aprendizaje, N        *.\n')
fprintf('*  (este número se hará  hasta que cumpla el error        *.\n')
fprintf('*  • la constante de ajuste   u                           *.\n')
fprintf('*  • el vector de pesos inicial. Los voltajes de las      *.\n')
fprintf('*  antenas V(f)                                           *.\n')
fprintf('*  • el error deseado puede ser ingresado por teclado     *.\n')
fprintf('*    error                                                *.\n')
fprintf('*                                                         *.\n')
fprintf('*  A su salida devolverá:                                 *.\n')
fprintf('*  • la salida en cada iteración durante el proceso       *.\n')
fprintf('*  de ajuste, y(t)                                        *.\n')
fprintf('*  • la curva de aprendizaje (grafico de la curva)        *.\n')
fprintf('*  • los vectores de pesos obtenidos durante el pro-      *.\n')
fprintf('*  ceso. W(t)                                             *.\n')
fprintf('*  • la Potencia de la antena P(k)                        *.\n')
fprintf('*  • el numero de iteraciones en que converge la          *.\n')
fprintf('*    curva y se cumple el valor del error                 *.\n')
fprintf('*                                                         *.\n')
fprintf('* alumno: Juan Francisco Alvarez Alvarado                 *.\n')
fprintf('* alumno: Maribel del Rosario Chuez Gonzales              *.\n')
fprintf('*                                                         *.\n')
fprintf('* Profesor :Ing. Pedro Vargas                             *.\n')
fprintf('*                                                         *.\n')
fprintf('***********************************************************.\n')
t=input('teclee enter para proseguir');
 clc
% ingresamos el orden del filtro, cuyo coeficiente se va ha ajustar%
orden=input(' ingrese el orden del filtro, cuyo coeficiente se va ha
ajustar....')

% ingresamos el numero de antenas del sistema
disp(' ingrese el numero de antenas entre 1 y 15');
x=input(' ingrese el numero de antenas entre
1,2,3,4,5,6,7,8,9,10,11,12,13,14 o 15....');
if x==1
    D(1)=input('ingrese el voltaje1 (en V)..... voltaje1=');
end

if x==2
    D(1)=input('ingrese el voltaje1 (en V)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
end

if x==3
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
end
```

```
if x==4
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
end
if x==5
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
end
if x==6
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
end
if x==7
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
    D(7)=input(' ingrese la voltaje7 (en v)..... voltaje7=');
end
if x==8
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
    D(7)=input(' ingrese la voltaje7 (en v)..... voltaje7=');
    D(8)=input(' ingrese la voltaje8 (en v)..... voltaje8=');
end
if x==9
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
    D(7)=input(' ingrese la voltaje7 (en v)..... voltaje7=');
```

```
    D(8)=input(' ingrese la voltaje8 (en v)..... voltaje8=');
    D(9)=input(' ingrese la voltaje9 (en v)..... voltaje9=');
end
if x==10
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
    D(7)=input(' ingrese la voltaje7 (en v)..... voltaje7=');
    D(8)=input(' ingrese la voltaje8 (en v)..... voltaje8=');
    D(9)=input(' ingrese la voltaje9 (en v)..... voltaje9=');
    D(10)=input(' ingrese la voltaje10 (en v)..... voltaje10=');
end
if x==11
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
    D(7)=input(' ingrese la voltaje7 (en v)..... voltaje7=');
    D(8)=input(' ingrese la voltaje8 (en v)..... voltaje8=');
    D(9)=input(' ingrese la voltaje9 (en v)..... voltaje9=');
    D(10)=input(' ingrese la voltaje10 (en v)..... voltaje10=');
    D(11)=input(' ingrese la voltaje11 (en v)..... voltaje11=');
end
if x==12
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
    D(7)=input(' ingrese la voltaje7 (en v)..... voltaje7=');
    D(8)=input(' ingrese la voltaje8 (en v)..... voltaje8=');
    D(9)=input(' ingrese la voltaje9 (en v)..... voltaje9=');
    D(10)=input(' ingrese la voltaje10 (en v)..... voltaje10=');
    D(11)=input(' ingrese la voltaje11 (en v)..... voltaje11=');
    D(12)=input(' ingrese la voltaje12 (en v)..... voltaje12=');
end
if x==13
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
```

```
    D(7)=input(' ingrese la voltaje7 (en v)..... voltaje7=');
    D(8)=input(' ingrese la voltaje8 (en v)..... voltaje8=');
    D(9)=input(' ingrese la voltaje9 (en v)..... voltaje9=');
    D(10)=input(' ingrese la voltaje10 (en v)..... voltaje10=');
    D(11)=input(' ingrese la voltaje11 (en v)..... voltaje11=');
    D(12)=input(' ingrese la voltaje12 (en v)..... voltaje12=');
    D(13)=input(' ingrese la voltaje13 (en v)..... voltaje13=');
end
if x==14
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
    D(7)=input(' ingrese la voltaje7 (en v)..... voltaje7=');
    D(8)=input(' ingrese la voltaje8 (en v)..... voltaje8=');
    D(9)=input(' ingrese la voltaje9 (en v)..... voltaje9=');
    D(10)=input(' ingrese la voltaje10 (en v)..... voltaje10=');
    D(11)=input(' ingrese la voltaje11 (en v)..... voltaje11=');
    D(12)=input(' ingrese la voltaje12 (en v)..... voltaje12=');
    D(13)=input(' ingrese la voltaje13 (en v)..... voltaje13=');
    D(14)=input(' ingrese la voltaje14 (en v)..... voltaje14=');
end
if x==15
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
    D(7)=input(' ingrese la voltaje7 (en v)..... voltaje7=');
    D(8)=input(' ingrese la voltaje8 (en v)..... voltaje8=');
    D(9)=input(' ingrese la voltaje9 (en v)..... voltaje9=');
    D(10)=input(' ingrese la voltaje10 (en v)..... voltaje10=');
    D(11)=input(' ingrese la voltaje11 (en v)..... voltaje11=');
    D(12)=input(' ingrese la voltaje12 (en v)..... voltaje12=');
    D(13)=input(' ingrese la voltaje13 (en v)..... voltaje13=');
    D(14)=input(' ingrese la voltaje14 (en v)..... voltaje14=');
    D(15)=input(' ingrese la voltaje15 (en v)..... voltaje15=');
end

% ingresamos el ruido que presenta cada antena del sistema
% Si existe ruido adicional al ocasionado por el sistema
if x==1
    E(1)=input(' ingrese el ruido en la antena1.....ruido1=');
    ruido=E(1);
end
```

```
if x==2
    E(1)=input('ingrese el ruido en la antena1.....ruido1==');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    ruido=(E(1)+E(2))/2;
end

if x==3
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    ruido=(E(1)+E(2)+E(3))/3;
end

if x==4
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    ruido=(E(1)+E(2)+E(3)+E(4))/4;
end
if x==5
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    ruido=(E(1)+E(2)+E(3)+E(4)+E(5))/5;
end
if x==6
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6))/6;
end
if x==7
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    E(7)=input('ingrese el ruido en la antena7.....ruido7=');
    ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6)+E(7))/7;
end
if x==8
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
```

```
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    E(7)=input('ingrese el ruido en la antena7.....ruido7=');
    E(8)=input('ingrese el ruido en la antena8.....ruido8=');
    ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6)+E(7)+E(8))/8;
end
if x==9
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    E(7)=input('ingrese el ruido en la antena7.....ruido7=');
    E(8)=input('ingrese el ruido en la antena8.....ruido8=');
    E(9)=input('ingrese el ruido en la antena9.....ruido9=');
    ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6)+E(7)+E(8)+E(9))/9;
end
if x==10
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    E(7)=input('ingrese el ruido en la antena7.....ruido7=');
    E(8)=input('ingrese el ruido en la antena8.....ruido8=');
    E(9)=input('ingrese el ruido en la antena9.....ruido9=');
    E(10)=input('ingrese el ruido en la antena10.....ruido10=');
    ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6)+E(7)+E(8)+E(9)+E(10))/10;
end
if x==11
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    E(7)=input('ingrese el ruido en la antena7.....ruido7=');
    E(8)=input('ingrese el ruido en la antena8.....ruido8=');
    E(9)=input('ingrese el ruido en la antena9.....ruido9=');
    E(10)=input('ingrese el ruido en la antena10.....ruido10=');
    E(11)=input('ingrese el ruido en la antena11.....ruido11=');

ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6)+E(7)+E(8)+E(9)+E(10)+E(11))/11;
end
```

```
if x==12
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    E(7)=input('ingrese el ruido en la antena7.....ruido7=');
    E(8)=input('ingrese el ruido en la antena8.....ruido8=');
    E(9)=input('ingrese el ruido en la antena9.....ruido9=');
    E(10)=input('ingrese el ruido en la antena10.....ruido10=');
    E(11)=input('ingrese el ruido en la antena11.....ruido11=');
    E(12)=input('ingrese el ruido en la antena12.....ruido12=');

ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6)+E(7)+E(8)+E(9)+E(10)+E(11)+E(12))/
12;
end
if x==13
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    E(7)=input('ingrese el ruido en la antena7.....ruido7=');
    E(8)=input('ingrese el ruido en la antena8.....ruido8=');
    E(9)=input('ingrese el ruido en la antena9.....ruido9=');
    E(10)=input('ingrese el ruido en la antena10.....ruido10=');
    E(11)=input('ingrese el ruido en la antena11.....ruido11=');
    E(12)=input('ingrese el ruido en la antena12.....ruido12=');
    E(13)=input('ingrese el ruido en la antena13.....ruido13=');

ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6)+E(7)+E(8)+E(9)+E(10)+E(11)+E(12)+E
(13))/13;
end
if x==14
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    E(7)=input('ingrese el ruido en la antena7.....ruido7=');
    E(8)=input('ingrese el ruido en la antena8.....ruido8=');
    E(9)=input('ingrese el ruido en la antena9.....ruido9=');
    E(10)=input('ingrese el ruido en la antena10.....ruido10=');
    E(11)=input('ingrese el ruido en la antena11.....ruido11=');
    E(12)=input('ingrese el ruido en la antena12.....ruido12=');
    E(13)=input('ingrese el ruido en la antena13.....ruido13=');
```

```
    E(14)=input('ingrese el ruido en la antena14.....ruido14=');

ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6)+E(7)+E(8)+E(9)+E(10)+E(11)+E(12)+E
(13)+E(14))/14;
end
if x==15
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    E(7)=input('ingrese el ruido en la antena7.....ruido7=');
    E(8)=input('ingrese el ruido en la antena8.....ruido8=');
    E(9)=input('ingrese el ruido en la antena9.....ruido9=');
    E(10)=input('ingrese el ruido en la antena10.....ruido10=');
    E(11)=input('ingrese el ruido en la antena11.....ruido11=');
    E(12)=input('ingrese el ruido en la antena12.....ruido12=');
    E(13)=input('ingrese el ruido en la antena13.....ruido13=');
    E(14)=input('ingrese el ruido en la antena14.....ruido14=');
    E(15)=input('ingrese el ruido en la antena15.....ruido15=');

ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6)+E(7)+E(8)+E(9)+E(10)+E(11)+E(12)+E
(13)+E(14)+E(15))/15;
end

opcion_ruido=2;
disp('si desea que el ruido adicional vaya a la señal a la entrada
digite 1');
disp(' de lo contrario se considerara que el ruido adicional se
agregara a la señal de referencia');
opcion_ruido=input('opcion_ruido=');
error=input('ingrese el error del sistema=');
% numero de iteraciones del proceso hasta que se cumpla el error
opcion_senal=2;
disp('si desea que la señal de entrada se use como señal de referencia
digite 1');
disp('de lo contrario se considerara otra señal diferente como señal de
referencia');
opcion_senal=input('opcion_senal=');
N=2000;
j=0;
% valores recomendado por la central matlab para los pesos en una
funcion
%
% h=
% vamos a considerar que todas las antenas tienen la misma señal a la
% entrada del filtro por ejemplo  entrada=sin((2*pi*k)/N);
%
```

```
if x==1
    h=[D(1)]
    p=[D(1)]
end
if x==2
    h=[D(1);D(2)]
    p=[(D(1)+D(2))/2]+[0.8*(D(1)+D(2))/2];
end
if x==3
    h=[D(1);D(2);D(3)]
    p=[(D(1)+D(2)+D(3))/3]+[0.8*(D(1)+D(2)+D(3))/3];
end
if x==4
    h=[D(1);D(2);D(3);D(4)]
    p=[(D(1)+D(2)+D(3)+D(4))/4]+[0.8*(D(1)+D(2)+D(3)+D(4))/4];
end
if x==5
    h=[D(1);D(2);D(3);D(4);D(5)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5))/5]+[0.8*(D(1)+D(2)+D(3)+D(4)+D(5))/5];
end
if x==6
    h=[D(1);D(2);D(3);D(4);D(5);D(6)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6))/6]+[0.8*(D(1)+D(2)+D(3)+D(4)+D(5)+D(
6))/6];
end
if x==7
    h=[D(1);D(2);D(3);D(4);D(5);D(6);D(7)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7))/7]+[0.8*(D(1)+D(2)+D(3)+D(4)+D(
5)+D(6)+D(7))/7];
end
if x==8
    h=[D(1);D(2);D(3);D(4);D(5);D(6);D(7);D(8)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8))/8]+[0.8*(D(1)+D(2)+D(3)+D(
4)+D(5)+D(6)+D(7)+D(8))/8];
end
if x==9
    h=[D(1);D(2);D(3);D(4);D(5);D(6);D(7);D(8);D(9)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9))/9]+[0.8*(D(1)+D(2)+D(
3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9))/9];
end
if x==10
    h=[D(1);D(2);D(3);D(4);D(5);D(6);D(7);D(8);D(9);D(10)]
```

```
p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10))/10]+[0.8*(D(1)+
D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10))/10];
end
if x==11
    h=[D(1);D(2);D(3);D(4);D(5);D(6);D(7);D(8);D(9);D(10);D(11)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10)+D(11))/11]+[0.8*
(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10)+D(11))/11];
end
if x==12
    h=[D(1);D(2);D(3);D(4);D(5);D(6);D(7);D(8);D(9);D(10);D(11);D(12)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10)+D(11)+D(12))/12]
+[0.8*(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10)+D(11)+D(12))/
12];
end
if x==13

h=[D(1);D(2);D(3);D(4);D(5);D(6);D(7);D(8);D(9);D(10);D(11);D(12);D(13)
]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10)+D(11)+D(12)+D(13
))/13]+[0.8*(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10)+D(11)+D
(12)+D(13))/13];
end
if x==14

h=[D(1);D(2);D(3);D(4);D(5);D(6);D(7);D(8);D(9);D(10);D(11);D(12);D(13)
;D(14)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10)+D(11)+D(12)+D(13
)+D(14))/14]+[0.8*(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10)+D
(11)+D(12)+D(13)+D(14))/14];
end
if x==15

h=[D(1);D(2);D(3);D(4);D(5);D(6);D(7);D(8);D(9);D(10);D(11);D(12);D(13)
;D(14);D(15)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10)+D(11)+D(12)+D(13
)+D(14)+D(15))/15]+[0.8*(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D
(10)+D(11)+D(12)+D(13)+D(14)+D(15))/15];
end
M=30;
for k=1:400
    entrada(k)=sin((2*pi*k)/M);
    %senal de entrada
end
```

```
for k=1:400
    senal_referencia(k)=cos((2*pi*k)/M);
    % en algunos libros la señal de referencia se la llama señal
deseada
end
entrada';
entrada=entrada';
if opcion_ruido==1
    entrada=entrada+ruido; % aqui agregamos el ruido adicional a la
entrada
end
n=sqrt(0.05)*randn(400,1); % ruido a la entrada del filtro, genera 400
valores
% aleatorios de muestra
[b,a] = butter(2,0.25); % esta funcion calcula los valores de a y b
Gz = tf(b,a,-1); % la funcion de transferencia tf funciona en matlab 7
% esta funcion permite resolver la transformada z.

%Si no se tiene acceso a la caja de herramientas del matlab se puede
cargar valores de
%la central matlab usando el sample data
%load IIRsampledata;
%entrada= IIRsampledata(1:2000);
%d= IIRsampledata(1:2000);

% se puede usar ldiv para hallar las aproximaciones de pesos para un
filtro IIR (filtro de respuesta infinita)
% tambien se puede evaluar  y=h*u
%ldiv es una funcion que nos permite hallar la inversa de la
transformada Z (Matlab central file exchange)
%para hallar los valores de orden de los pesos se puede utilizar la
%siguiente formula
%use h=ldiv(b,a,sysorder)'; ==> here we use sysorder == 10

h=h(1:orden);
y = lsim(Gz,entrada);% simula respuestas arbitrarias de entrada,
relacion la señal Gz
% con la entrada
% para agregar el ruido, ruido adicional que genera el algoritmo rls
n = n * std(y)/(10*std(n));
if opcion_senal==1
senal_referencia = y + n;
end
if opcion_senal>1
    senal_referencia';
    senal_referencia=senal_referencia';
    senal_referencia=senal_referencia+n;
    %
end
```

```
if opcion_ruido>1
    senal_referencia=senal_referencia+ruido; % agrega el ruido
adicional
    % a la salida del filtro
end
totallength=size(senal_referencia,1);% nos da el numero de valores de
la funcion
% de entrada
% conociendo los voltajes de las antenas podemos determinar la potencia
% en ellas
% potencia de la señal a la salida (potencia del sistema
potencia=mean(senal_referencia.^2)
if x==1
% tambien calcularemos la potencia para cada una de las antenas
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
end
if x==2
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia1=mean(d2.^2)
end
if x==3
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia1=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
end
if x==4
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia1=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
```

```
potencia4=mean(d4.^2)
end
if x==5
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
end
if x==6
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
end
if x==7
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
```

```
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
% antena7
d7=D(7)+n+E(7);
potencia7=mean(d7.^2)
end
if x==8
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
% antena7
d7=D(7)+n+E(7);
potencia7=mean(d7.^2)
% antena8
d8=D(8)+n+E(8);
potencia8=mean(d8.^2)
end
if x==9
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
```

```
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
% antena7
d7=D(7)+n+E(7);
potencia7=mean(d7.^2)
% antena8
d8=D(8)+n+E(8);
potencia8=mean(d8.^2)
% antena9
d9=D(9)+n+E(9);
potencia9=mean(d9.^2)
end
if x==10
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
% antena7
d7=D(7)+n+E(7);
potencia7=mean(d7.^2)
% antena8
d8=D(8)+n+E(8);
potencia8=mean(d8.^2)
% antena9
d9=D(9)+n+E(9);
potencia9=mean(d9.^2)
% antena10
d10=D(10)+n+E(10);
```

```
potencia9=mean(d10.^2)
end
if x==11
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
% antena7
d7=D(7)+n+E(7);
potencia7=mean(d7.^2)
% antena8
d8=D(8)+n+E(8);
potencia8=mean(d8.^2)
% antena9
d9=D(9)+n+E(9);
potencia9=mean(d9.^2)
% antena10
d10=D(10)+n+E(10);
potencia10=mean(d10.^2)
% antena11
d11=D(11)+n+E(11);
potencia11=mean(d11.^2)
end
if x==12
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
```

```
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
% antena7
d7=D(7)+n+E(7);
potencia7=mean(d7.^2)
% antena8
d8=D(8)+n+E(8);
potencia8=mean(d8.^2)
% antena9
d9=D(9)+n+E(9);
potencia9=mean(d9.^2)
% antena10
d10=D(10)+n+E(10);
potencia10=mean(d10.^2)
% antena11
d11=D(11)+n+E(11);
potencia11=mean(d11.^2)
% antena12
d12=D(12)+n+E(12);
potencia12=mean(d12.^2)
end
if x==13
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
% antena7
d7=D(7)+n+E(7);
potencia7=mean(d7.^2)
% antena8
```

```
d8=D(8)+n+E(8);
potencia8=mean(d8.^2)
% antena9
d9=D(9)+n+E(9);
potencia9=mean(d9.^2)
% antena10
d10=D(10)+n+E(10);
potencia10=mean(d10.^2)
% antena11
d11=D(11)+n+E(11);
potencia11=mean(d11.^2)
% antena12
d12=D(12)+n+E(12);
potencia12=mean(d12.^2)
% antena13
d13=D(13)+n+E(13);
potencia13=mean(d13.^2)
end
if x==14
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
% antena7
d7=D(7)+n+E(7);
potencia7=mean(d7.^2)
% antena8
d8=D(8)+n+E(8);
potencia8=mean(d8.^2)
% antena9
d9=D(9)+n+E(9);
potencia9=mean(d9.^2)
% antena10
d10=D(10)+n+E(10);
potencia10=mean(d10.^2)
```

```
% antena11
d11=D(11)+n+E(11);
potencia11=mean(d11.^2)
% antena12
d12=D(12)+n+E(12);
potencia12=mean(d12.^2)
% antena13
d13=D(13)+n+E(13);
potencia13=mean(d13.^2)
% antena14
d14=D(14)+n+E(14);
potencia14=mean(d14.^2)
end
if x==15
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
% antena7
d7=D(7)+n+E(7);
potencia7=mean(d7.^2)
% antena8
d8=D(8)+n+E(8);
potencia8=mean(d8.^2)
% antena9
d9=D(9)+n+E(9);
potencia9=mean(d9.^2)
% antena10
d10=D(10)+n+E(10);
potencia10=mean(d10.^2)
% antena11
d11=D(11)+n+E(11);
potencia11=mean(d11.^2)
% antena12
d12=D(12)+n+E(12);
```

```
potencia12=mean(d12.^2)
% antena13
d13=D(13)+n+E(13);
potencia13=mean(d13.^2)
% antena14
d14=D(14)+n+E(14);
potencia14=mean(d14.^2)
% antena15
d15=D(15)+n+E(15);
potencia15=mean(d15.^2)
end
% calculo de la cota de la contante de ajuste
% cota=1/((orden+1)*potencia); el valor de cota se usa en el lms
%evalua 70 puntos por corrida ( N - orden 70 = 80 - 10 )
N=80;   % N=80
%begin of the algorithm
%forgetting factor valores recomendados por la biblioteca matlab
lamda = 0.999994115 ;% lamda= 0.9995
%initial P matrix
delta=100000000000000 % delta = 1e10 ;
P = delta * eye (orden);
w = zeros ( orden  , 1 ) ;
for n = orden : N
    u = entrada(n:-1:n-orden+1);
    phi = u' * P ;
    k = phi'/(lamda + phi * u );
    y(n)=w' * u;
    e(n) = senal_referencia(n) - y(n) ;
    w = (w) + k * e(n) ;
    % w=w/10;
    P = ( P - k * phi ) / lamda;
    % ajuste del ploteo, pesos
    Recordedw(1:orden,n)=w;
end
%check of results
for n =  N+1 : totallength
    u = entrada(n:-1:n-orden+1) ;
    y(n) = w' * u ;
    e(n) = senal_referencia(n) - y(n);
end
hold on
plot(senal_referencia)
plot(y,'r');
title('salida del sistema ') ;
xlabel('numero de muestras')
ylabel('valor estimado a la salida')
figure
semilogy((abs(e))) ;
title('curva de error') ;
```

```
xlabel('numero de muestras');
ylabel('valor del error');
figure
plot(Recordedw(1:orden,orden:N)');
title('convergencia de los pesos estimados') ;
xlabel('numero de muestras');
ylabel('valor de los pesos');
axis([1 N-orden min(min(Recordedw(1:orden,orden:N)'))
max(max(Recordedw(1:orden,orden:N)')) ]);
hold off
figure
plot(h, 'k+')
hold on
plot(w/10, 'r*')
legend('pesos actuales','pesos estimados');
title('Comparacion de los pesos actuales con los pesos estimados') ;
```

4.5 Descripción del ejemplo a correr en el programa.

Ingresaremos 2 curvas, la una será la señal de entrada:

entrada(k) = sin((2*pi*k)/M) y la segunda será la señal de referencia: señal_referencia(k) = cos((2*pi*k)/M) ingresaremos el orden del filtro, y el numero de valores reales (numero de antenas), también ingresaremos la señal de transferencia del filtro, también ingresaremos el error deseado, el algoritmo lms y rls suelen tener un error menor de 1 millonésima 0.0000000001, se ingresara un ruido adicional a las antenas, escogiéndose si se la quiere agregar el ruido a la entrada o a la salida del programa.

Ahora evaluaremos el algoritmo con los siguientes valores: (problema3)

```
entrada(k)=sin((2*pi*k)/M);
senal_referencia(k)=cos((2*pi*k)/M);
n=2000 muestras, rango de seguridad N=80.
```

También hicimos un programa problema1 que utiliza la misma función senoidal de entrada como función de referencia, y un programa problema2 que utiliza señales gaussiana aleatorias como señal de entrada y realizamos algunas simulaciones hicimos esto tanto para el algoritmo LMS como para el algoritmo RLS, estos los colocaremos en el anexo F .

4.6 Pruebas y resultados de las implementaciones RLS.

En este programa problema1_rls hemos colocado una señal de entrada(k)=sin((2*pi*k)/M); para n=2000 muestras. Se utilizo una señal de referencia igual a la señal de entrada ingresada en una función de transferencia (y(n)) más un ruido añadido por el programa rls, este ruido adicional es del orden de 2, 3 y hasta 10 por ciento.

Para todos los ejemplos que utilicen el programa problema1_rls son validos los siguientes objetivos: demostrar que se pueden utilizar funciones senosoidales en algoritmos rls y evaluar los pesos de antenas utilizando el algoritmo adaptativo rls. Comparar estas funciones senosoidales con valores reales ingresados por el usuario. Hacer un grafico de esta comparación, hacer un grafico del error y un grafico de comparación entre la señal de referencia y la señal obtenida por la simulación. También determinar que las señales senosoidales pueden aplicarse en los algoritmos rls al igual como se aplican las señales gausianas. Otros objetivos de la presente corrida es ver la correlación del número de muestras con la convergencia de la señal, es decir con cuantas muestras se obtiene la mejor comparación de las señales de salida del filtro con la señal de referencia de la señal senosoidal, también se consideraría como otro objetivo del presente

programa que este tipo de datos obtenidos en forma experimental podrían darnos valores aproximados de antenas reales y evitar construir un sistema costoso de antenas para obtener este tipo de valores de prueba, esto podría bajar costos a alguna empresa que trabaje con señales senosoidales en condiciones parecidas al presente programa.

Nota1.- Kalman propuso que el algoritmo rls puede realizar las mismas aplicaciones que el algoritmo lms, tanto para señales gaussianas, como para señales senosoidales, cumpliendo con las teorías de Bernard Widrow , Ted Hoff y Wiener para algoritmos adaptativos.

Nota2.- Cuando realicemos diferentes simulaciones con el algoritmo rls, utilizaremos 3 programas: problema1_rls, problema2_rls y problema3_rls, en los programas problema1_rls y problema3_rls se utilizaran señales senosoidales como señales de entrada y como señales de referencia, en el problema3_rls habrá una opción para utilizar una señal diferente a la señal de entrada como señal de referencia con la única condición de que sea parecida a la señal original. El programa problema2_rls funcionara únicamente con funciones gaussianas.

TABLA XXXVIII Datos de entrada para el Problema1_rls_a

Tipo de señal utilizada: senosoidal

Algoritmo utilizado: rls

Aplicación en matlab utilizado: problema1_rls

Nombre dado al ejemplo: problema1_rls_a

Orden de la señal: 5

Numero de antenas: 5

Valores iniciales de entrada para las antenas:

V1= 0.0976 V, V2=0.2873 V, V3=0.3360 V, V4=0.2210 V, V5=0.0964 V

Valores de ruido ingresados para las antenas:

Ruido1= 0.01 V, ruido2=0.01 V, ruido3= 0.01, ruido4=0.01, ruido5=0.02

Opción_ruido=1, numero de muestras =2000

Constantes de ajuste: lamda = 0.999994115 delta = 100000000000000

Error del sistema = 0.0000001.

TABLA XXXIX Datos obtenidos por el ejemplo Problema1_rls_a

Pesos reales:	0.0976V	0.2873V	0.3360V	0.2210V	0.0964V
Pesos obtenidos rls:	-0.65V	1.5V	1.65V	0.9V	-1.5V

Para ver las pantallas de visualización de las potencias del sistema, señal obtenida a la salida, comparación de señales y curvas de error del presente ejemplo ir al anexo F ejemplo problema1_rls_a

TABLA XL Datos de entrada para el Problema1_rls_b

Tipo de señal utilizada: senosoidal

Algoritmo utilizado: rls

Aplicación en matlab: utilizado: problema1_rls

Nombre dado al ejemplo: problema1_rls_b

Orden de la señal: 5

Numero de antenas: 5

Valores iniciales de entrada para las antenas:

V1= 0.0976 V, V2=0.2873 V, V3=0.3360 V, V4=0.2210 V, V5=0.0964 V

Valores de ruido ingresados para las antenas:

Ruido1= 0.01 V, ruido2=0.01 V, ruido3= 0.01, ruido4=0.01, ruido5=0.02

Opción_ruido=1, numero de muestras =2000

Constantes de ajuste: lamda = 0.999994115 delta = 100000000000000

Error del sistema =0.0000001

TABLA XLI Datos obtenidos por el ejemplo Problema1_rls_b

Pesos reales	0.0976V	0.2873V	0.3360V	0.2210V	0.0964V
Pesos obtenidos rls:	0.05V	0.06V	-0.1V	0.07V	0.03V

Para ver las pantallas de visualización de las potencias del sistema, señal obtenida a la salida, comparación de señales y curvas de error del presente ejemplo ir al anexo F ejemplo problema1_rls_b

Para ver si logramos disminuir aun más el error , ingresaremos valores que estén dentro de la curva. (problema1)

TABLA XLII Datos de entrada para el Problema1_rls_c

Tipo de señal utilizada: senosoidal

Algoritmo utilizado: rls

Aplicación en matlab utilizado: problema1_rls

Nombre dado al ejemplo: problema1_rls_c

Orden de la señal: 5

Numero de antenas: 5

Valores iniciales de entrada para las antenas:

V1= 0.10 V, V2=0.08 V, V3=0.02 V, V4=0.05 V, V5=0.1 V

Valores de ruido ingresados para las antenas:

Ruido1= 0.01 V, ruido2=0.01 V, ruido3= 0.01V, ruido4=0.02V, ruido5=0.02V

Opción_ruido=1, numero de muestras =2000

Constantes de ajuste: lamda = 0.999994115 delta = 100000000000000

Error del sistema = 0.0000000001

TABLA XLIII Datos obtenidos por el ejemplo Problema1_rls_c

Pesos reales	0.10V	0.08V	0.02V	0.05V	0.1V
Pesos obtenidos rls:	-0.09V	0.14V	0.07V	0.03V	-0.045

Para ver las pantallas de visualización de las potencias del sistema, señal obtenida a la salida, comparación de señales y curvas de error del presente ejemplo ir al anexo F ejemplo problema1_rls_c

Para el programa problema2 utilizaremos como señal de entrada una señal gaussiana aleatoria `entrada=randn(N,1);` como rango de seguridad utilizaremos N=80 (zona de comprobación), la señal de referencia será una señal gaussiana. La señal de entrada será una señal gaussiana mas un ruido que se agregara a esta señal (se sumaran) además de que se les sumara un ruido propio del filtro rls todo este valor junto será ingresado a una función de transferencia Gz. En el sistema se podrá ingresar un ruido adicional a la entrada (opción 1) o a la salida (opción 2). Como utilizaremos una señal gaussiana esperamos tener un error inferior al que tenemos utilizando el programa con señales senosoidales.

TABLA XLIV Datos de entrada para el Problema2_rls_a

Tipo de señal utilizada: gaussiana

Algoritmo utilizado: rls

Aplicación en matlab utilizado: problema2_rls

Nombre dado al ejemplo: problema2_rls_a

Orden de la señal: 5

Numero de antenas: 5

Valores iníciales de entrada para las antenas:

V1= 0.0976 V, V2=0.2873 V, V3=0.3359 V, V4=0.2209 V, V5=0.0963 V

Valores de ruido ingresados para las antenas:

Ruido1= 0.01 V, ruido2=0.01 V, ruido3= 0.01, ruido4=0.02, ruido5=0.02

Opción_ruido=1, numero de muestras =2000, rango seguridad =80 muestras

Constantes de ajuste: lamda = 0.999994115 delta = 100000000000000

Error del sistema = 0.0000001

TABLA XLV Datos obtenidos por el ejemplo Problema2_rls_a

Pesos reales	0.0976V	0.2873V	0.3359V	0.2209V	0.0963V
Pesos obtenidos rls:	0.096V	0.286V	0.35V	0.2208V	0.09V

Para ver las pantallas de visualización de las potencias del sistema, señal obtenida a la salida, comparación de señales y curvas de error del presente ejemplo ir al anexo F ejemplo problema2_rls_a

Ahora evaluaremos el problema2 con otros valores de pesos iníciales y el ruido adicional lo colocaremos a la salida del sistema (opción 2)

TABLA XLVI Datos de entrada para el problema Problema2_rls_b

Tipo de señal utilizada: gaussiana

Algoritmo utilizado: rls

Aplicación en matlab utilizado: problema2_rls

Nombre dado al ejemplo: problema2_rls_b

Orden de la señal: 4

Numero de antenas: 4

Valores iniciales de entrada para las antenas:

V1= 0.10 V, V2=0.30 V, V3=0.28 V, V4=0.20 V

Valores de ruido ingresados para las antenas:

Ruido1= 0.01 V, ruido2=0.01 V, ruido3= 0.02 V, ruido4=0.02 V

Opción_ruido=2, numero de muestras =2000, rango seguridad =80 muestras

Constantes de ajuste: lamda = 0.999994115 delta = 100000000000000

Error del sistema = 0.0000000001

TABLA XLVII Datos obtenidos por el ejemplo Problema2_rls_b

Pesos reales	0.10V	0.30V	0.28V	0.20V
Pesos obtenidos rls:	0.10V	0.285V	0.32V	0.23V

Para ver las pantallas de visualización de las potencias del sistema, señal obtenida a la salida, comparación de señales y curvas de error del presente ejemplo ir al anexo F ejemplo problema2_rls_b

En el siguiente ejemplo utilizaremos el programa problema3_rls con otra curva como señal de referencia. Los objetivos del programa problema3 son los mismos de los del programa1_rls.Con los siguientes datos:

entrada(k)=sin((2*pi*k)/M); senal_referencia(k)=cos((2*pi*k)/M);este programa tiene la ventaja de que la señal de referencia no es igual a la señal de entrada, pero se usa una señal parecida del mismo tipo senosoidal que tenga una alta correlación con la señal de entrada. Además en el programa si se desea cambiar la ecuación de referencia solo hay que cambiar la ecuación en el lazo for donde está la ecuación de referencia. Realizaremos el siguiente programa con N=400

muestras, y rango de seguridad de N=80. Utilizaremos valores de lamda =0.999994115 y delta = 1000000000000

Suponemos que al utilizar una señal senosoidal diferente como señal de referencia el error debe de aumentar.

TABLA XLVIII Datos de entrada para el Problema3_rls_a

Tipo de señal utilizada: senosoidal

Algoritmo utilizado: rls

Aplicación en matlab utilizado: problema3_rls

Nombre dado al ejemplo: problema3_rls_a

Orden de la señal: 5

Numero de antenas: 5

Valores iniciales de entrada para las antenas:

V1= 0.10 V, V2=0.20 V, V3=0.22 V, V4=0.15 V, V5=0.09 V

Valores de ruido ingresados para las antenas:

Ruido1= 0.01 V, ruido2=0.01 V, ruido3= 0.02 V, ruido4=0.02 V, ruido5=0.02V

Opción_ruido=1, opción_señal=2, numero de muestras =2000,

rango seguridad = 80 muestras

Constantes de ajuste: lamda = 0.999994115 delta = 100000000000000

Error del sistema = 0.0000000001

TABLA XLIX Datos obtenidos por el ejemplo Problema3_rls_a

Pesos reales	0.10V	0.20V	0.22V	0.15V	0.09V
Pesos obtenidos rls:	0.38V	-0.1V	-0.29V	-0.2V	0.2V

Para ver las pantallas de visualización de las potencias del sistema, señal obtenida a la salida, comparación de señales y curvas de error del presente ejemplo ir al anexo F ejemplo problema3_rls_a

Ahora evaluaremos el problema 3 con la señal de referencia en base a la señal de entrada. (opción_señal=1),

entrada(k)=sin((2*pi*k)/M); senal_referencia(k)=y(n)+ruido; luego compararemos el resultado con los obtenidos con la opción 2, seguiremos utilizando los mismos valores de entrada de pesos iniciales. Realizaremos el siguiente programa con N=400 muestras, y rango de seguridad de N=80. Utilizaremos valores de lamda = 0.999994115 ; y delta=100000000000000 utilizaremos y(n)=lsim(Gz,entrada)

TABLA L Datos de entrada para el Problema3_rls_b

Tipo de señal utilizada: senosoidal

Algoritmo utilizado: rls

Aplicación en matlab utilizado: problema3_rls

Nombre dado al ejemplo: problema3_rls_b

Orden de la señal: 5

Numero de antenas: 5

Valores iniciales de entrada para las antenas:

V1= 0.10 V, V2=0.20 V, V3=0.22 V, V4=0.15 V, V5=0.09 V

Valores de ruido ingresados para las antenas:

Ruido1= 0.01 V, ruido2=0.01 V, ruido3= 0.01V, ruido4=0.02V, ruido5=0.02V

Opción_ruido=1, opción_señal=1, numero de muestras =400,

Constantes de ajuste: lamda = 0.999994115 delta = 100000000000000

rango seguridad =80 muestras , Error del sistema = 0.0000000001.

TABLA LI Datos obtenidos por el ejemplo Problema3_rls_b

Pesos reales:	0.10V	0.20V	0.22V	0.15V	0.09V
Pesos obtenidos rls:	-0.025V	0.05V	0.07V	0.05V	-0.03V

Para ver las pantallas de visualización de las potencias del sistema, señal obtenida a la salida, comparación de señales y curvas de error del presente ejemplo ir al anexo F ejemplo problema3_rls_b.

CAPÍTULO 5

EL ALGORITMO ADAPTATIVO DE MÓDULO CONSTANTE (CMA)

El algoritmo adaptativo de módulo constante (CMA) es un algoritmo de igualación ciega, es decir que no requiere señal de referencia, ya que estos generan su propia señal de referencia desde las señales receptadas, esta es una ventaja que poseen sobre los algoritmos lms y rls, aunque en ocasiones no se utilice esta propiedad del algoritmo cma, ya que muchas veces se prefiere hacer un algoritmo hibrido lms-cma o rls-cma para aprovechar las características lineales que poseen los algoritmo lms y rls.

Fue creado en 1980 por GODARD, también fue propuesto independientemente por J.R. TREICHLER y M. G. AGEE y revisado en 1993 por SHALVI Y WEINSTEIN los cuales agregaron criterios adicionales al algoritmo de módulo constante. El algoritmo cma utiliza técnicas más complejas que el lms o el rls, utilizando características conocidas de la señal deseada, como alguna modulación, algún tipo de

ciloestacionariedad, etc. También se caracteriza por utilizar el algoritmo de kalman en su desarrollo.

Una característica del algoritmo módulo constante es que no requieren una fase diferenciada de entrenamiento, es decir que en pocas interacciones el sistema puede llegar a la convergencia, aunque para esto es necesario colocar buenos valores al ingresar los datos en el programa, y que no halla brincos en la señal que haga que se pierda, el algoritmo cma suele utilizar conocimientos y datos de la señal transmitida. Esta es una cualidad del algoritmo cma, que puede ser una ventaja del algoritmo, muchas personas al programar y usar el algoritmo cma suelen también desactivar esta cualidad y no utilizarla para decidir cuantas interacciones hacer, es decir que se convierten en un sistema dirigido por decisión, algunos al programar hacen esto para no depender por completo de lo que hace el programa y tener control sobre el mismo. El uso de técnicas de igualación completamente ciegas presenta las siguientes ventajas:

- Permite emplear protocolos de comunicación más sencillos, por ejemplo, eliminando la necesidad de retransmitir secuencias de entrenamiento tras desvanecimientos en el canal.

- Disminuye los problemas de interoperabilidad entre equipos, derivados del uso de distintas secuencias de entrenamiento.
- Ahorra ancho de banda en redes de difusión ya que evita intercalar periódicamente secuencias de entrenamiento.

5.1 Algoritmo CMA

La solución mínimo cuadrado (capitulo 3) y la solución mínimo cuadrado recursivo (capitulo 4) necesitan muchas iteraciones para llegar a su convergencia (el algoritmo lms varia de 200, 400, 800 y hasta 2000 iteraciones), en el algoritmo rls puede variar de 20, 80, 100, 200 y hasta 400 iteraciones según el grado de error que se quiera implementar, por eso es que se ideo hacer nuevos algoritmos como el cma, dmi o music que convergan mas rápido. Además de poseer nuevas características y que sean implementado en señales puntuales. El algoritmo CMA tiene la cualidad de presentarse en grupos en forma de granulos, simulando una característica del cerebro humano, de las neuronas del cerebro, esta es una cualidad que tiene el algoritmo cma.el algoritmo cma se suele utilizar en señales 16 QAM, 32 QAM, 64 QAM

El algoritmo de módulo constante o de Godard es el algoritmo de igualación ciega mas popular, el cual minimiza el valor de J en función de los pesos w. ecuación 5.1

$$J(w(n)) = E\left[(R-|y(n)|^2)^2\right] \quad (5.1)$$

Donde R es una constante positiva que depende de las propiedades del sistema, valor de u, número de elementos, etc. Esto lo vemos en la ecuación 5.2

$$w(n+1) = w(n) + \mu\,(R-y(n)y^*(n))y(n)u^*(n) \quad (5.2)$$

Donde el asterisco denota el valor conjugado de un escalar, o de todos los elementos de un vector, el uso del algoritmo cma presenta dos inconvenientes, el primero es que su aplicación únicamente permite recuperar una versión girada de los valores originales, por lo que es necesario introducir una etapa de recuperación de fase (esta fase de recuperación también es adaptativa), y el segundo inconveniente es cuando se utiliza el algoritmo cma para igualar canales en sistemas con modulaciones cuyos valores no son de norma constante, es decir que cambian de manera variable, y en otros de manera aritmética como en los

sistemas lineales, la igualación se ve sujeta a un ruido adicional además de la producida por el propio sistema, canal y algoritmo. Por lo que en algunos casos ese ruido adicional puede causar que el algoritmo tenga un desempeño inferior a los dados por un algoritmo lms funcionando en modo dirigido por decisión. Por eso algunos optan por hacer híbridos lms con cma, los cuales trabajarían de forma ciega, muchos prefieren trabajar con un filtro lms que actúa por decisión con el valor de referencia a la salida, y para los valores de entrada del filtro se suele utilizar los valores de los dos algoritmos lms y cma, siendo más importante para el programa los valores del cma. El igualador diseñado es completamente ciego, si se emplea un combinador cma-lms se actualizan simultáneamente un componente ciego y el otro es dirigido por decisión, combinándose en cada instante de manera adecuada. De esta manera los dos algoritmos dan cualidades al sistema, esto permite que cuando hay cambios brusco en la señal la parte lineal del lms lo devuelve a donde debería estar, otros prefieren colocar una fase de recuperación de señal para el filtro cma, aunque más fácil es utilizar un componente lms para hacer este trabajo.

Un ejemplo de esto es se realiza en los canales de comunicaciones.: considere un canal de comunicaciones de 64 QAM en el que la señal viaja con una respuesta al impulso de h transpuesta=[0.1 , 0.2 , 1 , -0.5 , -0.1 , 0.2] , el cual produce un

cambio instantáneo h transpuesta=[0.23 , 0.63 , 0.7 , -0.54]. El canal se ve sujeto a ruido aditivo gaussiano, cuya potencia se ha ajustado para obtener una relación señal a ruido (suele utilizar valores de 30, 32, 34 db a la entrada del igualador)

Un ejemplo de esto lo vemos en la figura 5.1, en donde se ve la relación error vs n, de un algoritmo cma, un lms y un hibrido cma-lms.

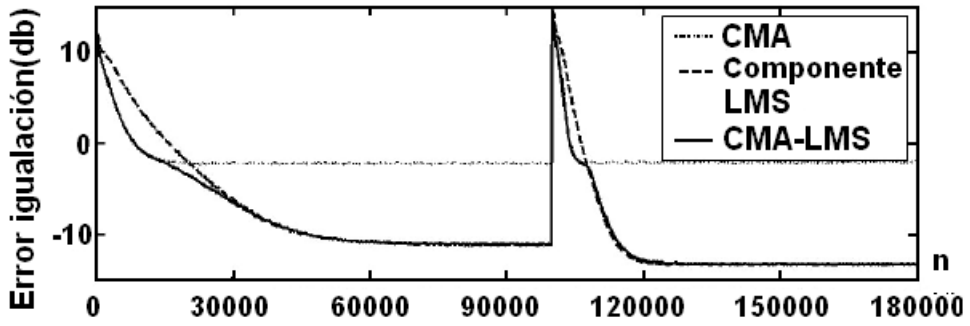


Fig 5.1 Error cuadratico medio de la señal 64-QAM recuperada por un igualador combinador CMA-LMS con ucma=5.10^{-7}, $u_{LMS}=5.10^{-6}$ y ua=10^{-3}.

En la figura 5.1 se ve la evolución de la diferencia media entre la señal transmitida y la recuperada por el igualador, tanto para un igualador cma u=5*10^-7 como para un combinador cma-lms con un paso u=5*10^-6. Se mezclan los dos y se obtiene un valor actualizado ua=10^-3. En el presente grafico se ha incluido la evolución del componente lms, para poder compararlo,

podemos comprobar cómo el componente cma-lms tiene una excelente prestación en los tramos estacionarios, y como el componente ciego es utilizado para la igualación inicial, logrando en ambos casos forzar una rápida convergencia del componente lms.

Un diagrama de bloques para un cancelador cma-lms se ve en la figura 5.2

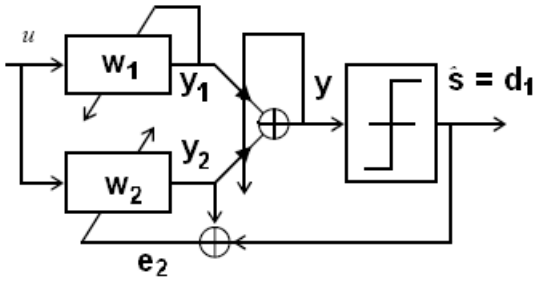


Fig 5.2 diagrama de Bloques del cancelador ciego CMA-LMS

En la figura 5.3 se muestra los valores obtenidos que aparecen tras la convergencia de un igualador ciego w1(n), con un componente cma (a) y con un componente hibrido cms-lms (b), en el cual se ve a simple vista la gran reducción del error y disminución de la dispersión de los valores obtenidos con el sistema cma-lms., se ve que conserva las propiedades del algoritmo cma más la garantía de convergencia del lms.

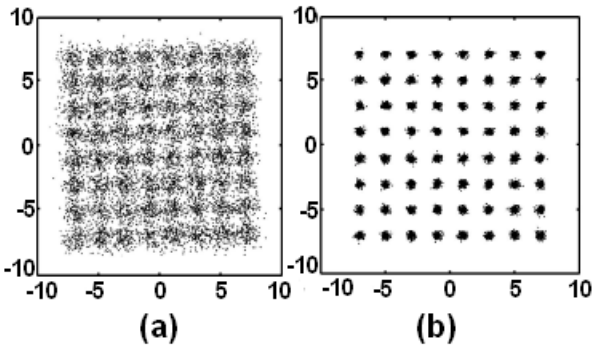


Fig 5.3 Constelaciones obtenidas por el igualador combinado CMA-LMS. (a) salida del componente CMA. (b) Salida del sistema completo CMA-LMS

Para hacer un análisis más a fondo del algoritmo CMA revise la información sobre el filtro de kalman en el anexo G. y el 5.1.2

5.1.1 Base teórica del algoritmo adaptativo de Modulo Constante CMA

El estudio del algoritmo adaptativo CMA está muy relacionado con el desarrollo del filtro de kalman.

Este método es muy útil para resolver problemas que utilizan, hasta un canal de segundo orden, es muy práctico para determinados fragmentos, como es de suponer al aumentar el orden aumenta el error, pero si se tiene buenos valores del filtro disminuye considerablemente.

El filtro de kalman y el algoritmo adaptativo CMA se utilizan para determinar valores, porcentajes, convergencias, etc. Para poder utilizarlas en algunas ocasiones se requiere de un conocimiento a priori de los segundos momentos de orden. A continuación se realizaremos un pequeño análisis matemático de los algoritmos CMA, primero se hablara un poco de los algoritmos CMA y como usa el filtro de kalman.

5.1.2 Análisis matemático del algoritmo adaptativo de Modulo Constante CMA

El algoritmo CMA es un algoritmo de igualación ciega, estos obtienen las señales a partir de las observaciones y de las características de la señal de entrada, como es la fase, la cicloestabilidad, el valor de n, las técnicas de igualación ciega utilizan las características estadísticas de la señal de entrada.

El algoritmo CMA proviene del algoritmo de SATO, desarrollado en 1974. son casos particulares de la minimización estocástica, utilizan formulas comunes siendo la única diferencia el orden que utilizan el algoritmo de

SATO es de orden=1, y el algoritmo CMA es de orden=2. En el orden 2 es cuando el algoritmo CMA tiene su máximo rendimiento.

La ecuación general seria la 5.3. notese que cuando p=2 se convierte en la ecuación del algoritmo CMA (ecuación 5.1)

$$J(\mathbf{w}) = E[(|y(n)|^p - R_p)^2] \quad \textbf{(5.3)}$$

El valor de Rp lo obtenemos de los valores estimados de la señal de entrada evaluados para un número de canal p. ecuación 5.4

$$R_p = \frac{E[|s(n)|^{2p}]}{E[|s(n)|^p]} \quad \textbf{(5.4)}$$

En la ecuación 5.5 vemos la ecuación de pesos para cualquier valor de p

$$\mathbf{w}_{n+1} = \mathbf{w}_n + \mu \mathbf{x}_n (R_p - |y(n)|^p) |y(n)|^{p-2} y(n) \mathbf{x}_n \quad \textbf{(5.5)}$$

p=1 ➡ Sato

p=2 ➡ CMA

Para valores complejos se utiliza Xn conjugada ecuación 5.6

Con constelaciones complejas

$$W_{n+1} = W_n + \mu X_n (R_p - |y(n)|^p) |y(n)|^{p-2} y(n) X_n^* \quad \textbf{(5.6)}$$

El algoritmo CMA suele emplearse de dos maneras distintas una es para igualación ciega y la otra es para desconvolución.

El método por desconvolución CMA fue desarrollado por Shalvi y weinstein en 1993, de este método hablaremos más a fondo en el 5.3.1 cuando hablemos de las variantes del algoritmo CMA.

El método por igualación ciega CMA la señal de entrada pertenece a un conjunto finito de valores. Este método utiliza el filtro de kalman para su desarrollo, es decir que necesitara el programa los valores de landa y de

delta para su funcionamiento. Esto lo vimos en la ecuación 4.6 del capitulo 4

$$\eta_n(k) = \lambda^{n-k} \qquad k=1,2,\ldots,n \tag{4.6}$$

Donde el valor de λ es menor que 1, y de $\eta_n(k)$, (este valor de landa es un coeficiente del filtro) el cual es confinado en el rango de:

$$0 < \eta_n(k) \leq 1 \text{ for } k = 1, 2, \ldots, n.$$

EL factor de peso de λ es también conocido como el factor olvidado (a veces puede ser importante para lograr una buena convergencia) subsecuentemente el peso de datos recientes tienden a olvidarse con el paso del tiempo. Esta propiedad es producida por los algoritmos adaptativos con alguna capacidad de rastreo. De hay que nosotros podemos minimizar la función de J(n) (4.7)

$$J(n) = \sum_{k=1}^{n} \lambda^{n-k} e^2(k) \tag{4.7}$$

Algunas veces si la convergencia no es buena en el algoritmo CMA, lo que se hace es normalizar el algoritmo y tratar de utilizar medios matemáticos que disminuyan el error, como manipular el valor de pesos W, el valor de landa y el valor de delta y u con eso se logra una mejor convergencia, otros han optado por agregar métodos de búsqueda (llamados BL) que ayudan a buscar la señal y aumentar la convergencia. Estos presentan muy buen comportamiento en optimización continua, dentro de estos se encuentran el algoritmo CMA-ES también llamado algoritmo adaptativo de matriz de covarianza, este algoritmo es una variante del algoritmo CMA, en este se logra una mejor convergencia, pero aun así no es tan segura como en el algoritmo LMS.

El valor mínimo de J(n) es logrado con la ecuación normal 4.8, una característica del algoritmo cma es que al normalizar la ecuación se puede disminuir el error.

$$\mathbf{R}_{\lambda}(n)\,\hat{w} = \mathbf{P}_{\lambda}(n) \quad (\hat{w} = \mathbf{R}_{\lambda}^{-1}\,\mathbf{P}_{\lambda}(n)) \tag{4.8}$$

Donde la matriz de correlación de MxM de la matriz $\mathbf{R}_{\lambda}(n)$ Nos produce valores más satisfactorios ecuaciones 4.9 y 4.10.

$$R_\lambda(n) = \sum_{k=1}^{n} \lambda^{n-k} x(k)x^T(k) = X^T \Lambda X \quad (4.9)$$

$$P_\lambda(n) = \sum_{k=1}^{n} \lambda^{n-k} x(k)d(k) = X^T \Lambda d \; ; \Lambda = diag[\lambda^{n-1} \; \lambda^{n-2} \ldots 1] \quad (4.10)$$

La diferencia más grande de este algoritmo CMA con el algoritmo RLS, es que el algoritmo CMA no posee señal de referencia, y puede trabajar para canales QAM, de hay el desarrollo matemático es casi igual excepto si se utiliza el método de desconvolución que tiene múltiples condiciones de funcionamiento.

Otra de las características del algoritmo CMA es que pretende restaurar la propiedad del módulo constante a la salida del igualador (aunque puede funcionar con modulaciones multinivel), en el algoritmo cma la igualación y la recuperación de la portadora pueden ser simultaneas, esto lo vemos en la ecuaciones de pesos y en la ecuación del error (ecuación 5.7 y ecuación 5.8).

$$J(\mathbf{w}) = E[(|y(n)|^p - R_p)^2] \quad (5.3)$$

$$R_p = \frac{E[|s(n)|^{2p}]}{E[|s(n)|^{p}]} \quad (5.4)$$

$$W(n+1) = W(n) - ux(n)\varepsilon^*(n) \quad (5.7)$$

$$\varepsilon(n) = [\,1-|y(n)|^2\,]\,y(n)x(n) \quad (5.8)$$

Los valores de pesos obtenidos por el algoritmo CMA, tenderán a moverse alrededor de los pesos w utilizados por el filtro CMA, para evitar esto se recomienda aplicar la conjugada de e (ecuación 5.7) y utilizar la ecuación 5.5, notece que en la ecuación 5.7 se utiliza una resta en la ecuación de pesos, también plantean utilizar el valor de Rp con diferentes valores y también con la señal de entrada. mientras que en la ecuación 5.5 se utiliza una suma. La ecuación 5.5 fue propuesta independientemente por Godard y por Treichler y M. G. Agee. Con la idea de restaurar la propiedad de modulo constante a la salida del igualador.

Shalvi y Weinstein determinaron que el algoritmo cma no siempre converge**.**

5.1.3 Variaciones con el algoritmo CMA

Analizaremos varias variaciones con el algoritmo CMA. Una de ellas es la ecuación de Shalvi – Weinstein, y la ecuación del algoritmo CMA Normalizado.

El Metodo de Shalvi – Weinstein.

Fue desarrollado en 1993, como una variante del algoritmo CMA, en el cual se colocan ciertas restricciones a las curvas para obtener nuevos resultados, siendo un método de manipulación matemática, se basa en el uso de valores llamados cumulantes, permiten desconvolucionar señales. En la desconvolución la señal de entrada tiene una función de p continuo (pueden ser datos sísmicos, puntos de imágenes, etc) en la desconvolución no es posible un método guiado por decisión, entre las condiciones de aplicación de la formula de shalvi weinstein tenemos que la desconvolución para antenas siso ciegas deben hacer uso explicito o implicito de estadísticos de orden superior, fig 5.4 :

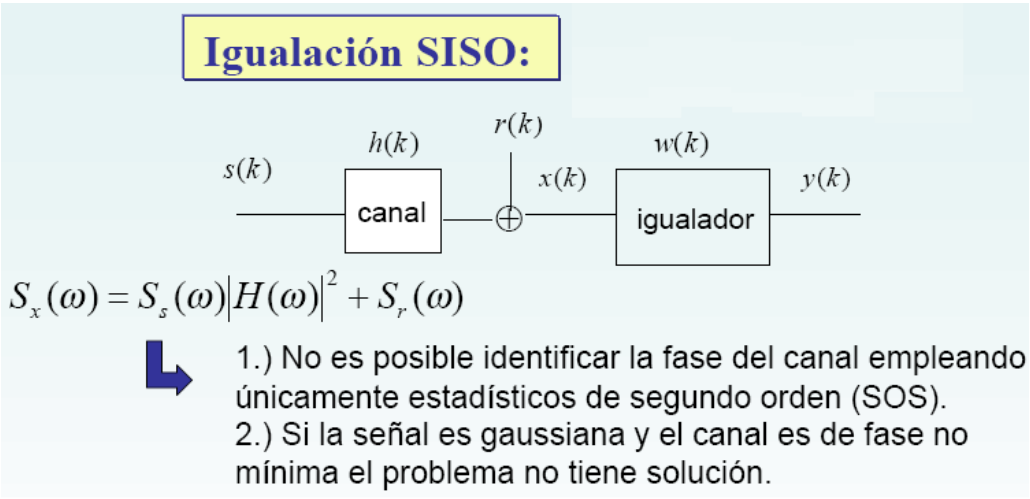


fig 5.4 Igualacion SISO por el metodo CMA Shalvi - Weinstein

Entre las condiciones del método CMA de Shalvi – Weinstein tenemos:

1.- Que no es posible identificar la fase del canal empleando únicamente estadísticos de segundo orden

2.- Si la señal es gaussiana y el canal es de fase no mínima el problema no tiene solución.

3.- Deben cumplir las características de los cumulantes, los cumulantes de orden 2, 3 y 4 se definen como:

$$cum(x_1, x_2) = E[x_1 x_2] \quad (5.9)$$

$$cum(x_1, x_2, x_3) = E[x_1 x_2 x_3] \quad (5.10)$$

$$cum(x_1, x_2, x_3, x_4) = E[x_1 x_2 x_3 x_4] - E[x_1 x_2]E[x_3 x_4] - E[x_1 x_3]E[x_2 x_4] - E[x_1 x_4]E[x_2 x_3] \quad (5.11)$$

4.- Si las variables son conjuntamente gaussianas, los cumulantes de orden >2 son todos nulos.

5.- Para una secuencia de x(n) complejas se lo define como:

$$C^{x}_{p,q} = cum(\underbrace{x(n), \ldots, x(n)}_{p}; \underbrace{x^*(n), \ldots, x^*(n)}_{q}) \quad (5.12)$$

6. – La respuesta combinada del canal y del igualador es f=h*w

7.- La relación entre los cumulantes a la salida del igualador y la secuencia de valores de p y q es: ecuación 5.13

$$C_{p,q}^{y} = C_{p,q}^{s} \sum_{n} f_n^{p} \left(f_n^{*}\right)^{q} \qquad (5.13)$$

8.- La propiedad básica de Shelvi y weinstein, que dice que la igualdad se cumple solo si f cumple las siguientes condiciones:

$$\text{si} \quad C_{1,1}^{y} = C_{1,1}^{s} \quad \text{entonces} \quad C_{p,q}^{y} \le C_{p,q}^{s} \qquad (5.14)$$

La igualdad se produce sólo si f es de la forma

$$f = Ke^{j\phi}\delta(n-d) \qquad (5.15)$$

9.- La familia de criterios de igualación es :

$$\max\left|C_{p,q}^{y}\right| \quad \text{sujeto a} \quad C_{1,1}^{y} = C_{1,1}^{s} \qquad (5.16)$$

10.- Para el caso de p=q=2 se debe cumplir que landa=2 y landa=3 , estos valores evaluados se los llamara kurtosis.

$$C_{2,2}^{y} = E\left[|y(n)|^4\right] - \lambda\left(E\left[|y(n)|^2\right]\right)^2 \begin{cases} \lambda = 3 & \text{Sec. reales} \\ \lambda = 2 & \text{Sec. complejas} \end{cases} \quad (5.17)$$

11.- El criterio de shelvi y weinstein se reduce a maximizar la kurtosis sujeto a una restriccción de la varianza.

Todas estas condiciones explican porque muy pocas personas utilizan el método CMA Shalvi - Weinstein por su complejidad matemática.

Variación utilizando el algoritmo NCMA (Algoritmo de modulo constante normalizado)

Utilizando el algoritmo CMA, se obtiene la cancelación de las interferencias (ruido), siempre y cuando la señal principal tenga una potencia superior que la de la señal interferente, esto lo investigaron Treichler en 1985 y Gooch en 1986, esta condición se cumple siempre en la segunda etapa del conformador de haz, ellos propusieron para mejorar la convergencia la normalización de la señal, utilizando la constante de desajuste, normalizada con respecto a la potencia estimada de la señal de entrada más una constante positiva (ecuación 5.18).

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \frac{\alpha}{a+\|\mathbf{x}(n)\|^2}[1-|y(n)|^2]y^*(n)\mathbf{x}(n) \qquad \textbf{(5.18)}$$

Esta normalización evita la amplificación del ruido del gradiente que tiene lugar para valores elevados de la señal de entrada en los algoritmos adaptativos de gradiente no normalizados.

5.2 Pantallas de visualización del algoritmo adaptativo Algoritmo de modulo constante CMA

Al hacer correr cualquiera de las aplicaciones en matlab del algoritmo adaptativo de modulo constante CMA que hemos realizado se presentaran algunas pantallas de visualización, algunas de estas pantallas son para información de los usuarios a usar el programa, otras son para ingresar valores de entrada, las ultimas son las generadas por el programa matlab y son pantallas de salida, con estos datos de entrada y salida formaremos tablas de datos y resultados obtenidos. Todas estas pantallas servirán tanto para señales senosoidales como para señales gaussianas. También podrían servir para otros tipos de señales que cumplan las características señaladas por kalman para filtros adaptativos.

Las pantallas a utilizar en el programa del algoritmo adaptativo CMA son:

Pantalla de presentación:

En esta pantalla se hace una explicación al usuario de las características de la aplicación y lo que resuelve, está el nombre de la universidad, los nombres de los programadores del programa, el profesor guía, el tipo de señal a usar en el programa, , una pequeña explicación de los datos de entrada y los de salida.

Pantalla de ingreso de datos:

En esta pantalla ingresamos el numero de antenas a utilizar (en el algoritmo CMA de 1 a 15). los valores iníciales de las antenas, los valores de ruido de las antenas.

En esta pantalla aparecerá una opción llamada opción ruido que si es igual a 1 el ruido se ingresara a la entrada del sistema (antes del filtro), si se escoge 2 el ruido se agregara a la salida del sistema. También se ingresara un valor de error del sistema

Pantalla de potencias obtenidas:

Esta pantalla nos dará la potencia del sistema y el valor de la potencia de cada una de las antenas.

Pantalla de comparación de pesos estimados:

En esta pantalla presentaremos los datos iníciales ingresados por teclado (los llamaremos también valores actuales) los representaremos por medio de una cruz negra, y los compararemos con los valores obtenidos por el algoritmo CMA, este

valor obtenido lo llamaremos valores estimados, y lo representaremos con un color rojo.

Pantalla de valor de la curva de error:

En esta pantalla se muestra el comportamiento de la ecuación de error del algoritmo CMA, esta nos servirá para compararla con el comportamiento de otros algoritmos adaptativos.

Pantalla de comparación entre la salida del sistema con la señal de referencia:

Esta pantalla es una de las más importantes, aparecerá la señal que obtenemos a la salida del sistema (color rojo) , y se la comparara con la señal de referencia (color azul).

En el anexo F se muestran las pantallas de visualización que obtendremos en nuestras implementaciones al utilizar diferentes valores de entrada y diferentes casos en cada uno de los algoritmos utilizados.

5.3 Programa en matlab del algoritmo adaptativo Algoritmo de modulo Constante CMA

Al hacer el programa en matlab del algoritmo CMA, decidimos utilizar como base la programación del algoritmo RSL, en el cual ya se utiliza el filtro de kalman que es utilizado por el algoritmo CMA, a este algoritmo se le agregara las ecuaciones del algoritmo CMA, principalmente su ecuación de pesos, esto también tiene su razón de ser en que leyendo la teoría del algoritmo CMA se suele presentar perdidas de la señal, y al aplicarlo dentro de un esquema RLS esperamos que estas pérdidas se disminuyan considerablemente, este algoritmo podría considerárselo un hibrido CMA-RLS. Al igual que en el programa RLS utilizamos una función principal que será llamada desde el programa matlab, esto permitirá ciertas ventajas ya que todo el programa está en una sola función.

Consideramos en el programa un orden 2 para la segunda etapa de un conformador de haz, esto es para que la ecuación de pesos que se utiliza en el programa sea la de un algoritmo CMA, este valor será constante en la ecuación, el valor del orden del filtro para determinar los valores de cota y de la potencia si variaran normalmente como si trabajaran en el algoritmo RLS.

Programa Principal y función principal

```
function[caratula]=problema3_cma;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
% Algoritmo CMA
% Autores: Juan Alvarez y Maribel Chuez
%
%Escuela Superior Politecnica del Litoral
%Facultad de Electricidad y Computación
%Guayaquil, Ecuador
% email :juan_alvarez8@latinmail.com
% email : mchuez@ceibo.espol.edu.ec
% Webpage : proximamente
% Date    : Miercoles 22 de Julio del 2009
% Version : 1.0.2
% Referencias : S. Haykin, Adaptive Filter Theory. 3rd edition, Upper
Saddle River, NJ: Prentice-Hall, 1996.
%
%clear all % esta funcion borra los valores anteriores que esten
memoria %
%close all % esta funcion borra cualquier ventana anterior activa %
%hold off   % esta funcion encera cualquier grafico que estuviera
activo %
 fprintf('*****************************************************.\n')
fprintf('*    ESCUELA SUPERIOR POLITECNICA DEL LITORAL        *.\n')
fprintf('*                                                    *.\n')
fprintf('* PROBLEMA3_CMA.- Determinar los pesos, el numero de *.\n')
fprintf('* iteraciones y la potencia de un grupo de antenas   *.\n')
fprintf('* inteligentes utilizando el algoritmo CMA-RLS       *.\n')
fprintf('* las antenas estan distribuidas en forma de celdas  *.\n')
fprintf('* para lo cual requeriremos de los siguientes datos: *.\n')
fprintf('*                                                    *.\n')
fprintf('*  • la señal aplicada a la entrada del filtro,      *.\n')
fprintf('*   ej  Xn=sin((2*pi*k)/N)                           *.\n')
fprintf('*  • el numero de antenas  x                         *.\n')
fprintf('*  • la señal de referencia, d(k)=cos((2*pi*k)/N);   *.\n')
fprintf('*  • el ruido de las diferentes antenas, (hay que    *.\n')
fprintf('*  tener presente que el algoritmo CMA en su         *.\n')
fprintf('*  estructura internamente produce un ruido que puede*.\n')
fprintf('*  ser de 2, 3 y hasta 10 %  aparte de este ruido    *.\n')
fprintf('*  generado por el mismo programa CMA se puede       *.\n')
fprintf('*  ingresar ruidos adicionales en el programa a la   *.\n')
fprintf('*  entrada o a la salida del programa.  n(f)         *.\n')
fprintf('*  • el orden del filtro FIR cuyos coeficientes se   *.\n')
fprintf('*  van a ajustar, orden                              *.\n')
```

```
fprintf('*  • el número de iteraciones en el aprendizaje, N   *.\n')
fprintf('*  (este número se hará  hasta que cumpla el error   *.\n')
fprintf('*  • la constante de ajuste   u                      *.\n')
fprintf('*  • el vector de pesos inicial. Los voltajes de las *.\n')
fprintf('*  antenas V(f)                                      *.\n')
fprintf('*  • el error deseado puede ser ingresado por teclado*.\n')
fprintf('*    error                                           *.\n')
fprintf('*                                                    *.\n')
fprintf('*  A su salida devolverá:                            *.\n')
fprintf('*  • la salida en cada iteración durante el proceso  *.\n')
fprintf('*  de ajuste, y(t)                                   *.\n')
fprintf('*  • la curva de aprendizaje (grafico de la curva)   *.\n')
fprintf('*  • los vectores de pesos obtenidos durante el pro- *.\n')
fprintf('*  ceso. W(t)                                        *.\n')
fprintf('*  • la Potencia de la antena P(k)                   *.\n')
fprintf('*  • el numero de iteraciones en que converge la     *.\n')
fprintf('*    curva y se cumple el valor del error            *.\n')
fprintf('*                                                    *.\n')
fprintf('* alumno: Juan Francisco Alvarez Alvarado            *.\n')
fprintf('* alumno: Maribel del Rosario Chuez Gonzales         *.\n')
fprintf('*                                                    *.\n')
fprintf('* Profesor :Ing. Pedro Vargas                        *.\n')
fprintf('*                                                    *.\n')
fprintf('******************************************************.\n')
t=input('teclee enter para proseguir');
 clc
% ingresamos el orden del filtro, cuyo coeficiente se va ha ajustar
fprintf('consideraremos un orden = 2  para la segunda etapa de un ')
fprintf('conformador de haz, esto es para que la ecuación de      ')
fprintf('pesos que se utiliza en el programa sea la de un         ')
fprintf('algoritmo CMA, este valor será constante en la ecuación  ')
fprintf('el valor del orden del filtro para determinar los valo   ')
fprintf('res de cota y de la potencia si variaran normalmente como')
fprintf('si trabajaran en un algoritmo RLS                        ')
fprintf('                                                         ')
orden=input(' ingrese el orden del filtro, cuyo coeficiente se va ha
ajustar....')
% ingresamos el numero de antenas del sistema
disp(' restriccion: ingrese un numero de antenas igual al del orden del
filtro ingresado)');
x=input(' ingrese el numero de antenas entre
1,2,3,4,5,6,7,8,9,10,11,12,13,14 o 15....');
if x==1
    D(1)=input('ingrese el voltaje1 (en V)..... voltaje1=');
end
if x==2
    D(1)=input('ingrese el voltaje1 (en V)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
end
```

```
if x==3
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
end

if x==4
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
end
if x==5
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
end
if x==6
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
end
if x==7
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
    D(7)=input(' ingrese la voltaje7 (en v)..... voltaje7=');
end
if x==8
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
    D(7)=input(' ingrese la voltaje7 (en v)..... voltaje7=');
    D(8)=input(' ingrese la voltaje8 (en v)..... voltaje8=');
end
if x==9
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
```

```
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
    D(7)=input(' ingrese la voltaje7 (en v)..... voltaje7=');
    D(8)=input(' ingrese la voltaje8 (en v)..... voltaje8=');
    D(9)=input(' ingrese la voltaje9 (en v)..... voltaje9=');
end
if x==10
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
    D(7)=input(' ingrese la voltaje7 (en v)..... voltaje7=');
    D(8)=input(' ingrese la voltaje8 (en v)..... voltaje8=');
    D(9)=input(' ingrese la voltaje9 (en v)..... voltaje9=');
    D(10)=input(' ingrese la voltaje10 (en v)..... voltaje10=');
end
if x==11
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
    D(7)=input(' ingrese la voltaje7 (en v)..... voltaje7=');
    D(8)=input(' ingrese la voltaje8 (en v)..... voltaje8=');
    D(9)=input(' ingrese la voltaje9 (en v)..... voltaje9=');
    D(10)=input(' ingrese la voltaje10 (en v)..... voltaje10=');
    D(11)=input(' ingrese la voltaje11 (en v)..... voltaje11=');
end
if x==12
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
    D(7)=input(' ingrese la voltaje7 (en v)..... voltaje7=');
    D(8)=input(' ingrese la voltaje8 (en v)..... voltaje8=');
    D(9)=input(' ingrese la voltaje9 (en v)..... voltaje9=');
    D(10)=input(' ingrese la voltaje10 (en v)..... voltaje10=');
    D(11)=input(' ingrese la voltaje11 (en v)..... voltaje11=');
    D(12)=input(' ingrese la voltaje12 (en v)..... voltaje12=');
end
if x==13
```

```
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
    D(7)=input(' ingrese la voltaje7 (en v)..... voltaje7=');
    D(8)=input(' ingrese la voltaje8 (en v)..... voltaje8=');
    D(9)=input(' ingrese la voltaje9 (en v)..... voltaje9=');
    D(10)=input(' ingrese la voltaje10 (en v)..... voltaje10=');
    D(11)=input(' ingrese la voltaje11 (en v)..... voltaje11=');
    D(12)=input(' ingrese la voltaje12 (en v)..... voltaje12=');
    D(13)=input(' ingrese la voltaje13 (en v)..... voltaje13=');
end
if x==14
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
    D(7)=input(' ingrese la voltaje7 (en v)..... voltaje7=');
    D(8)=input(' ingrese la voltaje8 (en v)..... voltaje8=');
    D(9)=input(' ingrese la voltaje9 (en v)..... voltaje9=');
    D(10)=input(' ingrese la voltaje10 (en v)..... voltaje10=');
    D(11)=input(' ingrese la voltaje11 (en v)..... voltaje11=');
    D(12)=input(' ingrese la voltaje12 (en v)..... voltaje12=');
    D(13)=input(' ingrese la voltaje13 (en v)..... voltaje13=');
    D(14)=input(' ingrese la voltaje14 (en v)..... voltaje14=');
end
if x==15
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
    D(7)=input(' ingrese la voltaje7 (en v)..... voltaje7=');
    D(8)=input(' ingrese la voltaje8 (en v)..... voltaje8=');
    D(9)=input(' ingrese la voltaje9 (en v)..... voltaje9=');
    D(10)=input(' ingrese la voltaje10 (en v)..... voltaje10=');
    D(11)=input(' ingrese la voltaje11 (en v)..... voltaje11=');
    D(12)=input(' ingrese la voltaje12 (en v)..... voltaje12=');
    D(13)=input(' ingrese la voltaje13 (en v)..... voltaje13=');
    D(14)=input(' ingrese la voltaje14 (en v)..... voltaje14=');
    D(15)=input(' ingrese la voltaje15 (en v)..... voltaje15=');
end
% ingresamos el ruido que presenta cada antena del sistema
% Si existe ruido adicional al ocasionado por el sistema
```

```
if x==1
    E(1)=input(' ingrese el ruido en la antena1.....ruido1=');
    ruido=E(1);
end

if x==2
    E(1)=input('ingrese el ruido en la antena1.....ruido1==');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    ruido=(E(1)+E(2))/2;
end

if x==3
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    ruido=(E(1)+E(2)+E(3))/3;
end

if x==4
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    ruido=(E(1)+E(2)+E(3)+E(4))/4;
end
if x==5
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    ruido=(E(1)+E(2)+E(3)+E(4)+E(5))/5;
end
if x==6
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6))/6;
end
if x==7
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
```

```
    E(7)=input('ingrese el ruido en la antena7.....ruido7=');
    ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6)+E(7))/7;
end
if x==8
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    E(7)=input('ingrese el ruido en la antena7.....ruido7=');
    E(8)=input('ingrese el ruido en la antena8.....ruido8=');
    ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6)+E(7)+E(8))/8;
end
if x==9
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    E(7)=input('ingrese el ruido en la antena7.....ruido7=');
    E(8)=input('ingrese el ruido en la antena8.....ruido8=');
    E(9)=input('ingrese el ruido en la antena9.....ruido9=');
    ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6)+E(7)+E(8)+E(9))/9;
end
if x==10
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    E(7)=input('ingrese el ruido en la antena7.....ruido7=');
    E(8)=input('ingrese el ruido en la antena8.....ruido8=');
    E(9)=input('ingrese el ruido en la antena9.....ruido9=');
    E(10)=input('ingrese el ruido en la antena10.....ruido10=');
    ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6)+E(7)+E(8)+E(9)+E(10))/10;
end
if x==11
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    E(7)=input('ingrese el ruido en la antena7.....ruido7=');
    E(8)=input('ingrese el ruido en la antena8.....ruido8=');
    E(9)=input('ingrese el ruido en la antena9.....ruido9=');
```

```
    E(10)=input('ingrese el ruido en la antena10.....ruido10=');
    E(11)=input('ingrese el ruido en la antena11.....ruido11=');

ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6)+E(7)+E(8)+E(9)+E(10)+E(11))/11;
end
if x==12
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    E(7)=input('ingrese el ruido en la antena7.....ruido7=');
    E(8)=input('ingrese el ruido en la antena8.....ruido8=');
    E(9)=input('ingrese el ruido en la antena9.....ruido9=');
    E(10)=input('ingrese el ruido en la antena10.....ruido10=');
    E(11)=input('ingrese el ruido en la antena11.....ruido11=');
    E(12)=input('ingrese el ruido en la antena12.....ruido12=');

ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6)+E(7)+E(8)+E(9)+E(10)+E(11)+E(12))/
12;
end
if x==13
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    E(7)=input('ingrese el ruido en la antena7.....ruido7=');
    E(8)=input('ingrese el ruido en la antena8.....ruido8=');
    E(9)=input('ingrese el ruido en la antena9.....ruido9=');
    E(10)=input('ingrese el ruido en la antena10.....ruido10=');
    E(11)=input('ingrese el ruido en la antena11.....ruido11=');
    E(12)=input('ingrese el ruido en la antena12.....ruido12=');
    E(13)=input('ingrese el ruido en la antena13.....ruido13=');

ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6)+E(7)+E(8)+E(9)+E(10)+E(11)+E(12)+E
(13))/13;
end
if x==14
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    E(7)=input('ingrese el ruido en la antena7.....ruido7=');
    E(8)=input('ingrese el ruido en la antena8.....ruido8=');
```

```
    E(9)=input('ingrese el ruido en la antena9.....ruido9=');
    E(10)=input('ingrese el ruido en la antena10.....ruido10=');
    E(11)=input('ingrese el ruido en la antena11.....ruido11=');
    E(12)=input('ingrese el ruido en la antena12.....ruido12=');
    E(13)=input('ingrese el ruido en la antena13.....ruido13=');
    E(14)=input('ingrese el ruido en la antena14.....ruido14=');

ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6)+E(7)+E(8)+E(9)+E(10)+E(11)+E(12)+E
(13)+E(14))/14;
end
if x==15
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    E(7)=input('ingrese el ruido en la antena7.....ruido7=');
    E(8)=input('ingrese el ruido en la antena8.....ruido8=');
    E(9)=input('ingrese el ruido en la antena9.....ruido9=');
    E(10)=input('ingrese el ruido en la antena10.....ruido10=');
    E(11)=input('ingrese el ruido en la antena11.....ruido11=');
    E(12)=input('ingrese el ruido en la antena12.....ruido12=');
    E(13)=input('ingrese el ruido en la antena13.....ruido13=');
    E(14)=input('ingrese el ruido en la antena14.....ruido14=');
    E(15)=input('ingrese el ruido en la antena15.....ruido15=');

ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6)+E(7)+E(8)+E(9)+E(10)+E(11)+E(12)+E
(13)+E(14)+E(15))/15;
end

opcion_ruido=2;
disp('si desea que el ruido adicional vaya a la señal a la entrada
digite 1');
disp(' de lo contrario se considerara que el ruido adicional se
agregara a la señal de referencia');
opcion_ruido=input('opcion_ruido=');
error=input('ingrese el error del sistema=');
% numero de iteraciones del proceso hasta que se cumpla el error
opcion_senal=2;
disp('si desea que la señal de entrada se use como señal de referencia
digite 1');
disp('de lo contrario se considerara otra señal diferente como señal de
referencia');
opcion_senal=input('opcion_señal=');
N=400;
j=0;
% valores recomendado por la central matlab para los pesos en una
funcion
```

```
%
% h=
% vamos a considerar que todas las antenas tienen la misma señal a la
% entrada del filtro por ejemplo  entrada=sin((2*pi*k)/N);
if x==1
    h=[D(1)]
    p=[D(1)]
end
if x==2
    h=[D(1);D(2)]
    p=[(D(1)+D(2))/2]+[0.8*(D(1)+D(2))/2];
end
if x==3
    h=[D(1);D(2);D(3)]
    p=[(D(1)+D(2)+D(3))/3]+[0.8*(D(1)+D(2)+D(3))/3];
end
if x==4
    h=[D(1);D(2);D(3);D(4)]
    p=[(D(1)+D(2)+D(3)+D(4))/4]+[0.8*(D(1)+D(2)+D(3)+D(4))/4];
end
if x==5
    h=[D(1);D(2);D(3);D(4);D(5)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5))/5]+[0.8*(D(1)+D(2)+D(3)+D(4)+D(5))/5];
end
if x==6
    h=[D(1);D(2);D(3);D(4);D(5);D(6)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6))/6]+[0.8*(D(1)+D(2)+D(3)+D(4)+D(5)+D(
6))/6];
end
if x==7
    h=[D(1);D(2);D(3);D(4);D(5);D(6);D(7)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7))/7]+[0.8*(D(1)+D(2)+D(3)+D(4)+D(
5)+D(6)+D(7))/7];
end
if x==8
    h=[D(1);D(2);D(3);D(4);D(5);D(6);D(7);D(8)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8))/8]+[0.8*(D(1)+D(2)+D(3)+D(
4)+D(5)+D(6)+D(7)+D(8))/8];
end
if x==9
    h=[D(1);D(2);D(3);D(4);D(5);D(6);D(7);D(8);D(9)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9))/9]+[0.8*(D(1)+D(2)+D(
3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9))/9];
end
```

```
if x==10
    h=[D(1);D(2);D(3);D(4);D(5);D(6);D(7);D(8);D(9);D(10)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10))/10]+[0.8*(D(1)+
D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10))/10];
end
if x==11
    h=[D(1);D(2);D(3);D(4);D(5);D(6);D(7);D(8);D(9);D(10);D(11)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10)+D(11))/11]+[0.8*
(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10)+D(11))/11];
end
if x==12
    h=[D(1);D(2);D(3);D(4);D(5);D(6);D(7);D(8);D(9);D(10);D(11);D(12)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10)+D(11)+D(12))/12]
+[0.8*(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10)+D(11)+D(12))/
12];
end
if x==13

h=[D(1);D(2);D(3);D(4);D(5);D(6);D(7);D(8);D(9);D(10);D(11);D(12);D(13)
]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10)+D(11)+D(12)+D(13
))/13]+[0.8*(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10)+D(11)+D
(12)+D(13))/13];
end
if x==14

h=[D(1);D(2);D(3);D(4);D(5);D(6);D(7);D(8);D(9);D(10);D(11);D(12);D(13)
;D(14)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10)+D(11)+D(12)+D(13
)+D(14))/14]+[0.8*(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10)+D
(11)+D(12)+D(13)+D(14))/14];
end
if x==15

h=[D(1);D(2);D(3);D(4);D(5);D(6);D(7);D(8);D(9);D(10);D(11);D(12);D(13)
;D(14);D(15)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10)+D(11)+D(12)+D(13
)+D(14)+D(15))/15]+[0.8*(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D
(10)+D(11)+D(12)+D(13)+D(14)+D(15))/15];
end
M=30;
for k=1:400
    entrada(k)=sin((2*pi*k)/M);
```

```
    %senal de entrada
end
for k=1:400
    senal_referencia(k)=cos((2*pi*k)/M);
    % en algunos libros la señal de referencia se la llama señal
deseada
end
entrada';
entrada=entrada';
if opcion_ruido==1
    entrada=entrada+ruido; % aqui agregamos el ruido adicional a la
entrada
end
n=sqrt(0.05)*randn(400,1); % ruido a la entrada del filtro, genera 400
valores
% aleatorios de muestra
[b,a] = butter(2,0.25); % esta funcion calcula los valores de a y b
Gz = tf(b,a,-1); % la funcion de transferencia tf funciona en matlab 7
% esta funcion permite resolver la transformada z.

%Si no se tiene acceso a la caja de herramientas del matlab se puede
cargar valores de
%la central matlab usando el sample data
%load IIRsampledata;
%entrada= IIRsampledata(1:2000);
%d= IIRsampledata(1:2000);

% se puede usar ldiv para hallar las aproximaciones de pesos para un
filtro IIR (filtro de respuesta infinita)
% tambien se puede evaluar  y=h*u
%ldiv es una funcion que nos permite hallar la inversa de la
transformada Z (Matlab central file exchange)
%para hallar los valores de orden de los pesos se puede utilizar la
%siguiente formula
%use h=ldiv(b,a,sysorder)'; ==> here we use sysorder == 10

h=h(1:orden);
y = lsim(Gz,entrada);% simula respuestas arbitrarias de entrada,
relacion la señal Gz
% con la entrada
% para agregar el ruido, ruido adicional que genera el algoritmo rls
n = n * std(y)/(10*std(n));
if opcion_senal==1
senal_referencia = y + n;
end
if opcion_senal>1
    senal_referencia';
    senal_referencia=senal_referencia';
    senal_referencia=senal_referencia+n;
```

```
    %
end
if opcion_ruido>1
    senal_referencia=senal_referencia+ruido; % agrega el ruido
adicional
    % a la salida del filtro
end
totallength=size(senal_referencia,1);% nos da el numero de valores de
la funcion
% de entrada
% conociendo los voltajes de las antenas podemos determinar la potencia
% en ellas
% potencia de la señal a la salida (potencia del sistema
potencia=mean(senal_referencia.^2)
if x==1
% tambien calcularemos la potencia para cada una de las antenas
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
end
if x==2
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
end
if x==3
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2);
% antena2
d2=D(2)+n+E(2);
potencia1=mean(d2.^2);
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2);
end
if x==4
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2);
% antena2
d2=D(2)+n+E(2);
potencia1=mean(d2.^2);
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2);
```

```
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2);
end
if x==5
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
end
if x==6
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
end
if x==7
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
```

```
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
% antena7
d7=D(7)+n+E(7);
potencia7=mean(d7.^2)
end
if x==8
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
% antena7
d7=D(7)+n+E(7);
potencia7=mean(d7.^2)
% antena8
d8=D(8)+n+E(8);
potencia8=mean(d8.^2)
end
if x==9
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
```

```
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
% antena7
d7=D(7)+n+E(7);
potencia7=mean(d7.^2)
% antena8
d8=D(8)+n+E(8);
potencia8=mean(d8.^2)
% antena9
d9=D(9)+n+E(9);
potencia9=mean(d9.^2)
end
if x==10
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
% antena7
d7=D(7)+n+E(7);
potencia7=mean(d7.^2)
% antena8
d8=D(8)+n+E(8);
potencia8=mean(d8.^2)
% antena9
d9=D(9)+n+E(9);
potencia9=mean(d9.^2)
```

```
% antena10
d10=D(10)+n+E(10);
potencia10=mean(d10.^2)
end
if x==11
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=d(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
% antena7
d7=D(7)+n+E(7);
potencia7=mean(d7.^2)
% antena8
d8=D(8)+n+E(8);
potencia8=mean(d8.^2)
% antena9
d9=D(9)+n+E(9);
potencia9=mean(d9.^2)
% antena10
d10=D(10)+n+E(10);
potencia10=mean(d10.^2)
% antena11
d11=D(11)+n+E(11);
potencia11=mean(d11.^2)
end
if x==12
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
```

```
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
% antena7
d7=D(7)+n+E(7);
potencia7=mean(d7.^2)
% antena8
d8=D(8)+n+E(8);
potencia8=mean(d8.^2)
% antena9
d9=D(9)+n+E(9);
potencia9=mean(d9.^2)
% antena10
d10=D(10)+n+E(10);
potencia10=mean(d10.^2)
% antena11
d11=d(11)+n+E(11);
potencia11=mean(d11.^2)
% antena12
d12=D(12)+n+E(12);
potencia12=mean(d12.^2)
end
if x==13
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
% antena7
d7=D(7)+n+E(7);
```

```
potencia7=mean(d7.^2)
% antena8
d8=D(8)+n+E(8);
potencia8=mean(d8.^2)
% antena9
d9=D(9)+n+E(9);
potencia9=mean(d9.^2)
% antena10
d10=D(10)+n+E(10);
potencia10=mean(d10.^2)
% antena11
d11=D(11)+n+E(11);
potencia11=mean(d11.^2)
% antena12
d12=D(12)+n+E(12);
potencia12=mean(d12.^2)
% antena13
d13=D(13)+n+E(13);
potencia13=mean(d13.^2)
end
if x==14
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
% antena7
d7=D(7)+n+E(7);
potencia7=mean(d7.^2)
% antena8
d8=D(8)+n+E(8);
potencia8=mean(d8.^2)
% antena9
d9=D(9)+n+E(9);
potencia9=mean(d9.^2)
% antena10
```

```
d10=D(10)+n+E(10);
potencia10=mean(d10.^2)
% antena11
d11=D(11)+n+E(11);
potencia11=mean(d11.^2)
% antena12
d12=D(12)+n+E(12);
potencia12=mean(d12.^2)
% antena13
d13=D(13)+n+E(13);
potencia13=mean(d13.^2)
% antena14
d14=D(14)+n+E(14);
potencia14=mean(d14.^2)
end
if x==15
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
% antena7
d7=D(7)+n+E(7);
potencia7=mean(d7.^2)
% antena8
d8=D(8)+n+E(8);
potencia8=mean(d8.^2)
% antena9
d9=D(9)+n+E(9);
potencia9=mean(d9.^2)
% antena10
d10=D(10)+n+E(10);
potencia10=mean(d10.^2)
% antena11
d11=D(11)+n+E(11);
potencia11=mean(d11.^2)
```

```
% antena12
d12=D(12)+n+E(12);
potencia12=mean(d12.^2)
% antena13
d13=D(13)+n+E(13);
potencia13=mean(d13.^2)
% antena14
d14=D(14)+n+E(14);
potencia14=mean(d14.^2)
% antena15
d15=D(15)+n+E(15);
potencia15=mean(d15.^2)
end
% calculo de la cota de la contante de ajuste
cota=1/((orden+1)*potencia);
%evalua 70 puntos por corrida ( N - orden 70 = 80 - 10 )
N=80;   % N=80
%begin of the algorithm
%forgetting factor valores recomendados por la biblioteca matlab
lamda = 0.999994115 ;% lamda= 0.9995
%initial P matrix
delta=1000000000000000 % delta = 1e10 ;
P = delta * eye (orden);
% w = zeros ( orden  , 1 )
if x==1
    w=[0.15]
    w=w'
end
if x==2
    w=[ 0.15 0.10]
    w=w'
end
if x==3
    w=[0.15 0.10 0.20 ]
    w=w'
end
if x==4
    w=[0.15 0.10 0.20 0.30]
    w=w'
end
if x==5
    w=[0.15 0.10 0.20 0.30 -0.20]
    w=w'
end
if x==6
    w=[0.15 0.10 0.20 0.30 -0.20 0.14]
    w=w'
end
if x==7
```

```
    w=[0.15 0.10 0.20 0.30 -0.20 0.14 0.12]
    w=w'
end
if x==8
    w=[0.15 0.10 0.20 0.30 -0.20 0.14 0.12 0.11]
    w=w'
end
if x==9
    w=[0.15 0.10 0.20 0.30 -0.20 0.14 0.12 0.11 0.05]
    w=w'
end
if x==10
    w=[0.15 0.10 0.20 0.30 -0.20 0.14 0.12 0.11 0.05 -0.05]
    w=w'
end
if x==11
    w=[0.15 0.10 0.20 0.30 -0.20 0.14 0.12 0.11 0.05 -0.05 -0.09]
    w=w'
end
if x==12
    w=[0.15 0.10 0.20 0.30 -0.20 0.14 0.12 0.11 0.05 -0.05 -0.09 0.05]
    w=w'
end
if x==13
    w=[0.15 0.10 0.20 0.30 -0.20 0.14 0.12 0.11 0.05 -0.05 -0.09 0.05
0.08]
    w=w'
end
if x==14
    w=[0.15 0.10 0.20 0.30 -0.20 0.14 0.12 0.11 0.05 -0.05 -0.09 0.05
0.08 0.06]
    w=w'
end
if x==15
    w=[0.15 0.10 0.20 0.30 -0.20 0.14 0.12 0.11 0.05 -0.05 -0.09 0.05
0.08 0.06 0.02]
    w=w'
end
% T10=[D(1) D(2) D(3) D(4) D(5)]
Rp =1;
for n = orden : N
    u = entrada(n:-1:n-orden+1)
    phi = u' * P ;
    k = phi'/(lamda + phi * u );
    y(n)= w'*u ;
    y(n)
    T(n)=[abs(y(n))].^2
    T2(n)=Rp-T(n); % Rp valor de la constante cma, si no converge,
                   % pruebe con otros valores de Rp
```

```
    T3=T2(n)*u
    e=T3*y(n) % e(n) = senal_referencia(n) - y(n); error para RLS
    % e(n)=senal_referencia(n)-y(n);
    T4=e'
    T5=T4*u
    mu=5*10^-7
    w=(w)-(mu*T5)  % w = (w) + k * e(n);  peso para RLS
    % si no converg utilice e* conjugada;
    P = ( P - k * phi ) / lamda;
    % ajuste del ploteo, pesos
    Recordedw(1:orden,n)=w
end
%check of results
for n =  N+1 : totallength
    u = entrada(n:-1:n-orden+1) ;
    y(n) = w' * u ;
    T(n)=[abs(y(n))].^2;
    T2(n)=1-T(n);
    e(n)=T2(n)*y(n);    %e(n) = senal_referencia(n) - y(n);rls
    % e(n)=e(n)*u;
end
hold on
plot(senal_referencia)
plot(y,'r');
title('salida del sistema ') ;
xlabel('numero de muestras')
ylabel('valor estimado a la salida')
figure
semilogy((abs(e))) ;
title('curva de error') ;
xlabel('numero de muestras');
ylabel('valor del error');
figure
plot(Recordedw(1:orden,orden:N)');
title('convergencia de los pesos estimados') ;
xlabel('numero de muestras');
ylabel('valor de los pesos');
axis([1 N-orden min(min(Recordedw(1:orden,orden:N)'))
max(max(Recordedw(1:orden,orden:N)')) ]);
hold off
figure
plot(h, 'k+')
hold on
plot(w, 'r*')
legend('pesos actuales','pesos estimados');
title('Comparacion de los pesos actuales con los pesos estimados') ;
```

5.4 Descripción del ejemplo a correr en el programa.

Ingresaremos 2 curvas, la una será la señal de entrada:

entrada(k) = sin((2*pi*k)/M) y la segunda será la señal de referencia: señal_referencia(k) = cos((2*pi*k)/M) ingresaremos el orden del filtro, y el número de valores reales (numero de antenas), el programa tiene una restricción al ingresar el orden del canal tiene que ingresarse ese mismo número como número de antenas, también ingresaremos la señal de transferencia del filtro, también ingresaremos el error deseado. en el algoritmo cma al igual que en el algoritmo lms y rls anteriores podremos también ingresar un ruido adicional a las antenas, escogiéndose si se quiere agregar el ruido a la entrada o a la salida del programa.

También hicimos un programa problema1 que utiliza la misma función senoidal de entrada como función de referencia, y un programa problema2 que utiliza señales gaussiana aleatorias como señal de entrada y realizamos algunas simulaciones hicimos esto tanto para el algoritmo LMS como para el algoritmo RLS, estos los colocaremos en el anexo F si se los quiere revisar.

5.5 Pruebas y resultados de las implementaciones CMA.

En este programa problema3_cma hemos colocado una señal de entrada(k)=sin((2*pi*k)/M); para n=200 muestras, y constante (u)cma=5*10^-7. Se utilizo un algoritmo cma-rls hibrido, se utiliza una señal de referencia igual a la señal de entrada ingresada en una función de transferencia (y(n)), más un ruido añadido por el programa rls, este ruido adicional es del orden de 2, 3 y hasta un 10 por ciento. El programa tiene una restricción al ingresar el numero de orden del canal, tiene que colocarse el mismo número de antenas. El programa genera este ruido en forma aleatoria.

Para todos los ejemplos que utilicen el programa problema3_cma son validos los siguientes objetivos: demostrar que se pueden utilizar funciones senosoidales en algoritmos cma, que el algoritmo cma es parecido al algoritmo rls y utiliza el filtro de kalman, ver si el algoritmo cma converge más o menos veces que los algoritmos lms y rls, necesitamos saber esto porque nos permitirá conocer si es más o menos confiable que los otros dos algoritmos. Comparar los resultados de las funciones senosoidales dados por el programa con los valores reales ingresados por el usuario. Hacer un grafico de esta comparación, hacer un grafico del error y un grafico de comparación entre la señal de referencia y la señal obtenida por la simulación. También determinar que las

señales senosoidales pueden aplicarse en los algoritmos cma al igual como se aplican las señales gausianas. Otros objetivos de la presente corrida es ver la correlación del número de muestras con la convergencia de la señal, también con cuantas muestras se obtiene la mejor comparación de las señales de salida del filtro con la señal de referencia de la señal senosoidal, también se consideraría como otro objetivo del presente programa que este tipo de datos obtenidos en forma experimental podrían darnos valores aproximados de antenas reales y evitar construir un sistema costoso de antenas para obtener este tipo de valores de prueba, esto podría bajar costos a alguna empresa que trabaje con señales senosoidales en condiciones parecidas al presente programa.

Desarrollo: En la corrida ingresaremos un filtro de orden = 5, a continuación el numero de antenas (n=5), con los siguientes valores de voltaje v1=0.10 V, v2=0.12 V, v3=0.2 V, v4=0.25 V, v5=-0.2 V. ingresaremos el ruido de las antenas con valores de ruido1=0.01 V, ruido2=0.01 V, ruido3=0.01V, ruido4=0.02 V y ruido5= 0.02V. y rango de seguridad = 40

A continuación hay una opción para elegir colocar este ruido adicional a la entrada del filtro (opción_ruido=1) o a la salida del filtro (opción_ruido=2) En

caso que no lo escojamos automáticamente agregara este ruido adicional a la salida del filtro. Es decir el ruido adicional se suma a la señal de referencia. En el presente ejemplo (problema3_cma_a) escogeremos la opción 1 para colocar el ruido adicional a la entrada del filtro.

A continuación ingresaremos el error del sistema = 0.0000001. después de esto nos saldrá el mensaje que debemos ingresar la opción_señal (la opción_señal si es igual a 1 escogera como señal de referencia a la misma señal de entrada) ,(en caso de escoger opción_señal=2 se escogerá una señal diferente como señal de referencia), esta señal debe de tener una alta correlación o similitud con la señal de entrada, cada una de estas opciones tiene ciertas ventajas, según el tipo de programa y modelo que se esté utilizando.

El programa al correr determinara la potencia del sistema y luego la potencia de cada una de las antenas. Luego evaluara los gráficos de comparación de pesos (comparara los valores reales, ingresados por teclado (los valores de las antenas (con negro)) con los valores evaluados por el programa (color rojo). Sera un grafico amplitud señal (V) vs B, donde B es un valor constante, Es decir en el eje de las y estará el valor de la amplitud (en V) vs un valor de B=6 en el eje de las x, en este valor de 6 estarán repartidas las muestras evaluadas por el programa, también se hará una grafica de el error en función de la amplitud de la señal vs el

número de muestras. Y al final un grafico de comparación de la señal de referencia con la señal del sistema a la salida. En amplitud de la señal vs el numero de muestras (la señal de referencia es la señal azul, y la señal a la salida del programa es la roja). Utilizaremos los siguientes datos:

TABLA LII Datos de entrada para el Problema3_cma_a

Tipo de señal utilizada: senosoidal

Algoritmo utilizado: cma

Aplicación en matlab utilizado: problema3_cma

Nombre dado al ejemplo: problema3_cma_a

Orden de la señal: 5

Numero de antenas: 5

Valores iniciales de entrada para las antenas:

V1= 0.10 V, V2=0.12 V, V3=0.2 V, V4=0.25 V, V5=-0.2 V

Valores de ruido ingresados para las antenas:

Ruido1= 0.01 V, ruido2=0.01 V, ruido3= 0.01 V, ruido4=0.02 V, ruido5=0.02 V

Opción_ruido=1, opción_señal=1

numero de muestras =400, rango de seguridad=40

constante de ajuste (u)cma=5*10^-7

Error del sistema = 0.0000001

TABLA LIII Datos obtenidos por el ejemplo Problema3_cma_a

Pesos reales:	0.10V	0.12V	0.2V	0.25V	0.2V
Pesos obtenidos cma:	0.15V	0.1V	0.2V	0.3V	-0.2V

Para ver las pantallas de visualización de las potencias del sistema, señal obtenida a la salida, comparación de señales y curvas de error del presente ejemplo ir al anexo F ejemplo problema3_cma_a.

TABLA LIV Datos de entrada para el Problema3_cma_b

Tipo de señal utilizada: senosoidal

Algoritmo utilizado: cma

Aplicación en matlab utilizado: problema3_cma

Nombre dado al ejemplo problema3_cma_b

Orden de la señal: 4

Numero de antenas: 4

Valores iniciales de entrada para las antenas:

V1= 0.15 V, V2=0.12 V, V3=0.05 V, V4=-0.10 V

Valores de ruido ingresados para las antenas:

Ruido1= 0.01 V, ruido2=0.01 V, ruido3= 0.02 V, ruido4=0.02 V

Opción_ruido=1, opción_señal=2

numero de muestras=200, rango de seguridad=80

constante de ajuste (u)cma=5*10^-7

Error del sistema = 0.0000001

TABLA LV Datos obtenidos por el ejemplo Problema3_cma_b

Pesos reales:	0.15V	0.12V	0.05V	0.10V
Pesos obtenidos cma:	0.15V	0.1V	0.2V	0.3V

Para ver las pantallas de visualización de las potencias del sistema, señal obtenida a la salida, comparación de señales y curvas de error del presente ejemplo ir al anexo F ejemplo problema3_cma_b.

Ahora utilizaremos el programa2_cma, en este programa se utiliza un algoritmo hibrido cma-rls, como señal de entrada y como señal de referencia se utilizara una señal gaussiana aleatoria, con valores de N=400 muestras y rango de seguridad de N=80, con una constate de ajuste cma de 5*10^-7

Para todos los ejemplos que utilicen el programa problema2_cma son validos los siguientes objetivos: demostrar que se pueden utilizar funciones gaussianas en algoritmos cma, que el algoritmo cma es parecido al algoritmo rls y utiliza el filtro de kalman, ver si el algoritmo cma converge más o menos veces que los algoritmos lms y rls, necesitamos saber esto porque nos permitirá conocer si es más o menos confiable que los otros dos algoritmos. Comparar los resultados de las funciones gaussianas dados por el programa con los valores reales ingresados por el usuario. Hacer un grafico de esta comparación, hacer un grafico del error y un grafico de comparación entre la señal de referencia y la señal obtenida por la simulación. Otros objetivos de la presente corrida es ver la correlación del número de muestras con la convergencia de la señal, también con cuantas muestras se obtiene la mejor comparación de las señales de salida del filtro con la señal de referencia de la señal senosoidal, también se consideraría como otro objetivo del presente programa que este tipo de datos obtenidos en forma experimental podrían darnos valores aproximados de

antenas reales y evitar construir un sistema costoso de antenas para obtener este tipo de valores de prueba, esto podría bajar costos a alguna empresa que trabaje con señales gaussianas en condiciones parecidas al presente programa.

TABLA LVI Datos de entrada para el Problema2_cma_a

Tipo de señal utilizada: gaussiana

Algoritmo utilizado: cma

Aplicación en matlab utilizado: problema2_cma

Nombre dado al ejemplo: problema2_cma_a

Orden de la señal: 5

Numero de antenas: 5

Valores iniciales de entrada para las antenas:

V1= 0.15 V, V2=0.18 V, V3=0.20 V, V4=0.12 V, V5=0.14 V

Valores de ruido ingresados para las antenas:

Ruido1= 0.01 V, ruido2=0.01 V, ruido3= 0.01 V, ruido4=0.02 V, ruido5=0.02 V

Opción_ruido=1

numero de muestras =400, rango de seguridad=80

constante de ajuste (u)cma=5*10^-7

Error del sistema = 0.0000001

TABLA LVII Datos obtenidos por el ejemplo Problema2_cma_a

Pesos reales:	0.15V	0.18V	0.20V	0.12V	0.14V
Pesos obtenidos cma:	0.02V	0.02V	0.03V	0.04V	-0.02V

Para ver las pantallas de visualización de las potencias del sistema, señal obtenida a la salida, comparación de señales y curvas de error del presente ejemplo ir al anexo F ejemplo problema2_cma_a.

con valores gaussianos el error es muy alto en el algoritmo cma, la señal tiende a no converger. El algoritmo cma no funciona correctamente con señales gaussianas. En conclusión el algoritmo cma con señales gaussianas tiende a no converger.

TABLA LVIII Datos de entrada para el Problema1_cma_a

Tipo de Señal utilizada: senosoidal

Algoritmo utilizado: cma

Aplicación en matlab utilizado: problema1_cma

Nombre dado al ejemplo problema1_cma_a

Orden de la señal: 5

Numero de antenas: 5

Valores iníciales de entrada para las antenas:

V1= 0.15 V, V2=0.18 V, V3=0.20 V, V4=0.12 V, V5=0.14 V

Valores de ruido ingresados para las antenas:

Ruido1= 0.01 V, ruido2=0.01 V, ruido3= 0.01 V, ruido4=0.02 V, ruido5=0.02 V

Opción_ruido=1

numero de muestras = 400, rango de seguridad = 80

constante de ajuste (u)cma=5*10^-7

Error del sistema = 0.0000001

TABLA LIX Datos obtenidos por el ejemplo Problema1_cma_a

Pesos reales:	0.15V	0.18V	0.20V	0.12V	0.14V
Pesos obtenidos cma:	0.15V	0.1V	0.19V	0.3V	-0.2V

Para ver las pantallas de visualización de las potencias del sistema, señal obtenida a la salida, comparación de señales y curvas de error del presente ejemplo ir al anexo F ejemplo problema1_cma_a

con una señal senosoidal la señal salió mejor que con valores gaussianos, en este caso si convergió. debe de ser alguna cualidad del algoritmo cma, aun así en algunas simulaciones que hicimos en el algoritmo cma converge menos veces que en el lms, rls y dmi.

Nota.- Algo interesante de anotar en la señal obtenida por el algoritmo cma con señales senosoidales es que algunos casos la señal obtenida fue un poco menos de la mitad de la señal de referencia, es decir que se tendría algo de pérdida de potencia, aunque la señal si converge, y puede ser útil en algunos casos, ya que la señal es filtrada correctamente, un ejemplo de esto se lo puede ver en el ejemplo problema3_cma_x.

En el algoritmo dmi se presento algo parecido a lo ocurrido en el algoritmo cma en ciertos tramos de la curva, en algunas partes de la curva obtenida por el algoritmo dmi la señal también disminuía y en otras no, esto está relacionado con las zonas de control de convergencia, cambiando la zona de control de convergencia se obtiene una curva que no perdía señal en estos tramos. Esto lo podemos ver en el ejemplo problema3_dmi_x.

CAPÍTULO 6

EL ALGORITMO ADAPTATIVO DE INVERSIÓN DE MATRIZ DIRECTA (DMI)

El algoritmo adaptativo de inversión de matriz directa (DMI) es un algoritmo de igualación no ciega de canal, también llamado algoritmo de matriz inversa, suele utilizarse como un algoritmo de búsqueda, en medicina es utilizado en redes neuronales al igual que el algoritmo cma.

El algoritmo dmi tiene la característica de que utiliza el filtro de wiener dentro de su programación, por lo que puede ser utilizado en ciertos casos para funciones lineales, su estructura es parecida al de algoritmo lms, converge más rápidamente que el algoritmo lms, y requiere señal de referencia, porque puede perder fácilmente la señal, se suele usar este algoritmo también en programación unix, en programación de juegos por computadora, juegos de ajedrez , es usado para determinar el comportamiento de las células en biología y en telecomunicaciones.

Los primeros en investigar este algoritmo fueron Dailei y Blight en 1991 y por Gangass en 1994, los cuales encontraron modos de utilizarlo en circuitos eléctricos.utilizando lazos, compuertas y condiciones matemáticas.

6.1 El Algoritmo DMI

La solución mínimo cuadrado (capitulo 3) , la solución mínimo cuadrado recursivo (capitulo 4) necesitan muchas iteraciones para llegar a su convergencia (el algoritmo lms varia de 200, 400, 800 y hasta 2000 iteraciones), en el algoritmo rls puede variar de 20, 80, 100, 200 y hasta 400 iteraciones según el grado de error que se quiera implementar, el algoritmo cma-rls realizado en la presente tesis tuvo una convergencia más rápida, y un desempeño casi tan bueno como el del algoritmo lms, pero aun así el algoritmo lms llego a una convergencia en una mayor cantidad de pruebas, el algoritmo dmi, al utilizar el filtro de wiener es probable que tenga un desempeño casi tan bueno como el del algoritmo lms, pero con una mayor rapidez de operación, un peligro que se tiene con el algoritmo dmi es que por su rápida convergencia pueda hacer que la señal se pierda y no se halle la convergencia en algunos casos.

El algoritmo DMI también es conocido como algoritmo de matriz inversa , es utilizado ampliamente en la teoría de juegos, y usado para determinar el comportamiento de las células en biología, por eso también es usado en medicina.

Para entender más sobre el funcionamiento del algoritmo dmi revise lo relativo al filtro de wiener en el capítulo 3.2 y el 6.1.2 del presente capítulo .

6.1.1 Base teórica del algoritmo adaptativo Inversión de Matriz Directa

El algoritmo dmi analiza variables en un lazo de transferencia de una matriz, la matriz puede ser del tipo N por N, también puede aceptar valores negativos, puede ser usado con lazos simples o con lazos estructurados, en función de tiempo, fase, e inclusive para canales. Pueden ser utilizados en varias puertas en diferentes canales, para determinadas fases, y para garantizar su convergencia utiliza una función de referencia, puede trabajar en múltiples canales en forma simultaneas independiente, en algunos casos se los calcula en zonas de seguridad para garantizar la convergencia. Algunos utilizan el algoritmo dmi en función de la frecuencia en db o rad/s o en función de fase en grados. La arquitectura del algoritmo dmi se puede ver en la fig 6.1 y como son utilizadas en antenas mimo en la fig 6.2

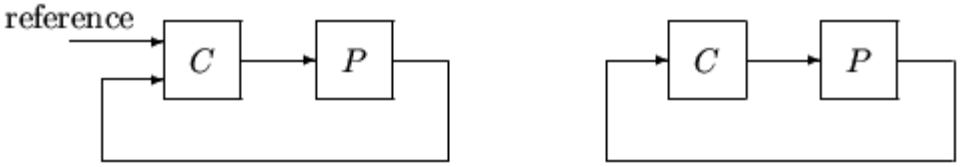


fig 6.1 Arquitectura de un lazo de control del algoritmo dmi

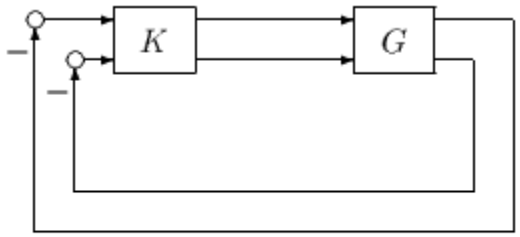


fig 6.2 multi entrada y multi salida de un sistema mimo
G y k son 2 multi entradas y multi salidas mimo

6.1.2 Análisis matemático del algoritmo adaptativo de Inversión de Matriz Directa DMI

El algoritmo DMI es un algoritmo que utiliza matrices en un lazo de control, puede usar multicanales, en función de frecuencia, fase, etc. Se requiere una mayor complejidad computacional que el del algoritmo lms, sus formulas principales son:

$$\hat{R}_{xx} = \sum_{i=N_1}^{N_2} x(i)x^H(i) \quad (6.1)$$

$$\hat{r}_{xd} = \sum_{i=N_1}^{N_2} d^*(i)x(i) \quad (6.2)$$

$$w = \hat{R}_{xx}^{-1}\hat{r}_{xd} \quad (6.3)$$

El algoritmo dmi utiliza las formulas de Rxx de la correlacion cruzada, Pero evaluada entre dos valores N1 y N2, donde N1 y N2 son los limites de iteraciones en donde se evaluara la sumatoria de Rxx que será uno de los valores que nos permitirá obtener los pesos en el algoritmo dmi, este valor de Rxx a su inversa se la multiplica por el valor de rxx.

Donde rxx sera igual a d(n)*u, d(n) es la señal de referencia, y u es la señal de entrada al filtro evaluada en el intervalo N1,N2.

La ecuación para hallar Rxx ya lo vimos en el capítulo 4 en la ecuación 4.53 y 4.54.

$$\mathbf{W}^* = \mathbf{R}_{XX}^{-1} \cdot \mathbf{R}_{Xd} = \begin{bmatrix} \dfrac{2 \cdot \cos\left(\frac{2\pi}{N}\right) \cdot \operatorname{sen}\left(\frac{2\pi}{N}\right)}{\left(1 + E\left[n[k]^2\right]\right)^2 - \cos^2\left(\frac{2\pi}{N}\right)} \\ \dfrac{-2 \cdot \left(1 + 2 \cdot E\left[n[k]^2\right]\right) \cdot \operatorname{sen}\left(\frac{2\pi}{N}\right)}{\left(1 + E\left[n[k]^2\right]\right)^2 - \cos^2\left(\frac{2\pi}{N}\right)} \end{bmatrix} \quad \textbf{(4.53)}$$

$$\xi_{\min} = \xi_{\min} = E\left[d_k^2\right] - \mathbf{W}^{*T}\,\mathbf{R}_{XX}\mathbf{W}^* - 2\mathbf{R}_{Xd}^T\mathbf{W}^* = E\left[d[k]^2\right] - \mathbf{R}_{Xd}^T \cdot \mathbf{W}^* \quad \textbf{(4.54)}$$

El lazo de evaluación de pesos quedaría asi:

```
for n = orden : N
    u = entrada(n:-1:n-orden+1);
T=u*u';
    if j==1
        Rxx=T*0;
    end
        Rxx=Rxx+T;
    T2=senal_referencia(n)*u;
    if j==1
```

```
        rxx=T2*0;
    end
        rxx=rxx+T2;
    w = (inv(Rxx))*rxx;  % ecuación de pesos para el
algoritmo dmi
end
```

u son los valores de la ecuación de entrada evaluada entre el límite

superior N y el límite inferior orden

orden es el numero de niveles del canal

Rxx es la sumatoria de u*u' entre N y orden

rxx es la sumatoria de senal_referencia(n)*u entre N y orden

W son los pesos en el algoritmo dmi

El lazo de chequeo de la curva queda igual que en el algoritmo lms.

```
% chequeo de resultados de error
N=200
for n =  N+1 : totallength
    u = entrada(n:-1:n-orden+1) ;
```

```
    y(n) = w' * u ;   % y(n) señal a la salida
    % e(n) = senal_referencia(n) - y(n) ;
    e(n)=senal_referencia(n)-y(n);
end
```

Hay que tener presente que en nuestro programa DMI no utilizamos el valor de la ganancia como en el algoritmo RLS y en el algoritmo CMA, pero la ganancia si se suele usar para antenas SISO al utilizar el algoritmo dmi y su fórmula es la ecuación 6.4

La ganancia para antenas SISO está dada por la formula 6.4

$$G: = \frac{1}{s^2 + \alpha^2}\begin{bmatrix} s-\alpha^2 & \alpha(s+1) \\ -\alpha(s+1) & s-\alpha^2 \end{bmatrix}, K=I_2 \qquad \textbf{(6.4)}$$

En donde alfa es una constante que se reemplaza en la formula de G(s), la variación de la fase y la garantía de su funcionamiento en una región está dado por la figura 6.3.

The guaranteed region of phase and gain variations for the closed-loop system can be illustrated graphically. The disk margin analysis, `dmi(2)`, indicates the closed-loop system will remain stable for simultaneous gain variations of 0.475 and 2.105 (± 6.465 dB) and phase margin variations of ± 39.18 degs in the second input channel. This is denoted by the region associated with the large ellipse in the following figure. The multivariable margin analysis at the input to the plant, `mmi`, indicates that the closed-loop system will be stable for independent, simultaneous, gain margin variation up to 0.728 and 1.373 (±2.753 dB) and phase margin variations up to ± 17.87 degs (the dark ellipse region) in both input channels.

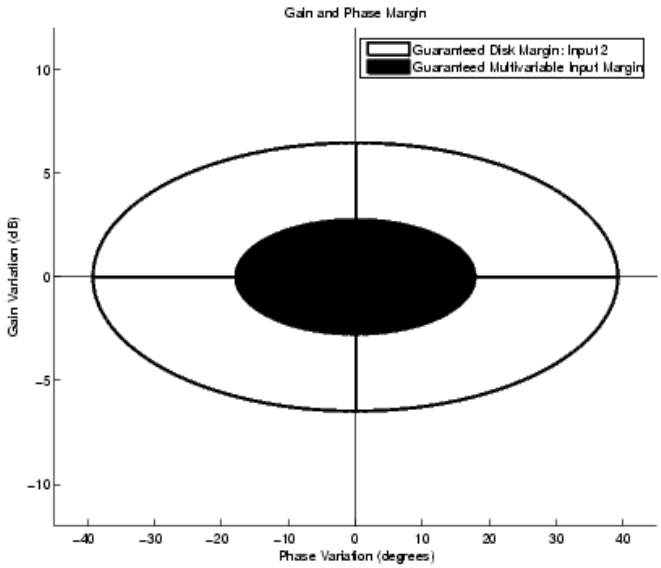


fig 6.3 Region de garantia de fase de un algoritmo dmi

Gráfico tomado de la central matlab, documentación preliminar para el analisis del algoritmo dmi. uso de herramientas de trabajo funciones y cajas

6.1.3 Variaciones con el algoritmo DMI

Analizaremos una variación con el algoritmo DMI. El método de Daily and Gangass. Existen también otras variaciones del algoritmo DMI, una de ellas es un desarrollo para determinar las células cerebrales (neuronas) y el desarrollo del cerebro, incluyendo un programa en dmi que determina el desarrollo de algunos tipos de cáncer cerebral, también leímos en un paper de un uso del algoritmo dmi para determinar el desarrollo de las células, como eran estudios médicos no aportaban información ni formulas sobre el algoritmo dmi. Solo hacían referencia que lo usaban con estas aplicaciones.

El Metodo de Blight, Daily and Gangass.

Fue desarrollado en 1994, como una variante y aplicación del algoritmo DMI, se lo utiliza en circuitos eléctricos, en donde ellos conociendo dos caminos o lazos en los cuales en uno de ellos se evalúa una señal de entrada, y señales de ruido, que se basan en la sensibilidad de la función S=(1-L)-1 , que es complementada por la formula T= L (1-L)-1 en donde L es un lazo o camino hacia una compuerta asociada a una matriz, y donde se forma un sistema de ecuaciones eléctrico como se ve en la figura 6.4.

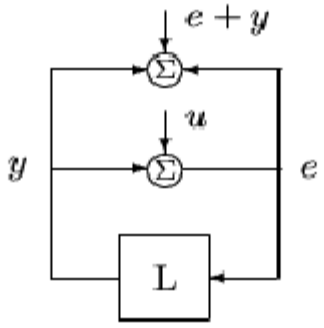


fig 6.4 Diagrama electrico de una aplicacion del algoritmo dmi de Blight, Daily and Gangass.

En donde S es la matriz transformada junto con entradas u y salidas, , en este algoritmo se debe cumplir máximos y mínimos, como condiciones de funcionamiento, las principales formulas que utiliza el método de Blight, Daily and Ganggas son:

$$e = (I-L)^{-1}u = Su \quad \textbf{(6.5)}$$

$$y = L(I-L)^{-1}u = Tu \quad \textbf{(6.6)}$$

$$e+y = (I+L)\cdot(I-L)^{-1}u = (S+T)u \quad \textbf{(6.7)}$$

6.2 Pantallas de visualización del algoritmo adaptativo inversión de matriz directa DMI

Al hacer correr cualquiera de las aplicaciones en matlab del algoritmo inversión de matriz directa DMI que hemos realizado se presentaran algunas pantallas de visualización, algunas de estas pantallas son para información de los usuarios a usar el programa, otras son para ingresar valores de entrada, las ultimas son las generadas por el programa matlab y son pantallas de salida, con estos datos de entrada y salida formaremos tablas de datos y resultados obtenidos. Todas estas pantallas servirán tanto para señales senosoidales como para señales gaussianas. También podrían servir para otros tipos de señales que cumplan las características señaladas por wiener para filtros adaptativos.

Las pantallas a utilizar en el programa del algoritmo adaptativo DMI son:

Pantalla de presentación:

En esta pantalla se hace una explicación al usuario de las características de la aplicación y lo que resuelve, está el nombre de la universidad, los nombres de los programadores del programa, el profesor guía, el tipo de señal a usar en el programa, , una pequeña explicación de los datos de entrada y los de salida.

Pantalla de ingreso de datos:

En esta pantalla ingresamos el numero de antenas a utilizar (en el algoritmo DMI de 1 a 5). los valores iníciales de las antenas, los valores de ruido de las antenas.

En esta pantalla aparecerá una opción llamada opción ruido que si es igual a 1 el ruido se ingresara a la entrada del sistema (antes del filtro), si se escoge 2 el ruido se agregara a la salida del sistema. También se ingresara un valor de error del sistema

Pantalla de potencias obtenidas:

Esta pantalla nos dará la potencia del sistema y el valor de la potencia de cada una de las antenas.

Pantalla de comparación de pesos estimados:

En esta pantalla presentaremos los datos iníciales ingresados por teclado (los llamaremos también valores actuales) los representaremos por medio de una cruz negra, y los compararemos con los valores obtenidos por el algoritmo DMI, este valor obtenido lo llamaremos valores estimados, y lo representaremos con un color rojo.

Pantalla de valor de la curva de error:

En esta pantalla se muestra el comportamiento de la ecuación de error del algoritmo DMI, esta nos servirá para compararla con el comportamiento de otros algoritmos adaptativos.

Pantalla de comparación entre la salida del sistema con la señal de referencia:

Esta pantalla es una de las más importantes, aparecerá la señal que obtenemos a la salida del sistema (color rojo) , y se la comparara con la señal de referencia (color azul).

En el anexo F se muestran las pantallas de visualización que obtendremos en nuestras implementaciones al utilizar diferentes valores de entrada y diferentes casos en cada uno de los algoritmos utilizados.

6.3 Programa en matlab del algoritmo adaptativo inversión de matriz directa DMI

Al hacer el programa en matlab del algoritmo DMI, decidimos utilizar como base la programación del algoritmo LMS, en el cual ya se utiliza el filtro de wiener que es utilizado por el algoritmo DMI, a este algoritmo se le agregara las ecuaciones del algoritmo DMI, principalmente su ecuación de pesos, este algoritmo podría considerárselo un hibrido DMI-LMS. Al igual que en el programa LMS utilizamos una función principal que será llamada desde el programa matlab, esto permitirá ciertas ventajas ya que todo el programa está en una sola función.

Con el orden del filtro determinaremos los valores de cota, los cuales nos servirán para hallar la potencia de la señal. Realizaremos 3 programas dmi, en el primero se utilizara la señal de entrada también como señal de referencia con una señal senosoidal, al cual se le deberá agregar un ruido aleatorio causado por el sistema dmi (el algoritmo dmi internamente produce un ruido que puede ser del 2, 3 y hasta 10% en el peor de los casos), se le podrá elegir si se desea aplicar estos ruidos adicionales a la entrada o a la salida del filtro, existirá una opción señal en la cual uno dictamina si desea utilizar como señal de referencia una curva diferente a la señal de entrada, esta señal de referencia auxiliar es del tipo senosoidal si la una es del tipo seno, la otra es del tipo coseno.

También hemos hecho un programa problema2_dmi en donde se utilizan señales gaussianas, como señales de entrada y como señales de referencia, y por ultimo un programa problema1_dmi en donde se evalúa solamente una señal de entrada que se la usa como señal de referencia y con ruidos adicionales (además del ruido del sistema, este programa la función de entrada es del tipo senosoidal).

Programa Principal y función principal

```
function[caratula]=problema3_dmi;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
% Algoritmo DMI
% Autores: Juan Alvarez , Maribel Chuez
%
%Escuela Superior Politecnica del Litoral
%Facultad de Electricidad y Computación
%Guayaquil, Ecuador
% email :juan_alvarez8@latinmail.com
% email : mchuez@ceibo.espol.edu.ec
% Webpage : proximamente
% Date    : Domingo 26 de Julio del 2009
% Version : 1.0.1
% Referencias : S. Haykin, Adaptive Filter Theory. 3rd edition, Upper
Saddle River, NJ: Prentice-Hall, 1996.
%
%clear all % esta funcion borra los valores anteriores que esten
memoria %
%close all % esta funcion borra cualquier ventana anterior activa %
%hold off   % esta funcion encera cualquier grafico que estuviera
activo %
 fprintf('*****************************************************.\n')
fprintf('*    ESCUELA SUPERIOR POLITECNICA DEL LITORAL        *.\n')
fprintf('*                                                    *.\n')
fprintf('* PROBLEMA3_dmi.- Determinar los pesos, el numero de *.\n')
fprintf('* iteraciones y la potencia de un grupo de antenas   *.\n')
fprintf('* inteligentes utilizando el algoritmo DMI (algoritmo*.\n')
fprintf('* adaptativo de inversión de matriz directa, las     *.\n')
fprintf('* antenas están distribuidas en forma de celdas      *.\n')
fprintf('* para lo cual requeriremos de los siguientes datos: *.\n')
fprintf('*                                                    *.\n')
fprintf('*  • la señal aplicada a la entrada del filtro,      *.\n')
fprintf('*   ej  Xn=sin((2*pi*k)/N)                           *.\n')
fprintf('*  • el numero de antenas  x                         *.\n')
fprintf('*  • la señal de referencia, d(k)=cos((2*pi*k)/N);   *.\n')
fprintf('*  • el ruido de las diferentes antenas, (hay que    *.\n')
fprintf('*  tener presente que el algoritmo CMA en su         *.\n')
fprintf('*  estructura internamente produce un ruido que puede*.\n')
fprintf('*  ser de 2, 3 y hasta 10 %  aparte de este ruido    *.\n')
fprintf('*  generado por el mismo programa CMA se puede       *.\n')
fprintf('*  ingresar ruidos adicionales en el programa a la   *.\n')
fprintf('*  entrada o a la salida del programa.  n(f)         *.\n')
fprintf('*  • el orden del filtro FIR cuyos coeficientes se   *.\n')
```

```
fprintf('*  van a ajustar, orden                                  *.\n')
fprintf('*  • el número de iteraciones en el aprendizaje, N      *.\n')
fprintf('*  (este número se hará  hasta que cumpla el error      *.\n')
fprintf('*  • la constante de ajuste   u                         *.\n')
fprintf('*  • el vector de pesos inicial. Los voltajes de las    *.\n')
fprintf('*  antenas V(f)                                         *.\n')
fprintf('*  • el error deseado puede ser ingresado por teclado   *.\n')
fprintf('*    error                                              *.\n')
fprintf('*                                                       *.\n')
fprintf('*  A su salida devolverá:                               *.\n')
fprintf('*  • la salida en cada iteración durante el proceso     *.\n')
fprintf('*  de ajuste, y(t)                                      *.\n')
fprintf('*  • la curva de aprendizaje (grafico de la curva)      *.\n')
fprintf('*  • los vectores de pesos obtenidos durante el pro-    *.\n')
fprintf('*  ceso. W(t)                                           *.\n')
fprintf('*  • la Potencia de la antena P(k)                      *.\n')
fprintf('*  • el numero de iteraciones en que converge la        *.\n')
fprintf('*    curva y se cumple el valor del error               *.\n')
fprintf('*                                                       *.\n')
fprintf('* alumno: Juan Francisco Alvarez Alvarado               *.\n')
fprintf('* alumno: Maribel del Rosario Chuez Gonzales            *.\n')
fprintf('*                                                       *.\n')
fprintf('* Profesor :Ing. Pedro Vargas                           *.\n')
fprintf('*                                                       *.\n')
fprintf('*******************************************************.\n')
t=input('teclee enter para proseguir');
 clc
% ingresamos el orden del filtro, cuyo coeficiente se va ha ajustar
orden=input(' ingrese el orden del filtro, cuyo coeficiente se va ha
ajustar....')
% ingresamos el numero de antenas del sistema
disp(' ingrese el numero de antenas entre 1 y 5');
x=input(' ingrese el numero de antenas entre 1,2,3,4 o 5....');
if x==1
    D(1)=input('ingrese el voltaje1 (en V)..... voltaje1=');
end

if x==2
    D(1)=input('ingrese el voltaje1 (en V)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
end

if x==3
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
end

if x==4
```

```
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
end
if x==5
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
end
% ingresamos el ruido que presenta cada antena del sistema
% Si existe ruido adicional al ocasionado por el sistema
if x==1
    E(1)=input(' ingrese el ruido en la antena1.....ruido1=');
    ruido=E(1);
end

if x==2
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    ruido=(E(1)+E(2))/2;
end

if x==3
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    ruido=(E(1)+E(2)+E(3))/3;
end

if x==4
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    ruido=(E(1)+E(2)+E(3)+E(4))/4;
end
if x==5
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    ruido=(E(1)+E(2)+E(3)+E(4)+E(5))/5;
end
opcion_ruido=2;
```

```
disp('si desea que el ruido adicional vaya a la señal a la entrada
digite 1');
disp(' de lo contrario se considerara que el ruido adicional se
agregara a la señal de referencia');
opcion_ruido=input('opcion_ruido=');
error=input('ingrese el error del sistema=');
% numero de iteraciones del proceso hasta que se cumpla el error
opcion_senal=2;
disp('si desea que la señal de entrada se use como señal de referencia
digite 1');
disp(' de lo contrario se considerara otra señal diferente como señal
de referencia ');
opcion_senal=input('opcion_señal=');
% numero de iteraciones del proceso hasta que se cumpla el error
N=2000;
j=0;
% vamos a considerar que todas las antenas tienen la misma señal a la
% entrada del filtro por ejemplo  entrada=sin((2*pi*k)/N)
if x==1
    h=[D(1)]
    p=[D(1)]
end
if x==2
    h=[D(1);D(2)]
    p=[(D(1)+D(2))/2]+[0.8*(D(1)+D(2))/2];
end
if x==3
    h=[D(1);D(2);D(3)]
    p=[(D(1)+D(2)+D(3))/3]+[0.8*(D(1)+D(2)+D(3))/3];
end
if x==4
    h=[D(1);D(2);D(3);D(4)]
    p=[(D(1)+D(2)+D(3)+D(4))/4]+[0.8*(D(1)+D(2)+D(3)+D(4))/4];
end
if x==5
    h=[D(1);D(2);D(3);D(4);D(5)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5))/5]+[0.8*(D(1)+D(2)+D(3)+D(4)+D(5))/5];
end
M=30;
for k=1:400
entrada(k)=sin((2*pi*k)/M);
% senal_referencia(k)=cos((2*pi*k)/M); % señal de referencia en algunos
libros se la llama señal deseada
end
for k=1:400
%entrada(k)=sin((2*pi*k)/M);
senal_referencia(k)=cos((2*pi*k)/M); % señal de referencia en algunos
libros se la llama señal deseada
```

```
end
% señal a la entrada del filtro, genera 400 valores aleatorios de
muestra
entrada';
entrada=entrada';
if opcion_ruido==1
    entrada=entrada+ruido;
end
n = sqrt(0.05)*randn(400,1); % ruido a la entrada del filtro, genera
400 valores aleatorios de muestra
[b,a] = butter(2,0.25) % usaremos otra forma para calcular a y b
Gz = tf(b,a,-1) % la funcion de transferencia tf funciona en matlab 7
% esta funcion permite resolver la transformada z,
% el primer orden evalua valores de pesos con la funcion ldiv
%h=ldiv(b,a,orden)';
% if you use ldiv this will give h :filter weights to be
% ingreso de los voltajes en la señal de entrada (estos voltajes seran
% los valores de pesos iniciales que serviran para hallar los pesos W
y = lsim(Gz,entrada); % simula respuestas arbitrarias de entrada,
relaciona la señal Gz con la entrada
% para agregar el ruido
% este es el ruido que se genera internamente por el metodo LMS
% por lo general el valor de ruido que genera el algoritmo LMS puede
llegar al 10 % de la señal
n = n * std(y)/(10*std(n)); % aqui se genera el ruido del sistema
% el valor a la entrada del filtro es la suma de la señal entrante mas
el ruido
if opcion_senal==1
senal_referencia = y + n;
end
if opcion_senal>1
    senal_referencia';
    senal_referencia=senal_referencia';
    senal_referencia=senal_referencia+n;
end
if opcion_ruido>1
    senal_referencia=senal_referencia+ruido;
end
totallength=size(senal_referencia,1) % da el largo de la funcion d
entrada

% conociendo los voltajes de las antenas podemos determinar la potencia
% en ellas
% potencia de la señal a la salida (potencia del sistema
potencia=mean(senal_referencia.^2)
if x==1
% tambien calcularemos la potencia para cada una de las antenas
% antena1
entrada1=D(1)+n+E(1);
```

```
potencia1=mean(entrada1.^2)
end
if x==2
% antena1
entrada1=D(1)+n+E(1);
potencia1=mean(entrada1.^2)
% antena2
entrada2=D(2)+n+E(2);
potencia2=mean(entrada2.^2)
end
if x==3
% antena1
entrada1=D(1)+n+E(1);
potencia1=mean(entrada1.^2)
% antena2
entrada2=D(2)+n+E(2);
potencia2=mean(entrada2.^2)
% antena3
entrada3=D(3)+n+E(3);
potencia3=mean(entrada3.^2)
end
if x==4
% antena1
entrada1=D(1)+n+E(1);
potencia1=mean(entrada1.^2)
% antena2
entrada2=D(2)+n+E(2);
potencia2=mean(entrada2.^2)
% antena3
entrada3=D(3)+n+E(3);
potencia3=mean(entrada3.^2)
% antena4
entrada4=D(4)+n+E(4);
potencia4=mean(entrada4.^2)
end
if x==5
% antena1
entrada1=D(1)+n+E(1);
potencia1=mean(entrada1.^2)
% antena2
entrada2=D(2)+n+E(2);
potencia2=mean(entrada2.^2)
% antena3
entrada3=D(3)+n+E(3);
potencia3=mean(entrada3.^2)
% antena4
entrada4=D(4)+n+E(4);
potencia4=mean(entrada4.^2)
% antena5
```

```
entrada5=D(5)+n+E(5);
potencia5=mean(entrada5.^2)
end

% calculo de la cota de la contante de ajuste
cota=1/((orden+1)*potencia);
%para 60 puntos  por corrida determinamos los pesos w
%begin of algorithm
Rxx=0;
rxx=0;
% w = zeros ( orden  , 1 )
 w = zeros ( orden  , 1 )
N=80
for n = orden : N
    u = entrada(n:-1:n-orden+1);
u(find(u>1))=(u(1)+u(2)+u(3)+u(4)+u(5))/5;% para que los valores de
u(find(u<-1))=(u(1)+u(2)+u(3)+u(4)+u(5))/5;% entrada estén dentro de
%la curva en esta función seno
%u(find(u>1))=0.886;
%u(find(u<-1))=-0.886;
    u
    n
    y(n)= w' * u;  % y(n) señal a la salida
    % e(n) = senal_referencia(n) - y(n);
    e(n)=senal_referencia(n)-y(n);
    j=j+1 % nos da el numero de iteraciones hasta que se cumpla el
error
     if e(n)<error
        fprintf('el numero de iteraciones hasta que se completo el
error ');
        j
     end

    if n < 20
        mu=cota/2; % los valores de mu no son usados por el algoritmo
dmi en la determinación de los pesos. Pero usa el valor de cota para
calcular las potencias
    else
        mu=cota/4;
    end
    T=u*u';
    if j==1
        Rxx=T*0;
    end
    Rxx=Rxx+T;
    T2=senal_referencia(n)*u;
    if j==1
        rxx=T2*0;
    end
```

```
    rxx=rxx+T2;
    T3=inv(Rxx);
    w = (inv(Rxx))*rxx;  % ecuacion de pesos para el algoritmo dmi
w(find(w>1))=(w(1)+w(2)+w(3)+w(4)+w(5))/5; % valores de peso maximo
para que la función seno no se pierda
    w(find(w<-1))=(w(1)+w(2)+w(3)+w(4)+w(5))/5;
w

% w=w+mu*u*e(n)   ecuacion de pesos en el algoritmo lms
    % algunos autores usan el valor de w = w + 2*mu * u * e(n); para
    % evaluar los pesos
end
% chequeo de resultados de error
N=200
for n =  N+1 : totallength
    u = entrada(n:-1:n-orden+1) ;
    y(n) = w' * u ;   % y(n) señal a la salida
    % e(n) = senal_referencia(n) - y(n) ;
    e(n)=senal_referencia(n)-y(n);
end
hold on
plot(senal_referencia)
plot(y,'r');
title('salida del sistema') ;
xlabel('numero de muestras ')
ylabel('valor estimado a la salida ')
figure
semilogy((abs(e))) ;
title('curva de error') ;
xlabel('numero de muestras ')
ylabel('valor de error')
figure
plot(h, 'k+')
hold on
plot(w, 'r*')
legend('pesos actuales','pesos estimados')
title('comparacion de los pesos actuales con los pesos estimados') ;
% axis([coordenada eje x- coordenada eje x+  coordenada y- coordenada
y+])
axis([0 6 -1 1])
```

6.4 Descripción del ejemplo a correr en el programa.

Ingresaremos 2 curvas, la una será la señal de entrada:

entrada(k) = sin((2*pi*k)/M) y la segunda será la señal de referencia:

señal_referencia(k) = cos((2*pi*k)/M) ingresaremos el orden del filtro, y el numero de valores reales (numero de antenas), también ingresaremos la señal de transferencia del filtro, también ingresaremos el error deseado. en el algoritmo dmi al igual que en el algoritmo lms , rls y cma anteriores podremos también ingresar un ruido adicional a las antenas, escogiéndose si se quiere agregar el ruido a la entrada o a la salida del programa.

Ahora evaluaremos el algoritmo con los siguientes valores: (problema3)

```
entrada(k)=sin((2*pi*k)/M);
senal_referencia(k)=cos((2*pi*k)/M);(opción 2)
n=400 muestras, rango de seguridad N=80.
```

También hicimos un programa problema1 que utiliza la misma función senosoidal de entrada como función de referencia, y un programa problema2 que utiliza señales gaussianas aleatorias como señal de entrada y realizamos algunas simulaciones hicimos esto tanto para los algoritmos lms, rls y cma.

Estos los colocaremos en el anexo F por si se los quiere revisar.

6.5 Pruebas y resultados de las implementaciones DMI.

En este programa problema3_dmi hemos colocado una señal de entrada(k)=sin((2*pi*k)/M); para n=400 muestras, se utilizo el algoritmo dmi (algoritmo de matriz inversa), se utiliza una señal de referencia igual a la señal de entrada ingresada en una función de transferencia (y(n)) más un ruido añadido por el programa dmi, este ruido adicional es del orden de 2, 3 y hasta 10 por ciento en forma aleatoria. Utilizaremos los siguientes datos:

TABLA LX Datos de entrada para el Problema3_dmi_a

Tipo de señal utilizada: senosoidal

Algoritmo utilizado: dmi

Aplicación en matlab utilizado: problema3_dmi

Nombre dado al ejemplo: problema3_dmi_a

Orden de la señal: 5

Numero de antenas: 5

Valores iniciales de entrada para las antenas:

V1= 0.26 V, V2=0.25 V, V3=0.21 V, V4=0.18 V, V5=0.15 V

Valores de ruido ingresados para las antenas:

Ruido1= 0.01 V, ruido2=0.01 V, ruido3= 0.01 V, ruido4=0.02V, ruido5=0.02V

Opción_ruido=1, opción_señal=1

numero de muestras = 400, rango de seguridad=200

Error del sistema = 0.0000001

TABLA LXI Datos obtenidos por el ejemplo Problema3_dmi_a

Pesos reales:	0.26V	0.25V	0.21V	0.18V	0.15V
Pesos obtenidos dmi:	0.03V	0.32V	0.7V	0.05V	0.02V

Para ver las pantallas de visualización de las potencias del sistema, señal obtenida a la salida, comparación de señales y curvas de error del presente ejemplo ir al anexo F ejemplo problema3_dmi_a.

TABLA LXII Datos de entrada para el Problema3_dmi_b

Tipo de señal utilizada: senosoidal

Algoritmo utilizado: dmi

Aplicación en matlab utilizado: problema3_dmi

Nombre dado al ejemplo: problema3_dmi_b

Orden de la señal: 5

Numero de antenas: 5

Valores iniciales de entrada para las antenas:

V1= 0.30 V, V2=0.25 V, V3=0.20 V, V4=0.15V, V5=0.12 V

Valores de ruido ingresados para las antenas:

Ruido1= 0.01 V, ruido2=0.01 V, ruido3= 0.01 V, ruido4=0.02 V, ruido5=0.02 V

Opción_ruido=1, opción_señal=1

numero de muestras =400, N=80 evaluación de pesos.

rango de seguridad=200

Error del sistema = 0.0000001

TABLA LXIII Datos obtenidos por el ejemplo Problema3_dmi_b

Pesos reales:	0.30V	0.25V	0.20V	0.15V	0.12V
Pesos obtenidos dmi:	0.20V	0.26V	0.20V	0.39V	0.08V

Para ver las pantallas de visualización de las potencias del sistema, señal obtenida a la salida, comparación de señales y curvas de error del presente ejemplo ir al anexo F ejemplo problema3_dmi_b.

Ahora en el mismo problema3.dmi lo evaluaremos para la opción 2, esperamos tener un error mayor que en la opción 1. Utilizaremos una señal senosoidal seno a la entrada y una señal coseno como señal de referencia, utilizaremos 400 muestras, N=80 para la evaluación de pesos . y N=200 en el lazo de control de error.

TABLA LXIV Datos de entrada para el Problema3_dmi_c

Tipo de señal utilizada: senosoidal

Algoritmo utilizado: dmi

Aplicación en matlab utilizado: problema3_dmi

Nombre dado al ejemplo: problema3_dmi_c

Orden de la señal: 5

Numero de antenas: 5

Valores iniciales de entrada para las antenas:

V1= 0.30 V, V2=0.25 V, V3=0.20 V, V4=0.15V, V5=0.12 V

Valores de ruido ingresados para las antenas:

Ruido1= 0.01 V, ruido2=0.01 V, ruido3= 0.01 V, ruido4=0.02 V, ruido5=0.02 V

Opción_ruido=1, opción_señal=2

numero de muestras =400, N=80 evaluación de pesos.

rango de seguridad=200

Error del sistema = 0.0000001

TABLA LXV Datos obtenidos por el ejemplo Problema3_dmi_c

Pesos reales:	0.30V	0.25V	0.20V	0.15V	0.12V
Pesos obtenidos dmi:	0.18V	-0.09V	-0.93V	-0.75V	-0.78V

Para ver las pantallas de visualización de las potencias del sistema, señal obtenida a la salida, comparación de señales y curvas de error del presente ejemplo ir al anexo F ejemplo problema3_dmi_c.

observación: En algunas partes de la curva obtenida por el algoritmo dmi, el valor de la curva obtenida es del doble de la curva de la señal de referencia, si mantiene la forma de la curva original, pero aumentando al doble el tamaño de la señal. En otras secciones es menor, esto está relacionado con los lazos de control de convergencia y el lazo de control del error. (zonas de seguridad)

Ahora utilizaremos el programa2.dmi, en el programa se usaran señales Gausianas de entrada, para N=2000 muestras, el ruido del sistema será agregado aleatoriamente, la señal de referencia será la misma señal de entrada.

En el lazo de pesos usaremos un N=80, y en el lazo de control de la curva de error usaremos N=200.

TABLA LXVI Datos de entrada para el Problema2_dmi_a

Tipo de señal utilizada: gaussiana

Algoritmo utilizado: dmi

Aplicación en matlab utilizado: problema2_dmi

Nombre dado al ejemplo: problema2_dmi_a

Orden de la señal: 5

Numero de antenas: 5

Valores iniciales de entrada para las antenas:

V1= 0.0976 V, V2=0.2873 V, V3=0.3360 V, V4=0.2210V, V5=0.0964 V

Valores de ruido ingresados para las antenas:

Ruido1= 0.01 V, ruido2=0.01 V, ruido3= 0.01 V, ruido4=0.01 V, ruido5=0.01 V

Opción_ruido=1

numero de muestras =2000, N=80 evaluación de pesos.

rango de seguridad=200 para el lazo de control

Error del sistema = 0.0000001

TABLA LXVII Datos obtenidos por el ejemplo Problema2_dmi_a

Pesos reales:	0.0976V	0.2873V	0.3360V	0.2210V	0.0964V
Pesos obtenidos dmi:	0.04V	0.27V	0.32V	0.20V	0.05V

Para ver las pantallas de visualización de las potencias del sistema, señal obtenida a la salida, comparación de señales y curvas de error del presente ejemplo ir al anexo F ejemplo problema2_dmi_a.

Ahora utilizaremos el programa1.dmi, en el programa se usaran señales Senosoidales como valores de entrada y como la señal de referencia, utilizaremos en el programa valores de N=400 muestras para la entrada, el ruido del sistema será agregado aleatoriamente. En el lazo de pesos usaremos un N=80, y en el lazo de control de la curva de error usaremos N=200.

TABLA LXVIII Datos de entrada para el Problema1_dmi_a

Tipo de señal utilizada: senosoidal

Algoritmo utilizado: dmi

Aplicación en matlab utilizado: problema1_dmi

Nombre dado al ejemplo: problema1_dmi_a

Orden de la señal: 5

Numero de antenas: 5

Valores iniciales de entrada para las antenas:

V1= 0.26 V, V2=0.24 V, V3=0.21 V, V4=0.18 V, V5=0.15 V

Valores de ruido ingresados para las antenas:

Ruido1= 0.01 V, ruido2=0.01 V, ruido3= 0.01 V, ruido4=0.02 V, ruido5=0.02 V

Opción_ruido=1

numero de muestras =400, N=80 evaluación de pesos.

rango de seguridad=200

Error del sistema = 0.0000001

TABLA LXIX Datos obtenidos por el ejemplo Problema1_dmi_a

Pesos reales:	0.26V	0.24V	0.21V	0.18V	0.15V
Pesos obtenidos dmi:	no dato	0.12V	0.2V	0.05V	no dato

Los otros 2 valores se disparan y salen fuera de la grafica escogida en matlab.

Para ver las pantallas de visualización de las potencias del sistema, señal obtenida a la salida, comparación de señales y curvas de error del presente ejemplo ir al anexo F ejemplo problema1_dmi_a.

6.6 Comparación entre los resultados dados por los algoritmo lms, rls, cma y dmi

A continuación haremos una comparación de los pesos hallados para los programas problema3_lms, problema3_rls, problema3_cma y problema3_dmi.

A este ejemplo lo llamaremos ejemplo problema3_lms_x, problema3_rls_x, problema3_cma_x y problema3_dmi_x. Este problema tiene una señal senosoidal a la entrada del tipo senosoidal y otra función senosoidal (distinta a la salida) como señal de referencia. Usaremos un numero de muestras N=400, con un rango de seguridad de n=80, con cinco valores de entrada como pesos iniciales. lamda = 0.999994115 ; y delta=100000000000000, para la parte del ejemplo rls.

TABLA LXX Datos de entrada para realizar la comparación de los resultados de los programas problema3_lms, problema3_rls, problema3_cma, problema3_dmi (ejemplo problema3_lms_x, ejemplo problema3_rls_x, ejemplo problema3_cma_x y problema3_dmi_x)

Tipo de señal utilizada: senosoidal

Comparación de los resultados de los algoritmos lms, rls, cma y dmi

Aplicaciones en matlab utilizados: problema3_lms, problema3_rls, problema3_cma y Problema3_dmi

Nombre de los Ejemplos utilizados en la comparación: problema3_lms_x, problema3_rls_x, problema3_cma_x y problema3_dmi_x.

Orden de la señal: 5

Numero de antenas: 5

Valores iniciales de entrada para las antenas:

V1= 0.10 V, V2=0.08 V, V3=0.02 V, V4=0.05 V, V5=0.10 V

Valores de ruido ingresados para las antenas

Ruido1= 0.01 V, ruido2=0.01 V, ruido3= 0.01 V, ruido4=0.02 V, ruido5=0.02 V

opción_ruido=1 , opción_señal=2

numero de muestras =400, N=80 evaluación de pesos.

rango de seguridad=200 , Error del sistema = 0.0000001

Constantes de ajuste lms: mu=0.32 y mu=0.15

Constantes de ajuste rls: lamda = 0.999994115 delta = 100000000000000

Constante de ajuste cma: (u)cma=5*10^-7

El algoritmo dmi no usa ninguna constante de ajuste

Nota1.- la señal obtenida por el algoritmo cma fue aproximadamente de la mitad de la señal de referencia, conservaba la misma forma de la señal pero disminuía su amplitud

TABLA LXXI Datos obtenidos por el (ejemplo problema3_lms_x, ejemplo problema3_rls_x, ejemplo problema3_cma_x y problema3_dmi_x)

Pesos reales:	0.10V	0.08V	0.02V	0.05V	0.10V
Pesos obtenidos lms:	0.54V	0.2V	-0.09V	-0.4V	-0.62V
Pesos obtenidos rls:	0.185V	-0.01V	-0.11V	-0.09V	0.01V
Pesos obtenidos cma:	0.15V	0.11V	0.21V	0.30V	-0.20V
Pesos obtenidos dmi:	0.41V	-0.09V	-0.4V	-0.23V	-0.9V

Para ver las pantallas de visualización de las potencias del sistema, señal obtenida a la salida, comparación de señales y curvas de error del presente

ejemplo ir al anexo F comparación de gráficos ejemplo problema3_lms_x, problema3_rls_x, problema3_cma_x, problema3_dmi_x.

A continuación haremos otro ejemplo para el programa problema3 lms , rls, cma y dmi. Con la opción_señal = 2.

TABLA LXXII Datos de entrada para realizar la comparación de los resultados de los programas problema3_lms, problema3_rls, problema3_cma, problema3_dmi (ejemplo problema3_lms_z, ejemplo problema3_rls_z, ejemplo problema3_cma_z y problema3_dmi_z)

Tipo de señal utilizada: senosoidal

Comparación de los resultados de los algoritmos lms, rls, cma y dmi

Aplicaciones en matlab utilizados: problema3_lms, problema3_rls, problema3_cma y Problema3_dmi

Nombre de los ejemplos utilizados en la comparación: problema3_lms_z, problema3_rls_z, problema3_cma_z y problema3_dmi_z.

Orden de la señal: 5

Numero de antenas: 5

Valores iníciales de entrada para las antenas:

V1= 0.10 V, V2=0.20 V, V3=0.22 V, V4=0.15 V, V5=0.09 V

Valores de ruido ingresados para las antenas:

Ruido1= 0.01 V, ruido2=0.01 V, ruido3= 0.01 V, ruido4=0.02 V, ruido5=0.02 V

opción_ruido=1 , opción_señal=2

numero de muestras = 400, N = 80 evaluación de pesos.

rango de seguridad=200 , Error del sistema = 0.0000001

Constantes de ajuste lms: mu=0.32 y mu=0.15

Constantes de ajuste rls: lamda = 0.999994115 delta = 100000000000000

Constante de ajuste cma: (u)cma=5*10^-7

El algoritmo dmi no usa ninguna constante de ajuste

TABLA LXXIII Datos obtenidos por el (ejemplo problema3_lms_z, ejemplo problema3_rls_z, ejemplo problema3_cma_z y problema3_dmi_z)

Pesos reales:	0.10V	0.20V	0.22V	0.15V	0.09V
Pesos obtenidos lms:	0.45V	0.19V	-0.15V	-0.45V	-0.69V
Pesos obtenidos rls:	0.98V	-0.38V	-0.82V	-0.42V	0.79V
Pesos obtenidos cma:	0.15V	0.10V	0.20V	0.30V	-0.20V
Pesos obtenidos dmi:	0.45V	0.19V	-0.15V	-0.45V	-0.69V

Para ver las pantallas de visualización de las potencias del sistema, señal obtenida a la salida, comparación de señales y curvas de error del presente ejemplo ir al anexo F comparación de gráficos ejemplo problema3_lms_z, problema3_rls_z, problema3_cma_z, problema3_dmi_z.

Ahora compararemos los resultados obtenidos utilizando los algoritmo lms , rls, cma y dmi con los valores recomendados por la central matlab. (problema2_lms, problema2_rls, problema2_cma, problema2_dmi)

TABLA LXXIV Comparación de los resultados de los programas problema2_lms, problema2_rls, problema2_cma, problema2_dmi (ejemplo problema2_lms_x, ejemplo problema2_rls_x, ejemplo problema2_cma_x y problema2_dmi_x)

Tipo de señal utilizada: gaussiana

Comparacion de los resultados de los algoritmos lms, rls, cma y dmi

Aplicaciones en matlab utilizados: problema2_lms, problema2_rls, problema2_cma y problema2_dmi

Nombre de los ejemplos utilizados en la comparación: problema2_lms_x, problema2_rls_x, problema2_cma_x y problema2_dmi_x.

Orden de la señal=5

Numero de antenas=5

Valores iniciales de entrada para las antenas:

V1= 0.09763 V, V2=0.28731 V, V3=0.335965 V, V4=0.2209 V, V5=0.0963 V

Valores de ruido ingresados para las antenas:

Ruido1= 0.01 V, ruido2=0.01 V, ruido3= 0.01 V, ruido4=0.02 V, ruido5=0.02 V

Opción_ruido=1

numero de muestras =2000 en el lms y rls y 400 muestras en el cma y dmi

N=80 evaluación de pesos.

rango de seguridad=200 , Error del sistema = 0.0000001

Constantes de ajuste lms: mu=0.32 y mu=0.15

Constantes de ajuste rls: lamda = 0.999994115 delta = 100000000000000

Constante de ajuste cma: (u)cma=5*10^-7

El algoritmo dmi no usa ninguna constante de ajuste

TABLA LXXV Datos obtenidos por el (ejemplo problema2_lms_x, ejemplo problema2_rls_x, ejemplo problema2_cma_x y problema2_dmi_x)

Pesos reales:	0.09763V	0.28731V	0.335915V	0.2209V	0.0963V
Pesos obtenidos lms:	0.138V	0.25V	0.35V	0.242V	0.075V
Pesos obtenidos rls:	0.096V	0.28V	0.35V	0.22V	0.09V
Pesos obtenidos cma:	0.15V	0.10V	0.20V	0.30V	-0.20V
Pesos obtenidos dmi:	0.0426V	0.2929V	0.3785V	0.2093V	0.0056V

Para ver las pantallas de visualización de las potencias del sistema, señal obtenida a la salida, comparación de señales y curvas de error del presente ejemplo ir al anexo F comparación de gráficos ejemplo problema2_lms_x, problema2_rls_x, problema2_cma_x, problema2_dmi_x.

Ahora compararemos los resultados obtenidos por el problema1, problema1_lms , problema1_rls, problema1_cma, problema1_dmi con los mismos valores. Estos utilizan una señal senosoidal a su entrada como a su salida. La señal senosoidal será la misma para la entrada antes del filtro y para la señal de referencia.

Utilizaremos los siguientes valores de entrada:

entrada(k)=sin((2*pi*k)/M); senal_referencia(k)=cos((2*pi*k)/M);este programa tiene la ventaja de que la señal de referencia no es igual a la señal de entrada, pero se usa una señal parecida del mismo tipo senosoidal que tenga una alta correlación con la señal de entrada. Además en el programa si se desea cambiar la ecuación de referencia solo hay que cambiar la ecuación en el lazo for donde está la ecuación de referencia. Realizaremos el siguiente programa con N=2000 muestras, y rango de seguridad de N=80. Utilizaremos valores de lamda = 0.999994115 ; y delta=100000000000000. En el uso del algoritmo rls.

TABLA LXXVI Datos de entrada para realizar la comparación de los resultados de los programas problema1_lms, problema1_rls, problema1_cma, problema1_dmi (ejemplo problema1_lms_x, ejemplo problema1_rls_x, ejemplo problema1_cma_x y problema1_dmi_x)

Tipo de señal utilizada: senosoidal

Comparación de los resultados de los algoritmos lms, rls, cma y dmi

Aplicaciones en matlab utilizados: problema1_lms, problema1_rls, problema1_cma y problema1_dmi

Nombre de los Ejemplo utilizados en la comparación: problema1_lms_x, ejemplo problema1_rls_x, ejemplo problema1_cma_x y ejemplo problema1_dmi_x.

Orden de la señal=5

Numero de antenas=5

Valores iníciales de entrada para las antenas:

V1= 0.20 V, V2=0.25 V, V3=0.21 V, V4=0.18 V, V5=0.15 V

Valores de ruido ingresados para las antenas:

Ruido1= 0.01 V, ruido2=0.01 V, ruido3= 0.01 V, ruido4=0.02 V, ruido5=0.02 V

Opción_ruido=1

numero de muestras =400, N=80 evaluación de pesos.

rango de seguridad=200

Error del sistema = 0.0000001

Constantes de ajuste lms: mu=0.32 y mu=0.15

Constantes de ajuste rls: lamda = 0.999994115 delta = 100000000000000

Constante de ajuste cma: (u)cma=5*10^-7

El algoritmo dmi no usa ninguna constante de ajuste

TABLA LXXVII Datos obtenidos por el (ejemplo problema1_lms_x, ejemplo problema1_rls_x, ejemplo problema1_cma_x y problema1_dmi_x)

Pesos reales:	0.20	0.25V	0.21V	0.18V	0.15V
Pesos obtenidos lms:	0.29V	0.27V	0.23V	0.18V	0.13V
Pesos obtenidos rls:	0.04V	0.06V	-0.1V	0.09V	0.02V
Pesos obtenidos cma:	0.15V	0.10V	0.20V	0.30V	-0.20V
Pesos obtenidos dmi:	0.1117V	0.0437V	0.2114V	-0.9640V	0.5390V

Para ver las pantallas de visualización de las potencias del sistema, señal obtenida a la salida, comparación de señales y curvas de error del presente ejemplo ir al anexo F comparación de gráficos ejemplo problema1_lms_x, problema1_rls_x, problema1_cma_x, problema1_dmi_x.

Al utilizar funciones senosoidales vemos que el error en el programa lms fue menor que en el programa rls, esto sucedió en la mayor parte de las pruebas que hicimos para diferentes valores en los algoritmos lms y rls, aunque en algunas ocasiones también tuvieron una mejor convergencia el algoritmo rls, por ejemplo para los valores que mostramos a continuación. con funciones senosoidales.

TABLA LXXVIII Datos de entrada para realizar la comparación de los resultados de los programas problema1_lms, problema1_rls, problema1_cma, problema1_dmi (ejemplo problema1_lms_z, ejemplo problema1_rls_z, ejemplo problema1_cma_z y problema1_dmi_z)

Tipo de señal utilizada: senosoidal

Comparación de los resultados de los algoritmos lms, rls, cma y dmi

Aplicaciones en matlab utilizados: problema1_lms, problema1_rls, problema1_cma y Problema1_dmi

Nombre de los ejemplos utilizados en la comparación: problema1_lms_z, problema1_rls_z, problema1_cma_z y problema1_dmi_z.

Orden de la señal=5

Numero de antenas=5

Valores iniciales de entrada para las antenas:

V1= 0.09763 V, V2=0.287310 V, V3=0.335965 V, V4=0.2209 V, V5=0.0963V

Valores de ruido ingresados para las antenas:

Ruido1= 0.01 V, ruido2=0.01 V, ruido3= 0.01 V, ruido4=0.01 V, ruido5=0.02 V

Opción_ruido=1

numero de muestras =400, N=80 evaluación de pesos.

rango de seguridad=200 , Error del sistema = 0.0000001

Constantes de ajuste lms: mu=0.32 y mu=0.15

Constantes de ajuste rls: lamda = 0.999994115   delta = 100000000000000

Constante de ajuste cma: (u)cma=5*10^-7

El algoritmo dmi no usa ninguna constante de ajuste

TABLA LXXIX Datos obtenidos por el (ejemplo problema1_lms_z, ejemplo problema1_rls_z, ejemplo problema1_cma_z y problema1_dmi_z)

Pesos reales:	0.097631	0.287310V	0.335965V	0.2209V	0.0963V
Pesos obtenidos lms:	0.236V	0.234V	0.225V	0.199V	0.175V
Pesos obtenidos rls:	-1.64V	1.48V	0.64V	0.98V	-1.5V
Pesos obtenidos cma:	0.175V	0.10V	0.22V	0.291V	-0.20V
Pesos obtenidos dmi:	0.0625V	0.1624V	0.5860V	0.3563V	-0.1829V

Para ver las pantallas de visualización de las potencias del sistema, señal obtenida a la salida, comparación de señales y curvas de error del presente ejemplo ir al anexo F comparación de gráficos ejemplo problema1_lms_z, problema1_rls_z, problema1_cma_z, problema1_dmi_z.

Por lo tanto concluimos que los algoritmos lms, rls, y dmi al utilizar funciones senosoidales aumenta el error en comparación a los valores que hayamos con funciones gaussianas, en funciones senosoidales el error es aceptable en unos casos y muy alto en otros, pero estos datos pueden ser útiles para tener una idea de los valores reales, sobre todo con los algoritmos lms y rls que salieron más exactos, en el futuro se podría tratar de mejorar estos algoritmos, con condiciones

de funcionamiento para mejorar los valores obtenidos. Aunque hacer esto aumentara el costo computacional, también nos damos cuenta que es adecuado hacer hibridos de los programas para mejorar su convergencia. El algoritmo cma en señales gaussianas no converge, pero si converge con señales senosoidales ,aunque la curva obtenida en el algoritmo cma en algunos tramos suele ser menor que la señal de referencia.

En las pruebas que hicimos cambiamos algunas veces el valor de w para mejorar el valor de pesos y disminuir el error, y lo disminuyo un poco, pero lo mejor sería combinar el algoritmo rls con otros algoritmos adaptativos. Esto demuestra que tanto el algoritmo lms como el rls tienen ciertas ventajas en sus propiedades.

La ventaja del algoritmo lms es que casi siempre encuentra la convergencia en cualquier tipo de curva, y la del algoritmo rls es que tiene una mejor convergencia para valores gaussianos. Y estos al ser unidos con otros algoritmos adaptativos mejoran la convergencia.

CONCLUSIONES Y RECOMENDACIONES

CONCLUSIONES

En este proyecto se ha presentado los algoritmos LMS, RLS, CMA y DMI. Por considerárselos los más relevantes que permiten un diseño adaptado a los cambios del escenario de filtros de mínimo MSE con respecto a una referencia. Estos sistemas son de uso obligado en el modelado o ecualización de sistemas variantes como es el caso de señales gaussianas con ruido blanco, señales de tipo seno o coseno y para canales de comunicación. Entre las conclusiones que tenemos al ejecutar estos algoritmos tenemos:

1.- El algoritmo lms casi siempre converge, en una cantidad de veces superior a los otros programas.

2.- Al ingresar varios valores de antenas y poder saber su comportamiento (valores probables de los pesos) nos permitirán saber el desempeño de las antenas inteligentes.

3.- Con el cálculo de la potencia de la señal esperamos que se pueda evaluar a futuro la distancia a las antenas. Conociendo la distancia sabremos cual es la cobertura de la red.

4.- Al ingresar varios valores de antenas nos permitirá conocer la capacidad de la red y el posible comportamiento de las antenas.

5.- Al aplicar señales senosoidales la convergencia de los valores ingresados con los valores estimados fue inferior al de los valores obtenidos para señales gaussianas. Por lo tanto el error aumento también en todos los algoritmos lms, rls, cma y dmi al utilizar señales senosoidales, esto era una suposición que teníamos y se cumplio.

6.- Al utilizar el algoritmo rls su convergencia utilizando valores gaussianos fue mucho mejor que al utilizar el algoritmo lms

7.- Se puede utilizar el mismo valor de N en el lazo donde se calculan los pesos y en el lazo que evalúa la zona de seguridad en los programas lms, rls y cma

8.- En el lazo para determinar la zona de seguridad puede ingresársele un valor de N menor al del número de muestras que tenga la señal.

9.- No se puede utilizar el mismo valor de N en el lazo de los pesos y en el lazo de seguridad del algoritmo dmi

10.- El valor de N en el lazo de cálculo de pesos del algoritmo dmi debe ser un numero bajo entre N=40, N=80, la curva se pierde a veces para valores más bajos como N=10, y tiene muchos saltos en la parte de la curva en donde N esta activa. Si se coloca un número menor de N, por ejemplo N=4 el programa dice error, valor de N debe ser mayor o igual al número de orden del filtro. Esto es causado por los valores del lazo for que depende del valor del orden del filtro.

11.- En el algoritmo dmi, se obtiene la convergencia a pesar del valor de N tan bajo, porque la convergencia es más rápida.

12.- Los algoritmos cma y dmi algunas veces no convergen

13.- En el algoritmo dmi usamos la función find para encontrar valores que estuvieran fuera del rango de la curva, y reemplazamos el valor errado por un promedio de las señales correcta. Esto no fue necesario en ninguno de los otros programas.

14.- La formula de cota que hemos utilizado en el programa lms, no es parte del algoritmo lms, pero puede ser usada en él para evaluar el valor de la constante mu.

15.- Al utilizar la zona de seguridad esta permitía que la señal dentro de este intervalo sea casi igual a la señal de referencia, funciono perfectamente en todos los algoritmos, lms, rls, cma y dmi.

16.- Al ingresar los ruidos adicionales en la entrada del filtro la curva y la convergencia de los valores fue más exacta.

17.- Al ingresar los ruidos adicionales a la salida del filtro al ser agregada a la señal de referencia, aumento el error y la señal obtenida no era tan buena.

18.- Al utilizar la misma señal de entrada (opción señal= 1) como señal de referencia la convergencia fue mejor que cuando se usaba otra señal que la reemplace (opción señal=2)

19.- La opción 2 que reemplaza la señal de entrada por otra señal, también tuvo un desempeño bastante bueno, no tan exacto como la de la señal original, pero puede ser útil en algunos casos en que no se pueda utilizar la señal original.

20.- Hemos determinado la diferencia de convergencia que existe entre los algoritmos lms, rls, cma y dmi.

21.- En el algoritmo dmi la convergencia fue casi instantánea, tal vez 100 veces más rápida que la del algoritmo lms.

22.- En el algoritmo dmi se presenta un problema en el desarrollo de la matriz al evaluarse Rxx y rxx, dice que le faltan 5 valores (el número del orden del filtro) para tener el número de valores a evaluar, el programa en matlab de todas maneras saca los resultados, y lo coloca como un error menor. (En nuestra opinión se debiera tratar de agregar algunas líneas de código en donde se pueda arreglar este pequeño problema)

23.- Una solución para disminuir el error en el algoritmo CMA fue hacer un hibrido CMA-RLS, con solo unas pocas líneas de código se obtenía una disminución del error bastante buena, sumada a su característica de una rápida convergencia lo hacía tan exacto como el LMS.

24.- En el algoritmo cma su convergencia fue superior que la del algoritmo rls por lo menos unas 4 o 5 veces más rápida.

25.- En el algoritmo cma en el lazo de seguridad garantiza la convergencia de la curva y una buena señal, pero el gráfico del error no se puede evaluar en el intervalo de control. Esto debe de ser causado por alguna característica del algoritmo cma

26.- Fue necesario hacer un hibrido cma-rls para tener en el programa cma un programa en donde se pueda comparar la curva obtenida por el programa con la señal de referencia.

27.- Al utilizar el algoritmo cma se presento algo interesante, solo permitía al programa funcionar si el numero de orden del filtro era igual al número de antenas, entonces optamos por escribir un mensaje de advertencia en el programa señalando esta limitación.

28.- Al colocar el valor de constante en el algoritmo cma se tiene que colocar valores muy bajos, utilizamos (u)cma=5*10^-7, cuando colocamos valores más altos (como los valores de mu=0.32 del lms) el programa no convergía, y señalaba error

29.- En algunos casos fue necesario variar los valores de los filtros para ayudar a que el resultado fuera más exacto.

30.- En algunas ocasiones con solo variar levemente los valores de los filtros la señal se perdía y nunca hallaba la convergencia.

31.- Al ingresar los valores de las señales debe de ingresarse un valor que este cerca y de preferencia dentro del rango de la señal, de lo contrario no se halla la convergencia.

32.- Un efecto interesante que se presento con el algoritmo cma y con el algoritmo dmi es que la grafica del error no se calculaba dentro de la zona de seguridad del programa.

33.- Hay que anotar que la señal obtenida por el algoritmo cma con señales Senosoi-Dales en algunos casos la señal obtenida fue un poco más de la mitad de la señal de Referencia, es decir que se tendría algo de pérdida de potencia, aunque la señal si converge, y puede ser útil en algunos casos, ya que la señal es filtrada correctamenmente, un ejemplo de esto se lo puede ver en el ejemplo problema3_cma_x.

34.- En el algoritmo dmi se presento algo parecido a lo ocurrido en el algoritmo cma en ciertos tramos de la curva, en algunas partes de la curva obtenida por el algoritmo dmi la señal también disminuía y en otras no, esto está relacionado con las zonas de control de convergencia, cambiando la zona de convergencia se obtiene una curva que no perdia señal en estos tramos para el algoritmo dmi. Esto lo podemos ver en

el ejemplo problema3_dmi_x.

35.- El cambiar el valor de los lazos de control si mejoro la señal obtenida en el Algoritmo dmi, en el algoritmo cma no presento diferencias.

RECOMENDACIONES

1.- Como futuras líneas de investigación planteamos que se construya algún circuito eléctrico o antena que simule un sistema de antenas real, también debería investigarse una mejora en el direccionamiento de la antena DOA. Buscar la fórmula adecuada para determinar la distancia a la antena utilizando la potencia, también se podría plantear hacer un algoritmo music y un algoritmo DKP (Back Propagation) para ver su comportamiento y si puede ser usado y en qué condiciones, también podría hacerse un algoritmo NCMA (algoritmo de modulo constante normalizado).

2.- Podrían hacerse algoritmos hibridos lms-dmi, lms-music, u otros algoritmos híbridos que permitan obtener mas rápidas convergencias, menor error, y utilizar las mejores características de los algoritmos adaptativos.

ANEXO A

Desarrollo de las ecuaciones del algoritmo LMS para valores discretos

Desde la ecuación A.1 hasta la ecuación de pesos haremos el mismo análisis que hicimos para el algoritmo LMS en función del tiempo, explicando más exhaustivamente de donde se origina cada ecuación.

Ecuacion A.1

Conociendo que:

$$\varepsilon^2(t) = [\, d^*(t) - W^H x(t) \,]^2 \qquad (3.3)$$

El valor de R en forma discreta es igual al valor esperado de:

$$\underline{R} = E\{ \underline{X}_n \underline{X}_n^H \} \qquad (A.1)$$

Y el valor de P es igual al valor esperado de la función de control d(n)Xn

$$\underline{P} = E\{d^*(n) X_n\} \qquad (A.2)$$

Con esto estimamos el gradiente teórico para un instante n ecuación (A.3).

$$\nabla\xi(n) = \underline{R}W_n - \underline{P} = E\{\underline{X}_n\underline{X}_n^H\}W_n - E\{d^*(n)X_n\} \qquad \textbf{(A.3)}$$

El LMS estima el gradiente de un modo aparentemente grosero pero de una sencillez y calidad espectaculares. De hecho el LMS aproxima los valores esperados de la formula anterior por sus valores instantáneos. Al tomar esta estimación y teniendo en cuenta que la salida del filtro obtenemos la ecuación A.4.

$$\underline{W}_{n+1} = W_n + \mu\underline{X}_n(d^*(n)-y^*(n)) = \underline{W}n+u\underline{X}_n\varepsilon^*(n) \qquad \textbf{(A.4)}$$

Donde wn es el valor de los pesos en forma iterativa, épsilon es el error medio y u es un valor de paso cuando actúa en forma variable.

Con la ecuación de adaptación nos ayudara a determinar el valor de ε(n) que es el error o señal diferencia entre la referencia y la salida del filtro (con valores

discretos). Un esquema del LMS se presenta en la Figura 3.1 (capitulo 3) donde puede apreciarse su simplicidad.

Todo lo expresado para el método de gradiente es válido, en términos de valores esperados o medios, para el LMS. De hecho, es fácil mostrar la validez de esta afirmación comprobando, de nuevo cuál sería el valor del μ □□más adecuado. Nótese que si se denomina ε(n) al error que producen los pesos wn con el vector de datos Xn, se puede calcular cual es el error εo(n) que producen, con los mismos datos, los nuevos coeficientes. La condición de convergencia es que dicho error, en potencia, ha de ser menor que el anterior. Pasando a expresar εo(n) obtenemos la ecuación A.5. reemplazando 3.1 en 3.11 (capitulo 3) obtenemos la ecuación A.6

$$y(t) = W^H x(t) \qquad (3.1)$$

$$\varepsilon_o(n) = d(n)-y(n) \qquad (A.5)$$

$$\varepsilon_o(n) = d(n) - \underline{W}_{n+1}^H \underline{X}_n = d(n) - (W_n^H - u\underline{X}_n^H \varepsilon(n))\underline{X}_n = \varepsilon(n)(1- u\underline{X}_n^H \underline{X}_n) \qquad (A.6)$$

luego con la ecuación de adaptación obtenemos la condición de u, figura A.1:

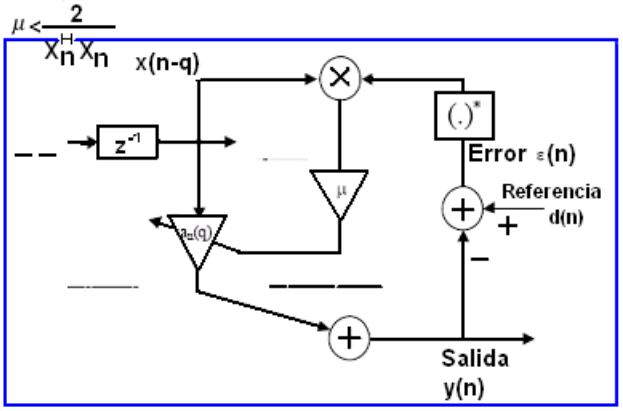


Figura A.1 **Detalle de la implementación del LMS para el peso q del filtro. Cada peso del FIR se actualiza con una fraccion del producto (o mezcla) de la muestra de entrada por el conjugado del error**

En esta expresión puede reconocerse el denominador que no es más que el valor instantáneo de Q, donde Q es la cantidad de veces la potencia de la señal de entrada. Por supuesto, el valor instantáneo haría fluctuar excesivamente el paso de adaptación y se toma precisamente el promedio para el siguiente valor.

Pasando al adecuado diseño del parámetro u □del paso de adaptación, éste ha de abordarse bajo el principio de que el LMS usa una variable aleatoria, y no el gradiente determinístico, en la actualización de los pesos. Este carácter aleatorio impide que el aprendizaje finalice. Incluso cuando el algoritmo alcanzase el mínimo, moviéndose alrededor del óptimo por si acaso éste cambiase. El efecto

es que los coeficientes del filtro pasan a ser una variable aleatoria cuya media es el filtro óptimo y cuya varianza es necesario determinar, así obtenemos el denominador de coeficientes ecuación A.7:

Donde w es el peso optimo que su valor esperado nos permitiría hallar el denominado error de coeficientes.

$$E\{\underline{W}n\} = Wopt$$

$$\underline{\Sigma}\, a = E\{\tilde{W}_n \tilde{W}_n^H\} \quad \text{siendo } \tilde{W}n = \underline{W}n - Wopt \quad \text{el denominado error de coeficientes} \qquad \textbf{(A.7)}$$

El valor esperado del peso es el peso optimo. Siendo $\tilde{W}_n$ igual a $W_n - W_{opt}$. Para calcular la varianza del error de coeficientes, de la ecuación de actualización de coeficientes en el LMS se resta en ambos lados el vector óptimo, con lo que obtenemos la ecuación A.8

$$W_{n+1} = \tilde{W}n - u\varepsilon^*(n)\underline{X}n \qquad \textbf{A(8)}$$

Se descompone el error entre la salida y la señal de referencia (error = d(t) – y(t)) en dos términos que llamaremos referencia compuesta, de estas dos

contribuciones diferentes tenemos que en la primera no se puede cancelar los valores con la entrada (valor de referencia) que denominaremos d(n), y la segunda contribución (salida del arreglo y(t)) es aquella en que con los pesos óptimos se suprimen valores. De la ecuación A.5 obtenemos la ecuación A.9:

$$\varepsilon(n) = d(n) - W_{opt}^{H}\underline{X}_n \qquad A(9)$$

Aquí determinamos el valor del error (destacándose que la potencia de d(n) es precisamente el MSE mínimoξmin).

Al usar esta última expresión en la ecuación A.7 y multiplicando a ambos lados por su transpuesto conjugado se obtiene la expresión siguiente ecuación A.10:

$$\tilde{W}_{n+1} = \tilde{W}_n + u X_n (d^*(n) - X_n^H \tilde{W}_n)$$

$$\tilde{W}_{n+1}\ \tilde{W}_{n+1}^H = [\tilde{W}_n + uX_n(d^*(n) - X_n^H \tilde{W}_n)][\tilde{W}_n^H + u(d(n) - \tilde{W}_n^H Xn)X\tilde{n}^H] \qquad A(10)$$

Al tomar el valor esperado, asumiendo que d(n) es independiente del resto de variables aleatorias y que los coeficientes del filtro son independientes de las muestras de la señal de entrada, obtenemos la ecuación A.11:

$$\underline{\underline{\Sigma}}_a^{n+1} = [\underline{\underline{I}} - u\underline{\underline{R}}]\underline{\underline{\Sigma}}_a^n[\underline{\underline{I}} - u\underline{\underline{R}}] + u^2\xi_{min}\underline{\underline{R}} \qquad \text{A(11)}$$

Llegados a este punto, si se considera el régimen permanente del algoritmo se puede suponer que la covarianza de los coeficientes se ha es estabilizado y no depende de n, con lo cual se obtiene la ecuación A.12:

$$0 = u\xi_{min}\underline{\underline{I}} - \underline{\underline{R}}\underline{\underline{\Sigma}}_a\underline{\underline{R}}^{-1} - \underline{\underline{\Sigma}}_a + u\underline{\underline{R}}\underline{\underline{\Sigma}}_a \qquad \text{A(12)}$$

Finalmente, teniendo en cuenta que el paso de adaptación se escogerá habitualmente de forma que $\mu << 1/\lambda max$,, el último término de la ecuación A.12 desaparece. Es fácil comprobar que en estas condiciones la expresión para la covarianza de los coeficientes nos da la ecuación A.13, el matlab realizara en funciones internamente todas estas operaciones:

$$\underline{\underline{\Sigma}}_a = \frac{u}{2}\xi_{min}\underline{\underline{I}} \qquad \text{A(13)}$$

y esta es solución de la ecuación A.12. , nótese que esta expresión revela que, para valores del paso de adaptación lejos de la cota superior (para que no diverja),

los coeficientes evolucionan independientemente unos de otros ya que la matriz de covarianza es diagonal. Obviamente, esto no ocurre al comienzo del aprendizaje por no darse las condiciones en que esta expresión se ha derivado. Esto se utiliza en inteligencia artificial, en términos de inteligencia artificial y considerando a los coeficientes como neuronas, al comienzo estas colaboran y cuando el aprendizaje ha terminado y están en proceso de espera de cambios para retomar el proceso estas evolucionan, alrededor de su valor, de manera independiente, eso si, pendientes de los cambios que pudieran producirse en el escenario para comenzar de nuevo su periodo de aprendizaje o colaborativo. Se puede decir, que en el proceso de mejora del conocimiento, la búsqueda de nueva información es independiente; mientras que, la asimilación de este es un proceso colaborativo.

Con lo anterior queda demostrado que el sistema adaptativo, bajo LMS, va a mostrar una fluctuación a su salida, que el usuario de ésta percibirá como ruido y que es debido a que el filtro no presenta un ajuste perfecto a los coeficientes óptimos, dado el carácter aleatorio del instrumento de aprendizaje que es el gradiente instantáneo. La cuestión es que este ruido de desajuste, en el caso de ecualización para comunicaciones, hará disminuir la SNR a la salida del ecualizador y podría afectar seriamente a la tasa de error del sistema. De hecho

este ruido de desajuste provocara que la potencia del error sea siempre, en media, superior al mínimo. Este efecto se hace evidente escribiendo la ecuación del MSE en función del error de coeficientes, ecuación A.14:

$$\xi(n) = \xi_{min} + \tilde{W}_n^H \underline{\underline{R}} \tilde{W}_n \qquad A(14)$$

Como puede verse la potencia del error se ha convertido en una variable aleatoria. Al tomar su valor esperado y teniendo en cuenta que (según ecuación A.13) obtenemos la ecuación A.15:

$$E\{\tilde{W}_n^H R \tilde{W}_n\} = \text{Traza}\,[\underline{\underline{\Sigma}}_a \underline{\underline{R}}] = \frac{u}{2}\xi_{min}\,\text{Traza}(R) \qquad A(15)$$

se puede concluir que el exceso de error, llamado también desajuste, está definido como la ecuación A.16:

$$\% = \frac{E\{\xi(n)\} - \xi_{min}}{\xi_{min}}\,100\% \qquad A(16)$$

En definitiva, con todo este análisis obtenemos las ecuaciones A.17 y A.18

$$\% = \frac{u}{2} \text{Traza}(\underline{\underline{R}}) \qquad \text{A(17)}$$

$$\mu = \frac{2\alpha}{Px(n)} \qquad \text{A(18)}$$

el ruido de desajuste será de α%. En forma numérica, un alfa de 0.1 equivale a tener un exceso de error o ruido de desajuste del 10%. Existe por tanto un compromiso entre velocidad de adaptación y nivel de desajuste. Esto completa el diseño del LMS.

El contenido de este apartado demuestra la manera correcta de proceder ante cualquier otra regla de aprendizaje, esté basada en el gradiente o no. Vale pues la pena recordar brevemente los pasos seguidos. En primer lugar se ha de probar la convergencia al óptimo o diseño deseado y comenzar la asignación pertinente de parámetros para que esto sea así. En segundo lugar se ha de analizar la velocidad de convergencia, tanto sobre una colección limitada de datos, iteraciones, como en caso contrario sobre vectores de datos ilimitado. Por último, en fase de seguimiento se ha de determinar el ruido de desajuste, o lo que es lo

mismo su velocidad de seguimiento o capacidad de reacción a cambios. Es también destacable la orientación en términos de sistema de aprendizaje en todas las fases mencionadas. Nótese que altas velocidades de convergencia, o aprendizaje rápido, llevan asociados desajustes elevados.

También se puede interpretar que con pasos de adaptación altos se aprende mucho pero con mayores valores de error el sistema 'medita' poco sobre lo aprendido y esa es la razón de su mayor tasa de error. En resumen, está más ávido por aprender que por sedimentar lo aprendido. El compromiso entre desajuste y velocidad de convergencia puede apreciarse en la figura A.2.

ANEXO B

DESARROLLO DE LA SOLUCION DE WIENER

Conociendo la ecuación 3.57 podemos obtener la evolución de los pesos a lo largo de las iteraciones, y con estos valores también podemos encontrar la potencia de la señal si conocemos el orden de la constante de ajuste, también nos permitirá calcular la cota de la constante de ajuste, esta cota es importante porque nos garantizara la convergencia de los valores de entrada.

Esta potencia se ve en la ecuación B.1

$$R_x W^o = P_{dx} \qquad \textbf{(B.1)}$$

Nosotros para poder encontrar la solución optima de wiener deberemos saber el orden de los coeficientes. Notamos que la matrix de correlación es una matrix de segundo orden. Si la matrix es invertible nos damos cuenta que esto le da características para procesar la señal en algunas aplicaciones que nos permitirán hallar el valor optimo (ecuación B.2)

$$W^o = R_x^{-1} P_{dx} \qquad \textbf{(B.2)}$$

La mejor manera de entender esto es con valores reales y su aplicación en

matlab

Considere el filtro FIR adaptativo cuyo esquema se presenta a continuación:

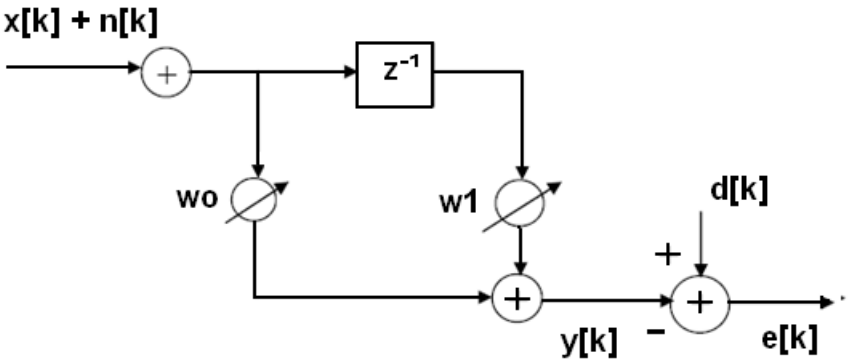


fig B.1 diagrama de un filtro adaptativo con una señal de entrada mas ruido, con una señal de referencia.

Donde:

$$x[k] = \operatorname{sen}\left[\frac{2\pi k}{N}\right] \qquad \textbf{(B.4)}$$

n[k] es un ruido aditivo, blanco y gaussiano de media nula y varianza igual a 0,05 **(B.5)**

$$d[k] = 2\cos\left[\frac{2\pi k}{N}\right] \qquad \textbf{(B.6)}$$

Matriz de autocorrlacion de la señal de entrada.Rxx:

$$Rxx = \begin{bmatrix} 0.5 + E\left[n^2[k]\right] & 0.5 * \cos\left[\frac{2\pi}{N}\right] \\ 0.5 * \cos\left[\frac{2\pi}{N}\right] & 0.5 + E\left[n^2[k]\right] \end{bmatrix} \qquad \textbf{(B.7)}$$

Matriz de correlacion cruzada entre la entrada y la salida deseada, Rxd:

$$R_{xd} = \begin{bmatrix} 0 \\ -\operatorname{sen}\left[\frac{2\pi}{N}\right] \end{bmatrix} \qquad \textbf{(B.8)}$$

El objetivo es encontrar el vector de pesos óptimo que minimice el error cuadrático medio entre la salida del filtro y la salida deseada.

Calculamos el vector de pesos óptimo y el error cuadrático medio mínimo para el problema planteado. Por medio de w* ecuación :

$$w^* = R_{XX}^{-1} \cdot R_{xd} = \begin{bmatrix} \dfrac{2 \cdot \cos\left[\frac{2\pi}{N}\right] \cdot \operatorname{sen}\left[\frac{2\pi}{N}\right]}{\left(1 + E\left[(n(k^2))\right]^2 - \cos^2\left[\frac{2\pi}{N}\right]\right.} \\ \dfrac{-2 \cdot \left(1 + 2.E\left[(n(k^2))\right]\right) \cdot \operatorname{sen}\left[\frac{2\pi}{N}\right]}{\left(1 + E\left[(n(k^2))\right]^2 - \cos^2\left[\frac{2\pi}{N}\right]\right.} \end{bmatrix} \qquad \textbf{(B.9)}$$

Y que el error cuadrático medio mínimo viene dado por:

$$\xi_{min} = \xi_{min} = E[d_k^2] - W^{*T} R_{xx} W^* - 2R_{xd}^T W^* = E[d[k]^2] - R_{xd}^T \cdot W^* \qquad \textbf{(B.10)}$$

En matlab esto se escribirá de la siguiente manera:

```
% Cálculo del vector de pesos óptimo.
W0_min = (2*cos(2*pi/N)*sin(2*pi/N))/(((1+2*0.05).^2)-(cos(2*pi/N)^2));
W1_min = (-2*(1+2*0.05)*sin(2*pi/N))/(((1+2*0.05).^2)-(cos(2*pi/N)^2));

% Cálculo del mínimo de la superficie de error:
Rxd_real = [0; -sin(2*pi/N)];
potencia_deseada = 2; % Potencia de la señal deseada:
error_minimo = potencia_deseada - (Rxd_real)'*[W0_min; W1_min];


% Cálculo de la cota de la constante de ajuste y la potencia de la señal:
% Potencia de la señal:
potencia = mean(entrada.^2);
```

```
orden=1;
Cota = 1/((orden + 1)*potencia);

% Generación de las señales.
k = 0:199; N = 30;x = sin((2*pi*k)/N); % Se generan 200 muestras de señal.
n = sqrt(0.05)*randn(1,200);
entrada = x + n;
señal_de_referencia_d(k) = 2*cos((2*pi*k)/N);
```

la señal de referencia debe de tener una alta correlación de parecido con la señal de entrada.

Para un filtro de orden M. Rx es una matrix de M x M, $\mathbf{W}^{O}$ es un vector M x 1, y p es un vector M x 1.

Hay que tener presente que si no existe una relación entre la señal y el dato ingresado demuestra que es un algoritmo tipo LMS, ya que si existiera una relación entre la señal significaría que el algoritmo seria del tipo RLS, el valor del error es igual a la varianza de la densidad de la señal**.**

Un problema que nosotros tenemos al utilizar este tipo de ecuaciones es el largo de el filtro M, este largo depende del número de coeficientes que se ingresen, para evaluar estas ecuaciones utilizamos operaciones computacionales que nos permitan determinar el valor optimo de los coeficientes, hay que utilizar un numero razonable de valores en las ecuaciones para evitar que la computadora se

inhiba de hacer los cálculos, esto dependerá de la capacidad de el procesador de la computadora y de la cantidad de ecuaciones que ingresemos. El programar utilizando el algoritmo LMS tiene la ventaja que es más fácil de entender que si utilizaremos otros algoritmos adaptativos, el algoritmo LMS tiene la ventaja de tener características de tipo lineal , por lo que nos garantiza una escasa complejidad matemática y que tendera a una convergencia lenta pero segura.

Con el uso de filtros obtendremos valores de muestras o iteraciones, el error rms y curvas de aprendizaje.

El filtro de wiener también es muy ampliamente usado en los algoritmos adaptativos dmi (algoritmo de inversión de matrix directa)

ANEXO C

EL PROGRAMA SIMULINK Y SUS PRINCIPALES LIBRERIAS.

En el presente anexo explicaremos cómo funciona el programa simulink para ingresar funciones y graficar

Simulink

Simulink es un sistema interactivo para simular sistemas dinamicos no lineales. La gran ventaja de Simulink es su interfaz grafica, mediante la cual se pueden implementar complicados modelos y obtener simulaciones en un tiempo extremadamente rapido a traves de los diagramas de bloques.

C.1. Diagramas de Bloques

En pocas palabras para utilizar Simulink solo hay que saber como traducir una ecuación diferencial o de diferencias a un diagrama de bloques. Por ejemplo si se desea simular la respuesta de la siguiente ecuación diferencial

$$\begin{aligned} \dot{y}(t) + 3y(t) &= 1 \\ y(0) &= 0 \end{aligned} \qquad \text{(C.1)}$$

primero hay que generar el diagrama de bloques del modelo, el cual se observa en la fig. C.1. La implementación en Simulink se muestra la fig. c.2, y atraves de esta se obtiene la respuesta del sistema descrito por la ecuación (c.1), la cual se aprecia en la fig. c.3.

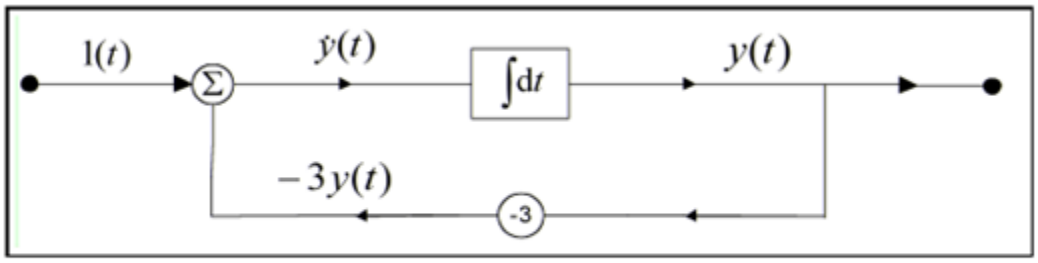


Figura C.1 Diagrama de Bloques

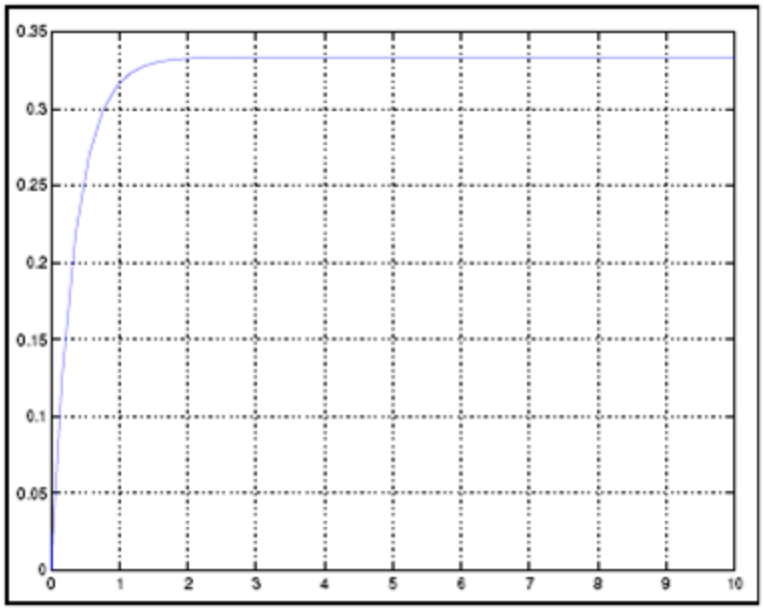


Figura C.2 Respuesta del sistema dinamico

C.2. Usando Simulink

Para utilizar Simulink solo es necesario construir el diagrama de bloques mediante los bloques predefinidos que vienen en la librería.

El uso de los bloques es bastante sencillo, solo hay que arrastrar el icono del bloque al modelo que se está construyendo. Si se desean cambiar los parámetros de los bloques solo hay que hacer un doble clic sobre su icono.

Para conectar los bloques hay que situar el puntero del mouse sobre el puerto de salida del primer bloque, con lo que el puntero deber¶³a cambiar a una cruz, y arrastrar el mouse hacia el puerto de entrada del siguiente bloque.

A continuación se detallan los bloques más utilizados.

C.1. Librería \Continuos"

En esta librería se encuentran todos los bloques de tiempo continuo para sistemas lineales, los cuales son:

Derivative: Derivada numérica de la señal de entrada.

Integrator: Integra la señal de entrada.

Memory: Retrasa la señal en un tiempo de integración.

State-Space: Representación en variables de estado.

Transfer-Fcn: Representación en función de transferencia. Expresión ma-

tricial para el numerador, expresión vectorial para el denominador. El ancho de la salida debe ser igual al número de las del numerador. Los coeficientes son potencias descendentes de s.

Transport Delay: Aplica el retraso especificado a la señal de entrada.

Variable Transport Delay: Aplica un retraso a la primera señal de entrada. La segunda entrada especifica el retardo.

Zero-Pole: Representación en polos y ceros. Expresión matricial para los ceros. Expresión vectorial para los polos y la ganancia. El ancho de la salida debe igualar el número de columnas de la matriz de ceros, o uno si los ceros son un vector.

C.2. Libreria \Discrete"

Esencialmente es identica a la anterior, pero para sistemas lineales de tiempo discreto, i.e. para ecuaciones de diferencias. Los bloques son:

Discrete Transfer-Fcn: Función de transferencia discreta, analogo al caso continuo.

Discrete Zero-Pole: Representación discreta en polos y ceros. Identico al caso continuo.

Discrete Filter: Filtro discreto. Expresión vectorial para el numerador y el denominador. Los coeficientes son para potencias ascendentes de z¡1.

denominador. Discrete State-Space: Representación en variables de estado de tiempo discreto.

Discrete-Time Integrator: Integración en tiempo discreto de la señal de

entrada.

First-Order Hold: Retenedor de primer orden.

Unit Delay: Muestrea y retiene con un periodo de muestreo de retraso.

Zero-Order Hold: Retenedor de orden cero.

C.3. Librerias \Sources" y \Sinks"

Estas librerías son aquellas que proveen las fuentes y los sumideros de los diagramas de bloques. Algunos de estos son:

Clock: Libreria \Sources". Genera el tiempo de simulación actual.

Constant: Libreria \Sources". Genera una constante especificada en el parametro `Constant value' . Si 'Constant value' es un vector y 'Interpret vector parameters as

1-D' esta arriba (on), el valor constante es tratado como un arreglo 1-D. Sino, la salida es una matriz con las mismas dimensiones que el valor constante.

Signal Generator: Libreria \Sources". Genera varias formas de onda.

Sine Wave: Libreria \Sources". Genera una onda sinusoidal.

Step: Libreria \Sources". Genera un escalón.

Display: Libreria \Sinks". Representación numérica de los valores de entrada.

Scope: Libreria \Sinks". Representación grafica de los valores de entrada versus el tiempo de simulación.

Stop Simulation: Libreria \Sinks". Detiene la simulación cuando la entrada es distinta de cero.

To File: Libreria \Sinks". Escribe el tiempo y la entrada al archivo MAT especificado en formato fila. El tiempo esta en la primera fila

To Workspace: Libreria \Sinks". Escribe la salida al arreglo o estructura

especificado en el workspace principal de Matlab. Los datos no están disponibles hasta que la simulación se detiene.

disponibles hasta que la simulación se detiene.

XY Graph : Libreria \Sinks". XY scope usando la ventana grafica de Matlab. la primera entrada es usada como base temporal. Se ingresan los rangos del gráfico.

C.4. Otras Librerias

Ademas de las librerias anteriormente detalladas, existen otras que proveen los siguientes bloques que son extremadamente utiles:

Fcn: Libreria \Functions & Tables". Bloque para una expresión general.

Usa \u" como el nombre de la variable de entrada. Ejemplo: sin(u[1]

* exp(2.3 * -u[2]))

MATLAB Fcn: Libreria \Functions & Tables". Pasa los valores de entra-

da a una función de Matlab para evaluarla. La función debe retornar un solo argumento vectorial del largo de `Output width'. Ejemplos:

sin, sin(u), foo(u(1), u(2))

Polynomial: Libreria \Functions & Tables". Evaluación polinomial. Calcula P(u) dado por el arreglo de coeficientes polinomiales P. P esta ordenado del mayor al menor orden, la forma aceptada por la función polyval de Matlab.

S-Function: Libreria \Functions & Tables". Bloque definible por el usuario. Los bloques pueden estar escritos en M, C, Fortran o Ada y deben cumplir los estandares de S-function. t,x,u y flag son automaticamente entregados a la S-function por Simulink. Parametros \Extra" pueden

ser especificados en el campo `S-function parameters'.

Abs: Librería \Math". Valor absoluto, i.e. y = |u|.

Dot Product: Librería \Math". Producto interno (punto). y = sum(conj(

u1).*u2)

Gain: Librería \Math". Ganancia elemento a elemento (y = K.*u) o ga-

nancia matricial (y = K*u o y = u*K).

Math Function: Librería \Math". Funciones matemáticas incluyendo fun-

ciones logarítmicas, exponencial, potenciación, y modulo.

Matrix Gain: Librería \Math". Ganancia elemento a elemento (y = K.*u)

o ganancia matricial (y = K*u o y = u*K).

MinMax: Librería \Math". La salida es el mínimo o máximo de la entrada.

Para una sola entrada, los operadores se aplican a traves del vector de entrada. Para múltiples entradas, los operadores se aplican a traves de las entradas.

Product: Libreria \Math". Multiplica o divide las entradas. Especificar una de las dos opciones siguientes:

1. * o / para cada puerto de entrada (ej., **/*)

2. Un escalar especicando el número de puertos de entrada a ser multiplicados.

El valor escalar `1' para producto elemento a elemento causa que todos los elementos de un solo vector de entrada sean multiplicados.

Sum: Librería \Math". Suma o substrae las entradas. Especificar una de las dos opciones siguientes:

1. Un string conteniendo + o - para cada puerto de entrada, | para espacio entre los puertos (ej. ++|-|++)

2. Un escalar ¸ 1. Un valor >1 suma todas las entradas; 1 suma los elementos de una solo vector de entrada.

Trigonometric Function: Libreria \Math". Funciones trigonometricas e hiperbolicas.

Demux: Libreria \Signals & Systems". Divide:

1. señales vectoriales a escalares o vectores más pequeños, o

2. señales tipo bus producidas por el bloques Mux en sus valores

escalares, vectoriales o matriciales constituyentes. Chequear `Bus Selection Mode' para dividir señales tipo bus.

Mux: Librería \Signals & Systems". Multiplexa señales escalares, vectoriales, o matriciales a un bus.

Terminator: Librería \Signals & Systems". Usado para \terminar"señales de salida. (Previene advertencias acerca puertos de salida no conectados.)

Bibliografia

[1] The Mathworks, \Getting Started with Matlab"

ANEXO D

Funciones más comunes utilizadas en matlab y en el presente programa

Close all	Cierra todas las figuras existentes
Inf	infinito
Figure	Se utiliza para pedir otra figura a matlab
I	Unidad imaginaria
Eps	Precisión relativa de punto flotante, 2^-52
Sin(x)	Función seno de x
Cos(x)	Funcion coseno(x)
Tan(x)	Funcion tangente de x
Exp(x)	Funcion exponencial de x
Log(x)	Función logaritmo natural de x
Plot(x,y)	Función que grafica y vs x
Clear(A)	Borra la variable A
Eig(A)	Calcula los valores y vectores propios de la matriz A
Poly(A)	Calcula los coeficientes del polinomio característico de la matriz A
Size(A)	Retorna el largo del vector x
Help función	Llama a la ayuda sobre la información sobre la funcion
Lookfor palabra	Retorna las funciones en las que aparece el string palabra
Hold on	Retiene un grafico actual y le agrega un nuevo grafico, hace lo mismo que plot
Clear all	Borra todas las variables
Clc	Borra pantalla
Hold off	Suelta la figura
Legend	Da leyenda a la figura
Stem(x,y)	Grafica señales en tiempo discreto
Subplot(m,n,i)	Permite agrupar varios graficos en una misma figura
Roots(coef)	Calcula las raíces del polinomio característico de la matriz A
Sum(x)	Suma lod elementos del vector x

Length(x)	Retorna el largo del vector x
Det(A)	Calcula el determinante de la variable A
J	Unidad utilizada en circuitos eléctricos
NaN	No es un numero
Load	Carga un programa, con los valores que tenia al momento de ser gravado
Sabe	Grava el programa como esta en ese instante, conservando el valor de las variables
Pi	3.1416

Numero	Variable x	Variable y

Entrada(n)	Señal a la entrada
Senal_referencia(n)	Señal de referencia
[b,a]=Butter(n1,n2)	Funcion qie devuelve los valores de a y b
h=ldiv(b,a,orden)	Devuelve el primer valor de pesos
tf(b,a,-1)	Función que devuelve la señal de transferencia
Randn(N,1)	Función randon devuelve valores aleatorios gaussianos
Orden	Numero del canal
Y=lsim(Gz,entrada)	Devuelve el valor de y evaluado en la función de transferencia, con los valores de entrada
Totallength=size(d,1)	Devuelve el numero de elementos que tiene la variable d
Xlabel	Leyenda eje x
Ylabel	Leyenda eje y
Axis(x1,x2,y1,y2)	Da el tamaño de la pantalla
Lamda	Valor de la constante del filtro de kalman, cercano a 1
P	Vector de valores, numero de elementos dados por la variable orden
K	Ganancia de kalman
T=abs(y(n))	Almacena el valor absoluto de la función y(n)

A(find(A>1))=0.886	Función encontrar, encuentra los valores mayores de 1 en la matriz, y reemplaza estos valores mayores de 1 por 0.886
y(n)’	Es la transpuesta de y(n)

ANEXO E

ANALISIS DEL ALGORITMO RLS PARA VALORES DISCRETOS.

Asi vemos que $\mathbf{R}_\lambda(n)$ (capitulo 4 pag 180) difiere de R siguiendo dos aspectos. 1) la matriz $\mathbf{x}(k)\mathbf{x}^{\mathrm{T}}(k)$ es una matriz de pesos. Con un factor exponencial de λ^{n-k}

2) El uso de ventanas es asumido, de acuerdo con los datos de prioridad de tiempos, k=1 y zero y para k=1 vuelve a limites menores de la sumatoria, el mismo que es dado por la ecuación de $\mathbf{p}_\lambda(n)$. El error cuadrado mínimo total es dado por la ecuación e.1

$$J_{\min} = \mathbf{d}^{\mathrm{T}}(n)\Lambda\mathbf{d}(n) - \mathbf{w}^{\mathrm{T}}(n)\mathbf{p}_\lambda(n) = \sum_{k=1}^{n} \lambda^{n-k} d^2(k) - \mathbf{w}^{\mathrm{T}}(n)\mathbf{p}_\lambda(n) \qquad \textbf{(e.1)}$$

Nosotros aplicamos valores para n por un tiempo n > M, donde la matriz R de λ en la práctica no es singular, y entonces los valores de $\mathbf{R}_\lambda$ and $\mathbf{p}_\lambda(n)$ pueden computarse. (en el mundo real no existen valores singulares (imaginarios)). Nosotros resolvemos la ecuación normal en la formula 4.8, con el cual obtenemos los valores del coeficiente del filtro $\hat{w}(n)$. Este valor es repetido con el arrivo de nuevos datos, x(n),d(n), que es para los tiempos n+1, n+2,….. Aislando el termino de k=n, nosotros podemos escribir la formula 4.8 en la forma de la ecuación e.2

$$\mathbf{R}_{\lambda}(n) = \lambda\left[\sum_{k=1}^{n-1} \lambda^{n-1-k}\mathbf{x}(k)\mathbf{x}^{\mathrm{T}}(k)\right] + \mathbf{x}(n)\mathbf{x}^{\mathrm{T}}(n) \qquad \textbf{(e.2)}$$

Por definición la expresión más usada es $\mathbf{R}_{\lambda}(n-1)$ y es utilizado en la expresión f.1 aplicandola nos queda la ecuación e.3

$$\mathbf{R}_{\lambda}(n) = \lambda\mathbf{R}_{\lambda}(n-1) + \mathbf{x}(n)\mathbf{x}^{\mathrm{T}}(n) \qquad \textbf{(e.3)}$$

En la ecuación anterior $\mathbf{R}_{\lambda}(n)$ que es llamada la nueva matrix de correlación , esta matrix actualiza los pesos de la vieja matrix de correlación $\mathbf{R}_{\lambda}(n-1)$ con el factor n-1 con un termino $\mathbf{x}(n)\mathbf{x}^{\mathrm{T}}(n)$.

Si utilizamos la ecuación 4.9 obtendremos la ecuación e.4

$$\mathbf{p}_{\lambda}(n) = \lambda\mathbf{p}_{\lambda}(n-1) + \mathbf{x}(n)d(n) \qquad \textbf{(e.4)}$$

Que actualiza el vector de correlación. Al hacer la prueba encontramos que el valor de w cuyas iteraciones resuelven la ecuación normal 4.8 (capitulo 4)

Al evaluar la inversa de $\mathbf{R}_{\lambda}(n)$ nosotros observamos que al comparar e.3 con e.4

Obtenemos una nueva expresión para R de landa inversa. Que llamaremos e.5

$$(\lambda\mathbf{A}+\mathbf{x}\mathbf{x}^{\mathrm{T}})^{-1}=\lambda^{-1}\mathbf{A}^{-1}-\frac{(\lambda^{-1}\mathbf{A}^{-1})\mathbf{x}\mathbf{x}^{\mathrm{T}}(\lambda^{-1}\mathbf{A}^{-1})}{1+\lambda^{-1}\mathbf{x}^{\mathrm{T}}\mathbf{A}^{-1}\mathbf{x}} \qquad \textbf{(e.5)}$$

$$\mathbf{R}_{\lambda}^{-1}(n)=\lambda^{-1}\mathbf{R}_{\lambda}^{-1}(n-1)-\frac{\lambda^{-2}\mathbf{R}_{\lambda}^{-1}(n-1)\mathbf{x}(n)\mathbf{x}^{\mathrm{T}}(n)\mathbf{R}_{\lambda}^{-1}(n-1)}{1+\lambda^{-1}\mathbf{x}^{\mathrm{T}}(n)\mathbf{R}_{\lambda}^{-1}(n-1)\mathbf{x}(n)} \qquad \textbf{(e.6)}$$

La relación que nosotros encontramos de A , $\mathbf{A}=\mathbf{R}_{\lambda}(n)$, $\mathbf{B}^{-1}=\lambda\mathbf{R}_{\lambda}\,(n-1)$, $\mathbf{C}=\mathrm{x}(n)$ y de los valores de C=x(n), y D=1 en e.7 nos permitirá definir el vector columna en g(n) en la ecuación e.8

$$\mathbf{A}^{-1}=\mathbf{B}-\mathbf{B}\mathbf{C}(\mathbf{D}+\mathbf{C}^{\mathrm{T}}\mathbf{B}\mathbf{C})^{-1}\mathbf{C}^{\mathrm{T}}\mathbf{B} \qquad \textbf{(e.7)}$$

$$\mathbf{g}(n)=\frac{\lambda^{-1}\mathbf{R}_{\lambda}^{-1}(n-1)\mathbf{x}(n)}{1+\lambda^{-1}\mathbf{x}^{\mathrm{T}}(n)\mathbf{R}_{\lambda}^{-1}(n-1)\mathbf{x}(n)}=\frac{\mathbf{R}_{\lambda}^{-1}(n-1)\mathbf{x}(n)}{\lambda+\mathbf{x}^{\mathrm{T}}(n)\mathbf{R}_{\lambda}^{-1}(n-1)\mathbf{x}(n)} \qquad \textbf{(e.8)}$$

En donde la función g(n) es conocida como la función de gain.

A continuación pondremos un ejemplo del desarrollo de R y como hallar los valores de gradiente J.

Example 9.2.1: Let the desired response be $d = [1 \;\; 1 \;\; 1 \;\; 1]$, and the two measured signals be $x_1 = [0.7 \;\; 1.4 \;\; 0.4 \;\; 1.3]^T$, $x_2 = [1.2 \;\; 0.6 \;\; 0.5 \;\; 1.1]^T$. Then we obtain

$$\mathbf{R} = \mathbf{X}^T\mathbf{X} = \begin{bmatrix} 0.7 & 1.4 & 0.4 & 1.3 \\ 1.2 & 0.6 & 0.5 & 1.1 \end{bmatrix} \begin{bmatrix} 0.7 & 1.2 \\ 1.4 & 0.6 \\ 0.4 & 0.5 \\ 1.3 & 1.1 \end{bmatrix} = \begin{bmatrix} 4.30 & 3.31 \\ 3.31 & 3.26 \end{bmatrix}$$

$$\mathbf{p} = \mathbf{X}^T\mathbf{d} = \begin{bmatrix} 3.8 \\ 3.4 \end{bmatrix}, \quad \hat{\mathbf{w}} = \mathbf{R}^{-1}\mathbf{p} = \begin{bmatrix} 0.3704 \\ 0.6669 \end{bmatrix}, \quad J_{\min} = 0.3252$$

$$\hat{\mathbf{y}} = \mathbf{X}\,\hat{\mathbf{w}} = [1.0595 \;\; 0.9187 \;\; 0.4816 \;\; 1.2150]$$

ANEXO F

PANTALLAS DE VISUALIZACION DE LAS IMPLEMENTACIONES EN MATLAB DE LOS ALGORITMOS LMS, RLS, CMA, DMI

Para el algoritmo lms

Problema3_lms_a

Pantalla 3.1 Problema3_lms_a

```
*************************************************************************.
*            ESCUELA SUPERIOR POLITECNICA DEL LITORAL                   *.
*                                                                       *.
* PROBLEMA3_lms.- Determinar los pesos, el numero de iteraciones y la potencia*.
* de un grupo de antenas inteligentes utilizando el algoritmo LMS (algoritmo  *.
* minimo cuadrado, las antenas estan distribuidas en forma de celdas para lo  *.
* cual requeriremos los siguientes datos:                               *.
*                                                                       *.
*  • la señal aplicada a la entrada del filtro,   ej  Xn=sin((2*pi*k)/N)      *.
*  • el numero de antenas  x                                            *.
*  • la señal de referencia, d(k)=5*cos((2*pi*k)/N);                    *.
*  • el ruido de las diferentes antenas, (hay que tener presente que el       *.
*    algoritmo LMS en su estructura internamente produce un ruido del 10      *.
*    aparte de este ruido puede haber otros.  n(f)                      *.
*  • el orden del filtro FIR cuyos coeficientes se van a ajustar, orden       *.
*  • el número de iteraciones en el aprendizaje, N (este número se hará       *.
*    hasta que cumpla el error)                                         *.
*  • la constante de ajuste   u                                         *.
*  • el vector de pesos inicial. Los voltajes de las antenas V(f)             *.
*  • el error deseado (puede ser ingresado por teclado) error                 *.
*                                                                       *.
*  A su salida devolverá:                                               *.
*  • la salida en cada iteración durante el proceso de ajuste, y(t)           *.
*  • la curva de aprendizaje (grafico de la curva)                      *.
*  • los vectores de pesos obtenidos durante el proceso. W(t)                 *.
*  • la Potencia de la antena P(k)                                      *.
*  • el numero de iteraciones en que converge la curva y se cumple el valor   *.
*    del error.  j                                                      *.
* alumno: Juan Francisco Alvarez Alvarado                               *.
* alumno: Maribel del Rosario Chuez Gonzales                            *.
*                                                                       *.
* Profesor :Ing. Pedro Vargas                                           *.
*                                                                       *.
*************************************************************************.
```

Pantalla 3.2 Datos del programa problema3_lms_a

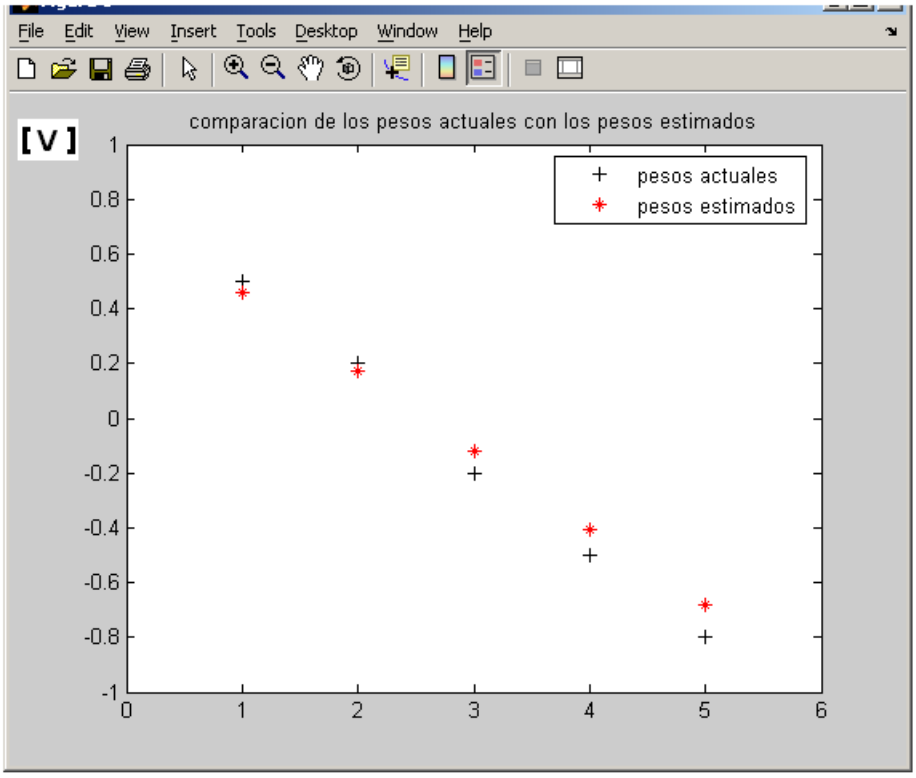
```
 ingrese el orden del filtro, cuyo coeficiente se va ha ajustar....5

orden =

     5

 ingrese el numero de antenas entre 1 y 5
 ingrese el numero de antenas entre 1,2,3,4 o 5....5
ingrese el voltaje1 (en v)..... voltaje1=0.5
 ingrese la voltaje2 (en V).... voltaje2=0.2
 ingrese la voltaje3 (en V).....voltaje3=-0.2
 ingrese la voltaje4 (en v)..... voltaje4=-0.5
 ingrese la voltaje5 (en v)..... voltaje5=-0.8
ingrese el ruido en la antena1.....ruido1=0.01
ingrese el ruido en la antena2.....ruido2=0.01
ingrese el ruido en la antena3.....ruido3=0.01
ingrese el ruido en la antena4.....ruido4=0.01
ingrese el ruido en la antena5.....ruido5=0.01
si desea que el ruido adicional vaya a la señal a la entrada digite 1
 de lo contrario se considerara que el ruido adicional se agregara a la señal de referencia
opcion_ruido=1
ingrese el error del sistema=0.0000001
si desea que la señal de entrada se use como señal de referencia digite 1
 de lo contrario se considerara otra señal diferente como señal de referencia
opcion_señal=1
```

Pantalla 3.3 Potencias obtenidas con el programa problema3_lms_a

```
totallength =

   400


potencia =

    0.5030


potencia1 =

    0.2647


potencia2 =

    0.0489


potencia3 =

    0.0412


potencia4 =

    0.2454


potencia5 =

    0.6296
```

Pantalla 3.4 Ejemplo problema3_lms_a Comparación de los pesos estimados

Ejemplo problema3_lms_a
comparacion de los pesos estimados de una señal senosoidal con los pesos reales o actuales de una señal senosoidal utilizando un algoritmo lms, orden=5, numero pesos=5



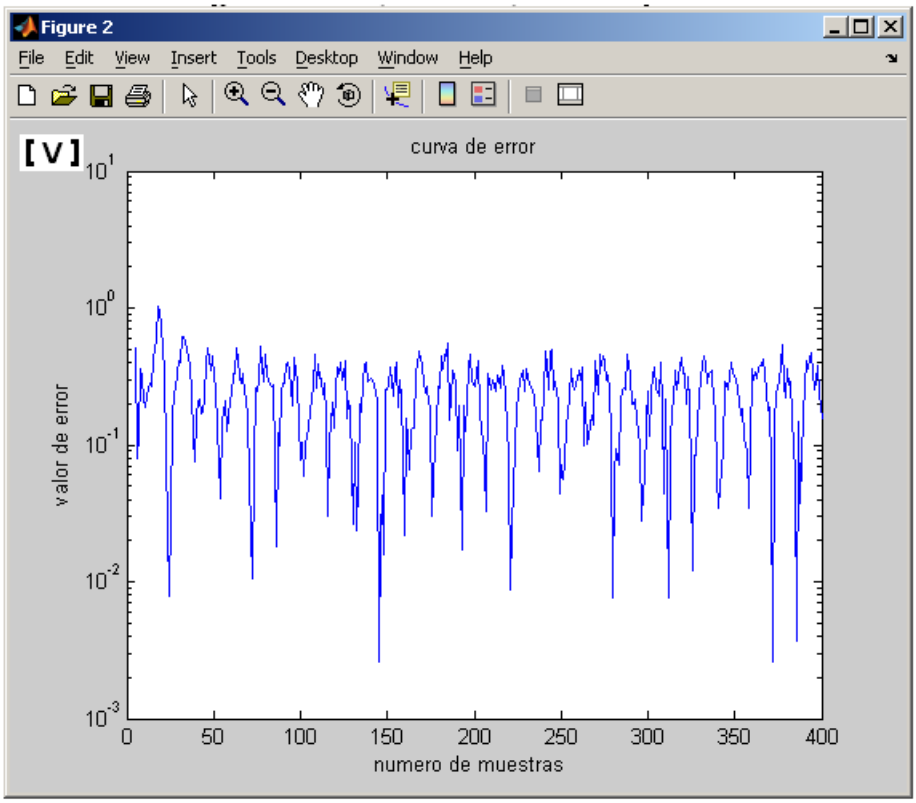
Eje de las x constante, eje de las y (amplitud de la señal en voltios). Amplitud_señal vs Valor constante

Vemos que el error de la curva es pequeño, aunque los puntos actuales deben de tener cierta cercanía, ya que si escogemos valores más distantes el error debe de

aumentar, y también se corre el riesgo de que el programa no encuentre la convergencia.

Pantalla 3.5 Ejemplo problema3_lms_a. Curva de error de una señal senosoidal en un algoritmo lms.

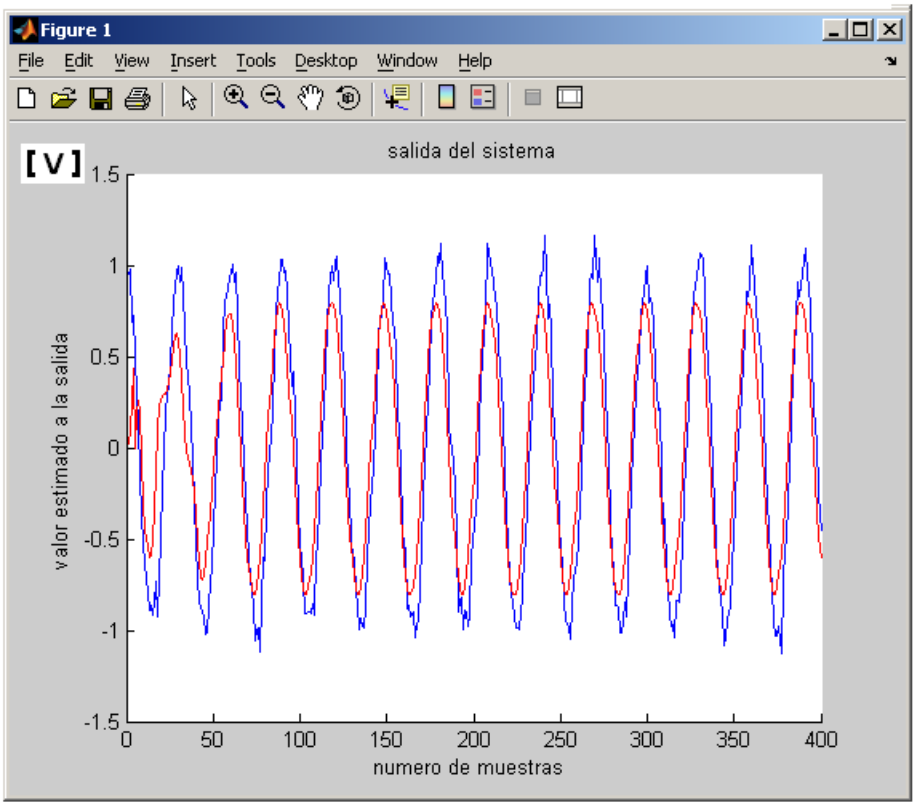
Ejemplo problema3_lms_a



Amplitud de la señal (voltios) vs numero de muestras

Pantalla 3.6 Ejemplo problema3_lms_a. Comparacion entre la salida del sistema de una señal senosoidal (lms) con la señal de referencia del sistema.

Ejemplo problema3_lms_a



Amplitud de la señal (voltios) vs numero de muestras
señal de referencia (color azul)
señal obtenida por el algoritmo lms (color rojo)

Si vemos el grafico de la salida del sistema se ve un pequeño error al principio de la curva para menos de 50 muestras, y después se va haciendo más exacta.

Problema2_lms_a

Pantalla 3.7 Problema2_lms_a

```
* PROBLEMA LMS2.- Determinar los pesos, el numero de iteraciones y la potencia*.
* de un grupo de antenas inteligentes utilizando el algoritmo LMS (algoritmo  *.
* minimo cuadrado, las antenas estan distribuidas en forma de celdas para lo  *.
* cual requeriremos los siguientes datos:                                      *.
*                                                                              *.
*  • la señal aplicada a la entrada del filtro,   ej  Xn=sin((2*pi*k)/N)       *.
*  • el numero de antenas  x                                                   *.
*  • la señal de referencia, ejemplo d(k)=cos((2*pi*k)/N);                     *.
*  • el ruido de las diferentes antenas, (hay que tener presente que el        *.
*    algoritmo LMS en su estructura internamente produce un ruido del 10       *.
*    por ciento aparte de este ruido puede haber otros ruidos.                 *.
*  • el orden del filtro FIR cuyos coeficientes se van a ajustar, orden        *.
*  • el número de iteraciones en el aprendizaje, N (este número se hará        *.
*    hasta que cumpla el error)                                                *.
*  • la constante de ajuste   u                                               *.
*  • el vector de pesos inicial. Los voltajes de las antenas V(f)              *.
*  • el error deseado (puede ser ingresado por teclado) error                  *.
*                                                                              *.
*  A su salida devolverá:                                                      *.
*  • la salida en cada iteración durante el proceso de ajuste, y(t)            *.
*  • la curva de aprendizaje (grafico de la curva)                             *.
*  • los vectores de pesos obtenidos durante el proceso. W(t)                  *.
*  • la Potencia de la antena P(k)                                             *.
*  • el numero de iteraciones en que converge la curva y se cumple el valor    *.
*    del error. j                                                              *.
* alumno: Juan Francisco Alvarez Alvarado                                      *.
* alumno: Maribel del Rosario Chuez Gonzales                                   *.
*                                                                              *.
* Profesor :Ing. Pedro Vargas                                                  *.
*                                                                              *.
********************************************************************************.
teclee enter para proseguir
```

En este ejemplo usaremos valores de ruido adicional de 0.01 en todas las señales

Pantalla 3.8 Datos del programa problema2_lms_a

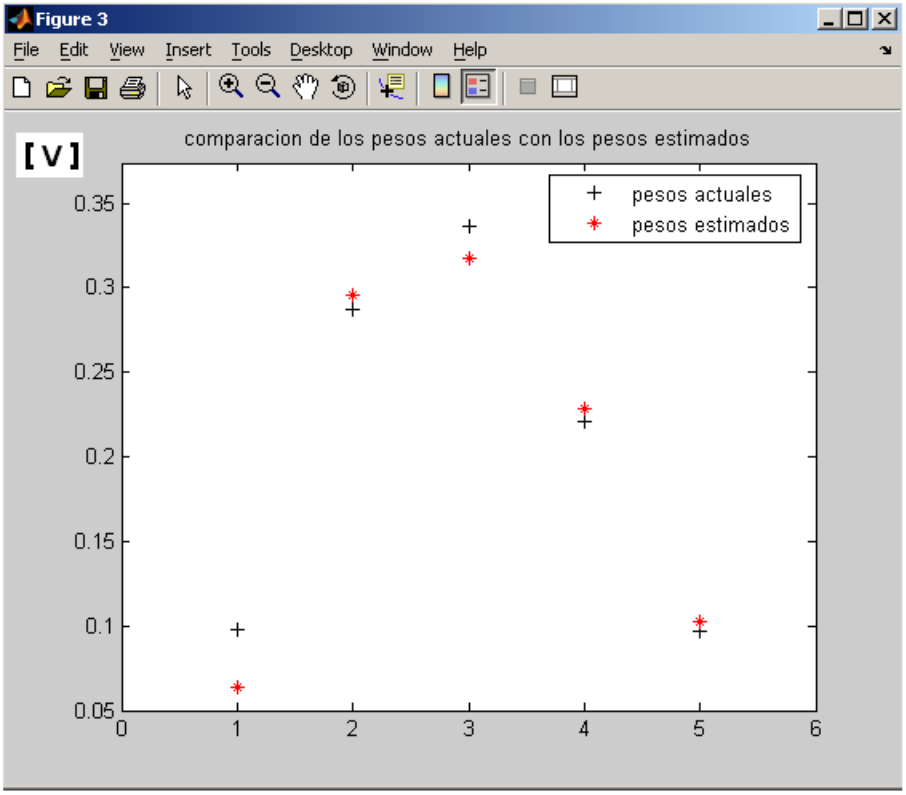
```
 ingrese el orden del filtro, cuyo coeficiente se va ha ajustar....5

orden =

     5

 ingrese el numero de antenas entre 1 y 5
 ingrese el numero de antenas entre 1,2,3,4 o 5....5
ingrese el voltaje1 (en v)..... voltaje1=0.0976
 ingrese la voltaje2 (en V).... voltaje2=0.2873
 ingrese la voltaje3 (en V).....voltaje3=0.3360
 ingrese la voltaje4 (en v)..... voltaje4=0.2210
 ingrese la voltaje5 (en v)..... voltaje5=0.0964
ingrese el ruido en la antena1.....ruido1=0.01
ingrese el ruido en la antena2.....ruido2=0.01
ingrese el ruido en la antena3.....ruido3=0.01
ingrese el ruido en la antena4.....ruido4=0.01
ingrese el ruido en la antena5.....ruido5=0.01
si desea que el ruido adicional vaya a la señal a la entrada digite 1
de lo contrario se considerara que el ruido adicional se agregara a la señal de referencia
opcion_ruido=1
ingrese el error del sistema=0.0000001
```

Pantalla 3.9 Potencias obtenidas con el programa problema2_lms_a

```
potencia =

    0.2481


potencia1 =

    0.0141


potencia2 =

    0.0910


potencia3 =

    0.1224


potencia4 =

    0.0559


potencia5 =

    0.0138
```

Pantalla 3.10 Ejemplo problema2_lms_a. Comparación de los pesos estimados

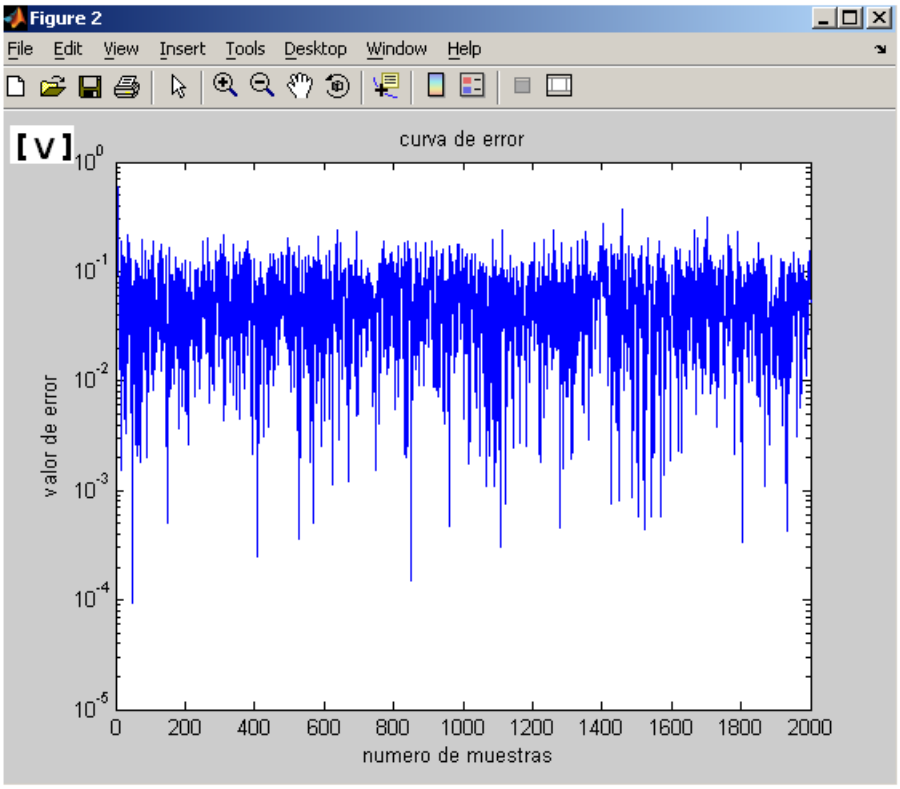
Ejemplo problema2_lms_a
comparacion de los pesos estimados de una señal gaussiana
con los pesos reales o actuales de una señal gaussiana
utlizando un algoritmo lms, orden del filtro=5, numero de pesos=5



Eje de las x constante, en el eje de las y amplitud de la señal en voltios. Amplitud_señal vs Valor constante.

Pantalla 3.11 Ejemplo problema2_lms_a. Curva de error de una señal gaussiana en un algoritmo lms.

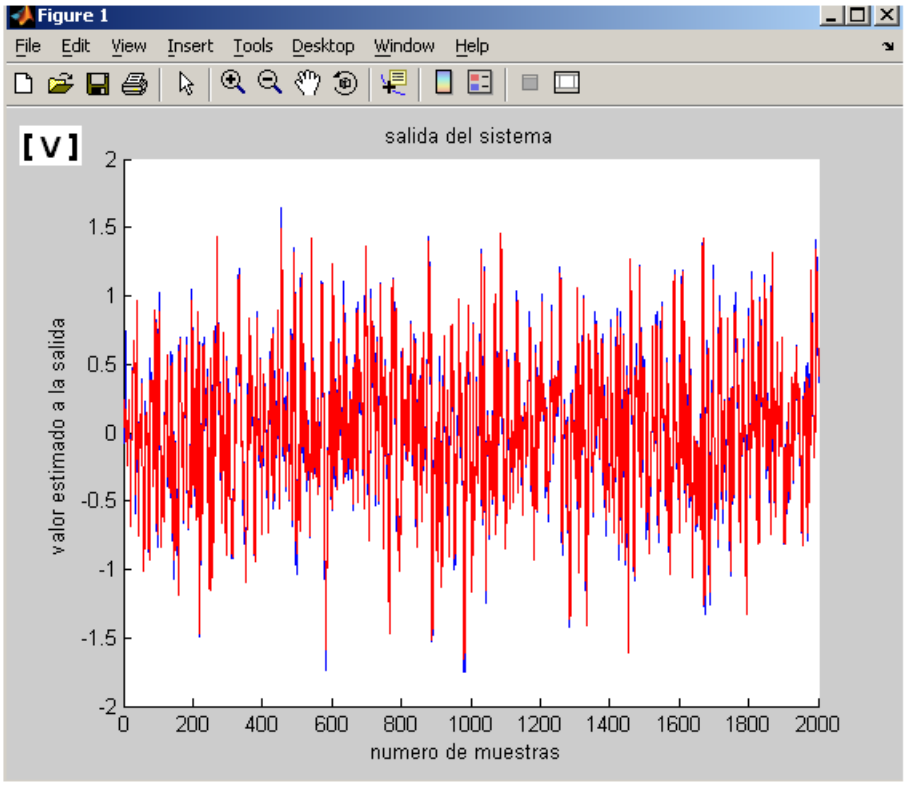
Ejemplo problema2_lms_a



Amplitud de la señal (voltios) vs numero de muestras

Pantalla 3.12 Ejemplo problema2_lms_a. Comparación entre la salida del sistema de una señal gaussiana (lms) con la señal de referencia del sistema.

Ejemplo problema2_lms_a



Amplitud de la señal (voltios) vs numero de muestras
señal de referencia (color azul)
señal obtenida por el algoritmo lms (color rojo)

Ahora compararemos el mismo ejemplo anterior con valores diferentes de ruido en las entradas (ruido2 y ruido3) y ver que diferencia hay entre los valores reales y los valores obtenidos por el programa.

Problema2_lms_b

Pantalla 3.13 Problema2_lms_b

```
* PROBLEMA LMS2.- Determinar los pesos, el numero de iteraciones y la potencia*.
* de un grupo de antenas inteligentes utilizando el algoritmo LMS (algoritmo  *.
* minimo cuadrado, las antenas estan distribuidas en forma de celdas para lo  *.
* cual requeriremos los siguientes datos:                                      *.
*                                                                              *.
*  • la señal aplicada a la entrada del filtro,   ej  Xn=sin((2*pi*k)/N)       *.
*  • el numero de antenas  x                                                   *.
*  • la señal de referencia, ejemplo d(k)=cos((2*pi*k)/N);                     *.
*  • el ruido de las diferentes antenas, (hay que tener presente que el        *.
*    algoritmo LMS en su estructura internamente produce un ruido del 10       *.
*    por ciento aparte de este ruido puede haber otros ruidos.                 *.
*  • el orden del filtro FIR cuyos coeficientes se van a ajustar, orden        *.
*  • el número de iteraciones en el aprendizaje, N (este número se hará        *.
*    hasta que cumpla el error)                                                *.
*  • la constante de ajuste   u                                               *.
*  • el vector de pesos inicial. Los voltajes de las antenas V(f)              *.
*  • el error deseado (puede ser ingresado por teclado) error                  *.
*                                                                              *.
*  A su salida devolverá:                                                      *.
*  • la salida en cada iteración durante el proceso de ajuste, y(t)            *.
*  • la curva de aprendizaje (grafico de la curva)                             *.
*  • los vectores de pesos obtenidos durante el proceso. W(t)                  *.
*  • la Potencia de la antena P(k)                                             *.
*  • el numero de iteraciones en que converge la curva y se cumple el valor    *.
*    del error. j                                                              *.
* alumno: Juan Francisco Alvarez Alvarado                                      *.
* alumno: Maribel del Rosario Chuez Gonzales                                   *.
*                                                                              *.
* Profesor :Ing. Pedro Vargas                                                  *.
*                                                                              *.
********************************************************************************.
teclee enter para proseguir
```

Pantalla 3.14 Datos del programa problema2_lms_b

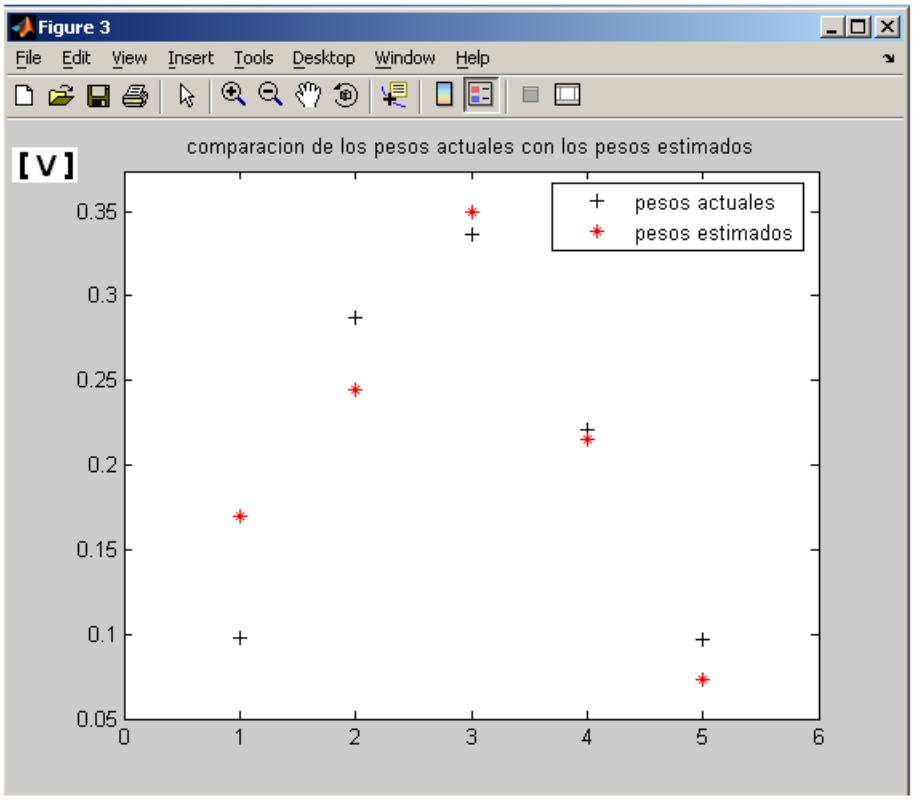
```
 ingrese el orden del filtro, cuyo coeficiente se va ha ajustar....5

orden =

     5

 ingrese el numero de antenas entre 1 y 5
 ingrese el numero de antenas entre 1,2,3,4 o 5....5
ingrese el voltaje1 (en v)..... voltaje1=0.0976
 ingrese la voltaje2 (en V).... voltaje2=0.2873
 ingrese la voltaje3 (en V).....voltaje3=0.3360
 ingrese la voltaje4 (en v)..... voltaje4=0.2210
 ingrese la voltaje5 (en v)..... voltaje5=0.0964
ingrese el ruido en la antena1.....ruido1=0.01
ingrese el ruido en la antena2.....ruido2=0.01
ingrese el ruido en la antena3.....ruido3=0.02
ingrese el ruido en la antena4.....ruido4=0.03
ingrese el ruido en la antena5.....ruido5=0.01
si desea que el ruido adicional vaya a la señal a la entrada digite 1
de lo contrario se considerara que el ruido adicional se agregara a la señal de referencia
opcion_ruido=1
ingrese el error del sistema=0.0000001
```

Pantalla 3.15 Potencias obtenidas con el programa problema2_lms_b

```
potencia =

    0.2778


potencia1 =

    0.0146


potencia2 =

    0.0918


potencia3 =

    0.1303


potencia4 =

    0.0663


potencia5 =

    0.0143
```

Pantalla 3.16 Ejemplo problema2_lms_b. Comparación de los pesos estimados

Ejemplo problema2_lms_b
Comparacion de los pesos estimados de una señal gaussiana con los pesos reales o actuales de una señal gaussiana utilizando un algoritmo lms, orden del filtro=5, numero de pesos=5

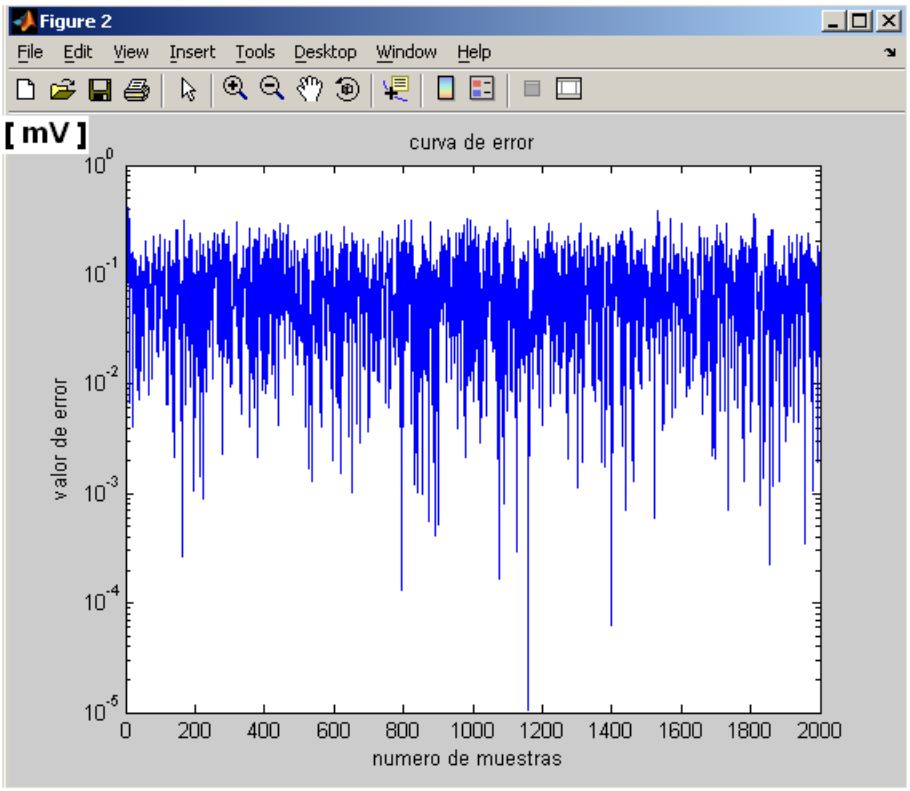


Eje de las x constante, en el eje de las y amplitud de la señal en voltios. Amplitud_señal vs Valor constante.

Observamos un pequeño aumento del error en los dos primeros valores reales, y después baja el error. .

Pantalla 3.17 Ejemplo problema2_lms_b. Curva de error de una señal gaussiana en un algoritmo lms.

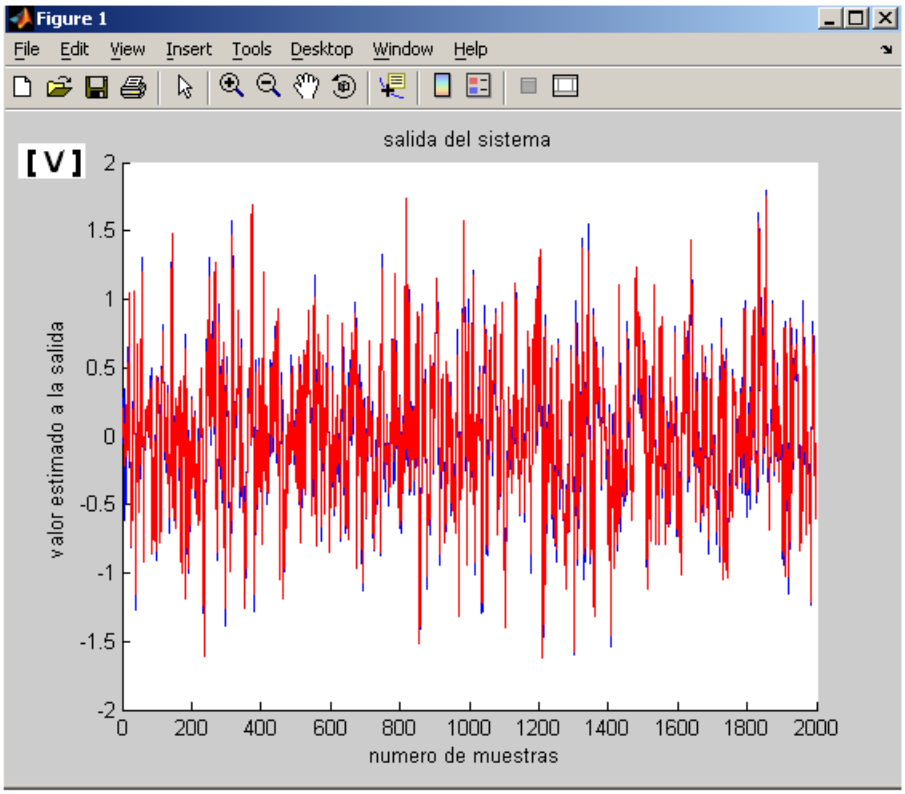
Ejemplo problema2_lms_b



Amplitud de la señal (voltios) vs numero de muestras

Pantalla 3.18 Ejemplo problema2_lms_b. Comparación entre la salida del sistema de una señal gaussiana (lms) con la señal de referencia del sistema.

Ejemplo problema2_lms_b



Amplitud de la señal (voltios) vs numero de muestras
señal de referencia (color azul)
señal obtenida por el algoritmo lms (color rojo)

Problema2_lms_c

Pantalla 3.19 Problema2_lms_c

```
   * PROBLEMA LMS2.- Determinar los pesos, el numero de iteraciones y la potencia*.
   * de un grupo de antenas inteligentes utilizando el algoritmo LMS (algoritmo  *.
   * minimo cuadrado, las antenas estan distribuidas en forma de celdas para lo  *.
   * cual requeriremos los siguientes datos:                                      *.
   *                                                                             *.
   *  • la señal aplicada a la entrada del filtro,   ej  Xn=sin((2*pi*k)/N)      *.
   *  • el numero de antenas  x                                                  *.
   *  • la señal de referencia, ejemplo d(k)=cos((2*pi*k)/N);                    *.
   *  • el ruido de las diferentes antenas, (hay que tener presente que el       *.
   *    algoritmo LMS en su estructura internamente produce un ruido del 10      *.
   *    por ciento aparte de este ruido puede haber otros ruidos.                *.
   *  • el orden del filtro FIR cuyos coeficientes se van a ajustar, orden       *.
   *  • el número de iteraciones en el aprendizaje, N (este número se hará       *.
   *    hasta que cumpla el error)                                               *.
   *  • la constante de ajuste   u                                               *.
   *  • el vector de pesos inicial. Los voltajes de las antenas V(f)             *.
   *  • el error deseado (puede ser ingresado por teclado) error                 *.
   *                                                                             *.
   *  A su salida devolverá:                                                     *.
   *  • la salida en cada iteración durante el proceso de ajuste, y(t)           *.
   *  • la curva de aprendizaje (grafico de la curva)                            *.
   *  • los vectores de pesos obtenidos durante el proceso. W(t)                 *.
   *  • la Potencia de la antena P(k)                                            *.
   *  • el numero de iteraciones en que converge la curva y se cumple el valor   *.
   *    del error. j                                                             *.
   * alumno: Juan Francisco Alvarez Alvarado                                     *.
   * alumno: Maribel del Rosario Chuez Gonzales                                  *.
   *                                                                             *.
   * Profesor :Ing. Pedro Vargas                                                 *.
   *                                                                             *.
   *******************************************************************************.
teclee enter para proseguir
```

Pantalla 3.20 Datos del programa problema2_lms_c

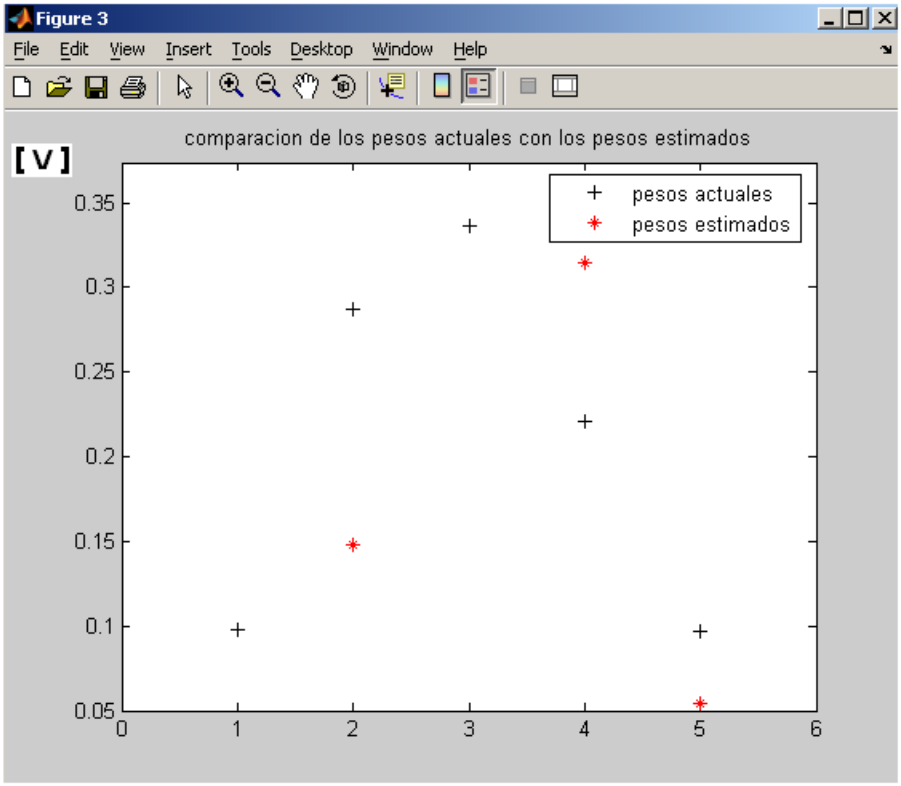
```
 ingrese el orden del filtro, cuyo coeficiente se va ha ajustar....8

orden =

     8

 ingrese el numero de antenas entre 1 y 5
 ingrese el numero de antenas entre 1,2,3,4 o 5....5
ingrese el voltaje1 (en v)..... voltaje1=0.0976
 ingrese la voltaje2 (en V).... voltaje2=0.2873
 ingrese la voltaje3 (en V).....voltaje3=0.3360
 ingrese la voltaje4 (en v)..... voltaje4=0.2210
 ingrese la voltaje5 (en v)..... voltaje5=0.0964
ingrese el ruido en la antena1.....ruido1=0.01
ingrese el ruido en la antena2.....ruido2=0.01
ingrese el ruido en la antena3.....ruido3=0.01
ingrese el ruido en la antena4.....ruido4=0.01
ingrese el ruido en la antena5.....ruido5=0.01
si desea que el ruido adicional vaya a la señal a la entrada digite 1
de lo contrario se considerara que el ruido adicional se agregara a la señal de referencia
opcion_ruido=1
ingrese el error del sistema=0.0000001
```

Pantalla 3.21 Potencias obtenidas con el programa problema2_lms_c

```
potencia =

    0.2788


potencia1 =

    0.0143


potencia2 =

    0.0911


potencia3 =

    0.1294


potencia4 =

    0.0657


potencia5 =

    0.0141
```

Pantalla 3.22 Ejemplo problema2_lms_c. Comparación de los pesos estimados

Ejemplo problema2_lms_c
Comparacion de los pesos estimados de una señal gaussiana con los pesos reales o actuales de una señal gaussiana utilizando un algoritmo lms, orden=8, numero de pesos=5



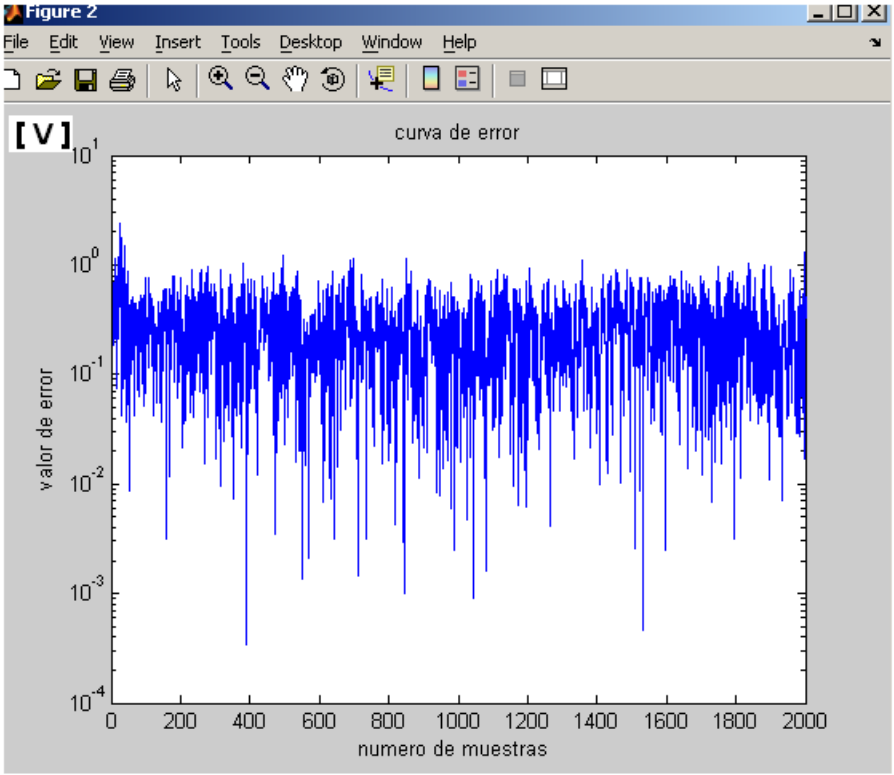
Eje de las x constante, en el eje de las y amplitud se la señal en voltios. Amplitud_señal vs Valor constante.

Para un filtro de nivel 8 y 5 antenas se ve un aumento considerable del error, el valor del primer peso y del tercer peso ni siquiera lo estima cerca de los valores

reales. Con lo cual se comprueba lo que dice la teoría de los algoritmos lms según wiener, que al variar el número de filtro y el número de antenas debe de aumentar el error. En otro ejemplo probaremos posteriormente esto mismo con alguna señal senosoidal.

Pantalla 3.23 Ejemplo problema2_lms_c. Curva de error de una señal gaussiana en un algoritmo lms.

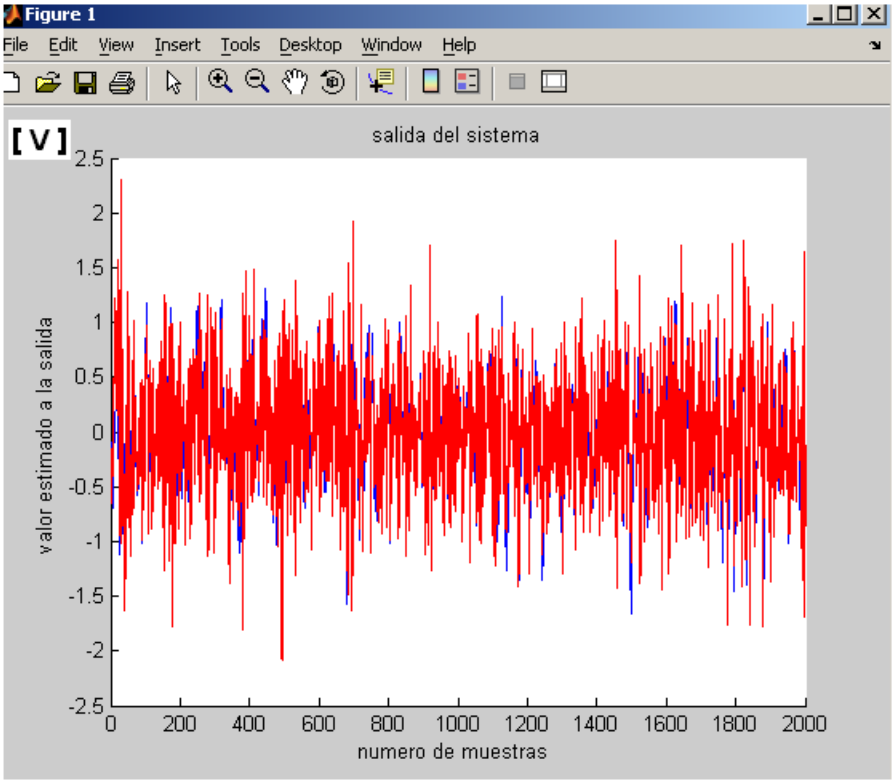
Ejemplo problema2_lms_c



Amplitud de la señal (voltios) vs numero de muestras

Pantalla 3.24 Ejemplo problema2_lms_c. Comparación entre la salida del sistema de una señal gaussiana (lms) con la señal de referencia del sistema.

Ejemplo problema2_lms_c



Amplitud de la señal (voltios) vs numero de muestras
señal de referencia (color azul)
señal obtenida por el algoritmo lms (color rojo)

Problema2_lms_d

Pantalla 3.25 Problema2_lms_d

```
   * PROBLEMA LMS2.- Determinar los pesos, el numero de iteraciones y la potencia*.
   * de un grupo de antenas inteligentes utilizando el algoritmo LMS (algoritmo  *.
   * minimo cuadrado, las antenas estan distribuidas en forma de celdas para lo  *.
   * cual requeriremos los siguientes datos:                                      *.
   *                                                                             *.
   *  • la señal aplicada a la entrada del filtro,   ej  Xn=sin((2*pi*k)/N)      *.
   *  • el numero de antenas  x                                                  *.
   *  • la señal de referencia, ejemplo d(k)=cos((2*pi*k)/N);                    *.
   *  • el ruido de las diferentes antenas, (hay que tener presente que el       *.
   *    algoritmo LMS en su estructura internamente produce un ruido del 10      *.
   *    por ciento aparte de este ruido puede haber otros ruidos.                *.
   *  • el orden del filtro FIR cuyos coeficientes se van a ajustar, orden       *.
   *  • el número de iteraciones en el aprendizaje, N (este número se hará       *.
   *    hasta que cumpla el error)                                               *.
   *  • la constante de ajuste   u                                               *.
   *  • el vector de pesos inicial. Los voltajes de las antenas V(f)             *.
   *  • el error deseado (puede ser ingresado por teclado) error                 *.
   *                                                                             *.
   *  A su salida devolverá:                                                     *.
   *  • la salida en cada iteración durante el proceso de ajuste, y(t)           *.
   *  • la curva de aprendizaje (grafico de la curva)                            *.
   *  • los vectores de pesos obtenidos durante el proceso. W(t)                 *.
   *  • la Potencia de la antena P(k)                                            *.
   *  • el numero de iteraciones en que converge la curva y se cumple el valor   *.
   *    del error. j                                                             *.
   * alumno: Juan Francisco Alvarez Alvarado                                     *.
   * alumno: Maribel del Rosario Chuez Gonzales                                  *.
   *                                                                             *.
   * Profesor :Ing. Pedro Vargas                                                 *.
   *                                                                             *.
   *******************************************************************************.
teclee enter para proseguir
```

Pantalla 3.26 Datos del programa problema2_lms_d

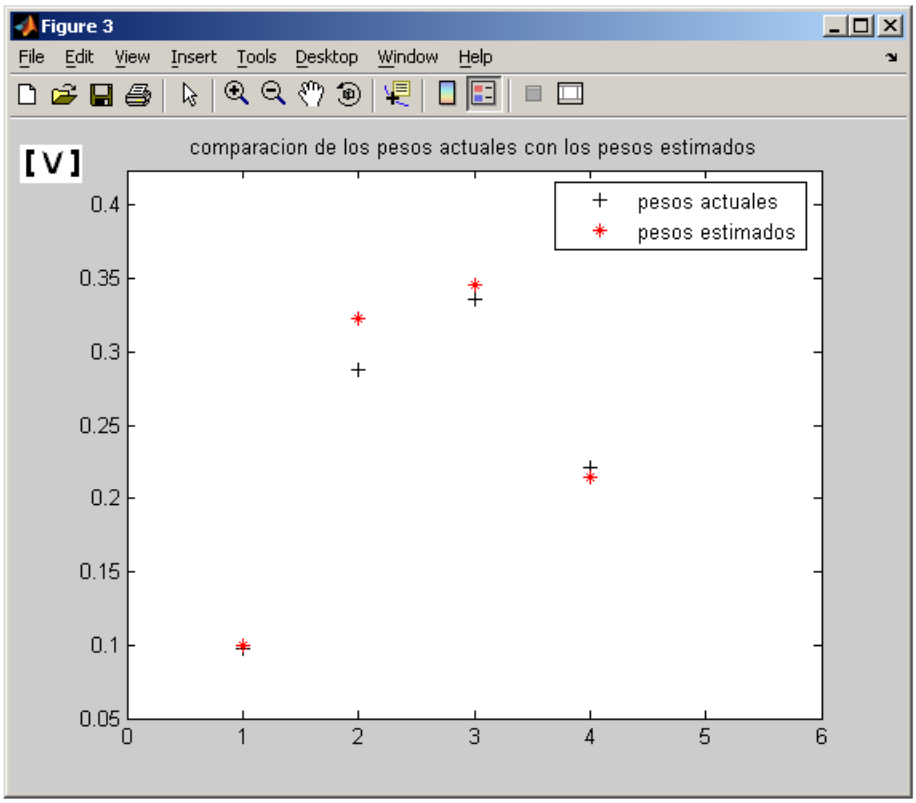
```
 ingrese el orden del filtro, cuyo coeficiente se va ha ajustar....5

orden =

     5

 ingrese el numero de antenas entre 1 y 5
 ingrese el numero de antenas entre 1,2,3,4 o 5....4
ingrese el voltaje1 (en v)..... voltaje1=0.0976
 ingrese la voltaje2 (en V).... voltaje2=0.2873
 ingrese la voltaje3 (en V).....voltaje3=0.3360
 ingrese la voltaje4 (en v)..... voltaje4=0.2210
ingrese el ruido en la antena1.....ruido1=0.01
ingrese el ruido en la antena2.....ruido2=0.01
ingrese el ruido en la antena3.....ruido3=0.02
ingrese el ruido en la antena4.....ruido4=0.03
si desea que el ruido adicional vaya a la señal a la entrada digite 1
de lo contrario se considerara que el ruido adicional se agregara a la señal de referencia
opcion_ruido=1
ingrese el error del sistema=0.0000001
```

Pantalla 3.27 Potencias obtenidas con el programa problema2_lms_d

```
potencia =

    0.2725


potencia1 =

    0.0141


potencia1 =

    0.0905


potencia3 =

    0.1217


potencia4 =

    0.0652
```

Pantalla 3.28 Ejemplo problema2_lms_d. Comparación de los pesos estimados

Ejemplo problema2_lms_d
Comparacion de los pesos estimados de una señal gaussiana con los pesos reales o actuales de una señal gaussiana utilizando un algoritmo lms, orden=5, numero de pesos=4

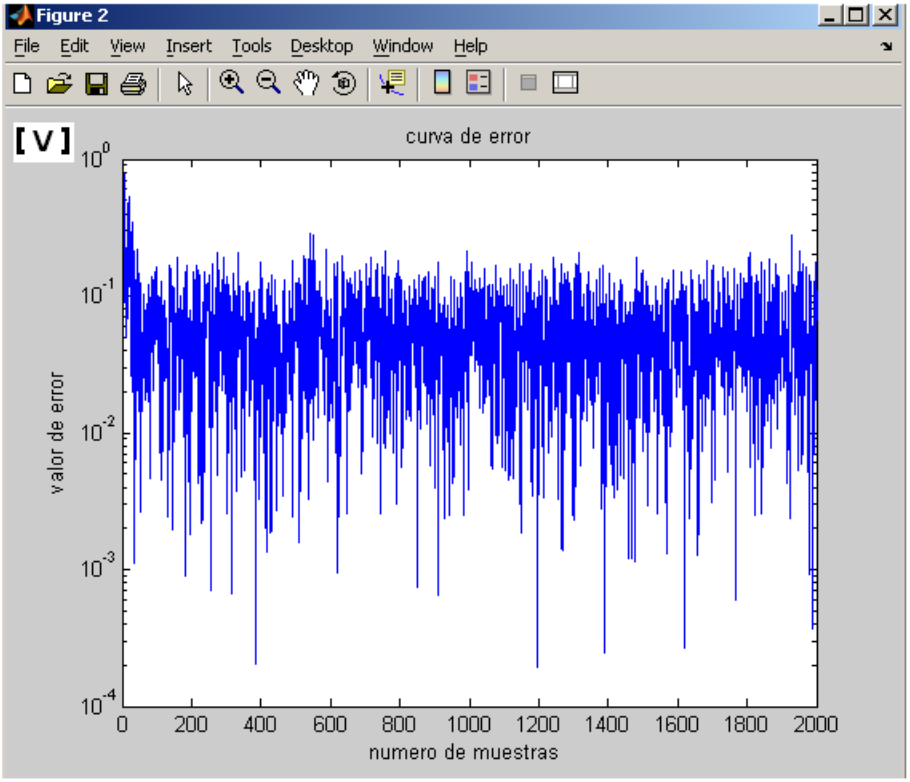


Eje de las x constante, en el eje de las y amplitud de la señal en voltios. Amplitud_señal vs Valor constante

Observamos que el error disminuyo inclusive en el peso1. Con lo que se cumple la teoría que el número de orden del filtro mientras más cercano es al número de valores reales o pesos más exacto y menor es el error. (para n=4 y orden=5)

Pantalla 3.29 Ejemplo problema2_lms_d. Curva de error de una señal gaussiana en un algoritmo lms.

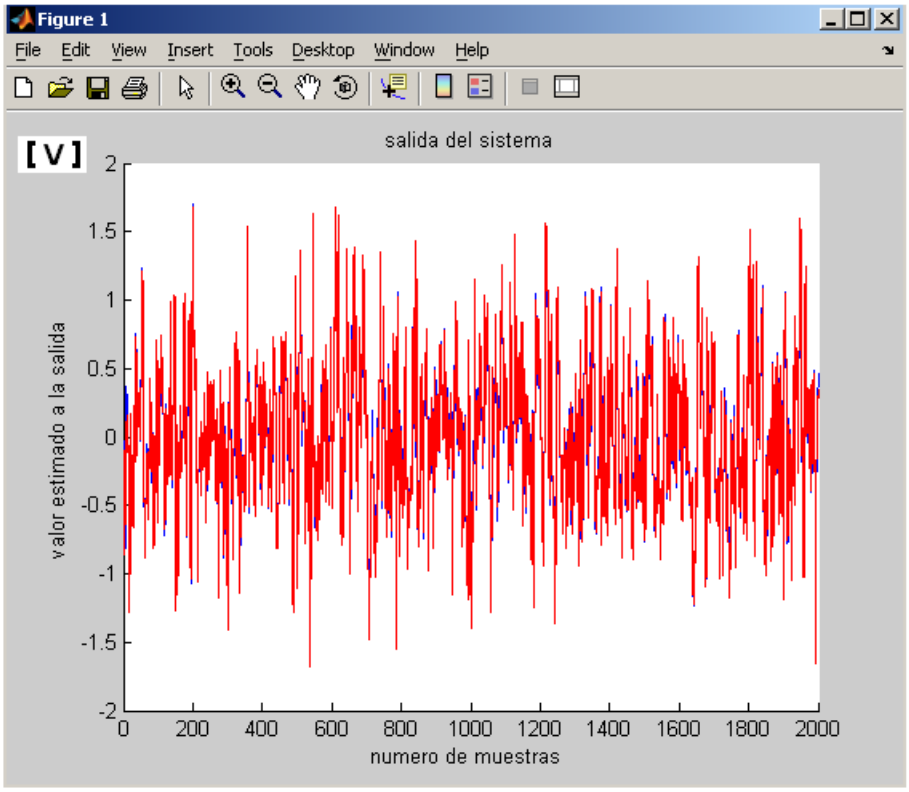
Ejemplo problema2_lms_d



Amplitud de la señal (voltios) vs numero de muestras

Pantalla 3.30 Ejemplo problema2_lms_d. Comparación entre la salida del sistema de una señal gaussiana (lms) con la señal de referencia del sistema.

Ejemplo problema2_lms_d



Amplitud de la señal (voltios) vs numero de muestras
señal de referencia (color azul)
señal obtenida por el algoritmo lms (color rojo)

Problema2_lms_e

Pantalla 3.31 Problema2_lms_e

```
* PROBLEMA LMS2.- Determinar los pesos, el numero de iteraciones y la potencia*.
* de un grupo de antenas inteligentes utilizando el algoritmo LMS (algoritmo  *.
* minimo cuadrado, las antenas estan distribuidas en forma de celdas para lo  *.
* cual requeriremos los siguientes datos:                                      *.
*                                                                              *.
*  • la señal aplicada a la entrada del filtro,   ej  Xn=sin((2*pi*k)/N)       *.
*  • el numero de antenas  x                                                   *.
*  • la señal de referencia, ejemplo d(k)=cos((2*pi*k)/N);                     *.
*  • el ruido de las diferentes antenas, (hay que tener presente que el        *.
*    algoritmo LMS en su estructura internamente produce un ruido del 10       *.
*    por ciento aparte de este ruido puede haber otros ruidos.                 *.
*  • el orden del filtro FIR cuyos coeficientes se van a ajustar, orden        *.
*  • el número de iteraciones en el aprendizaje, N (este número se hará        *.
*    hasta que cumpla el error)                                                *.
*  • la constante de ajuste   u                                               *.
*  • el vector de pesos inicial. Los voltajes de las antenas V(f)              *.
*  • el error deseado (puede ser ingresado por teclado) error                  *.
*                                                                              *.
*  A su salida devolverá:                                                      *.
*  • la salida en cada iteración durante el proceso de ajuste, y(t)            *.
*  • la curva de aprendizaje (grafico de la curva)                             *.
*  • los vectores de pesos obtenidos durante el proceso. W(t)                  *.
*  • la Potencia de la antena P(k)                                             *.
*  • el numero de iteraciones en que converge la curva y se cumple el valor    *.
*    del error. j                                                              *.
* alumno: Juan Francisco Alvarez Alvarado                                      *.
* alumno: Maribel del Rosario Chuez Gonzales                                   *.
*                                                                              *.
* Profesor :Ing. Pedro Vargas                                                  *.
*                                                                              *.
********************************************************************************.
teclee enter para proseguir
```

Pantalla 3.32 Datos del programa problema2_lms_e

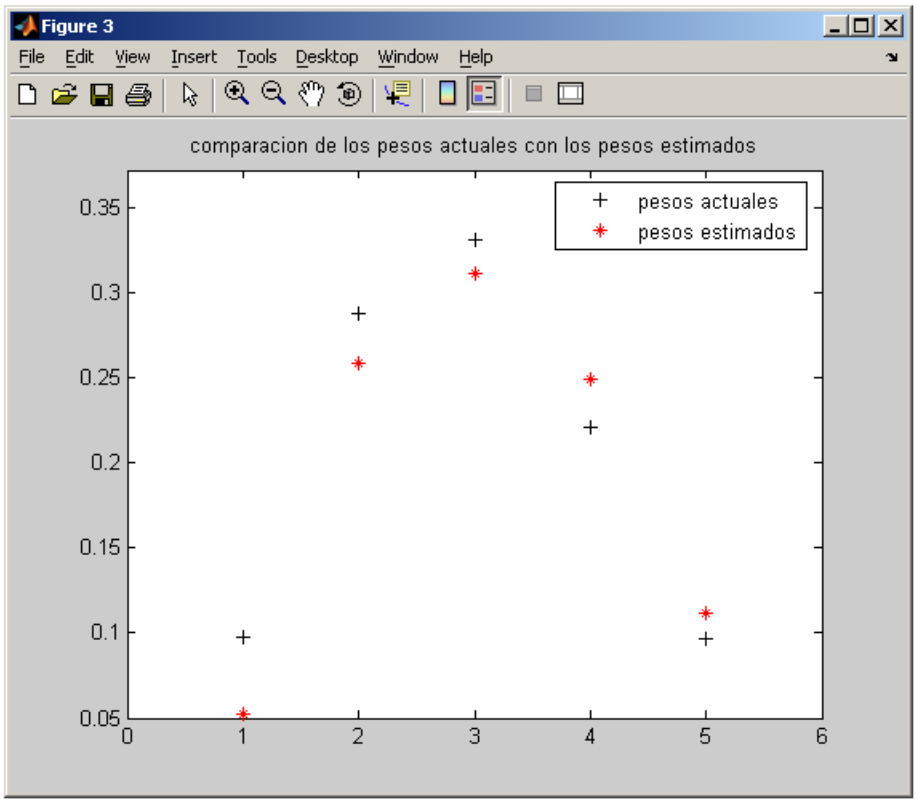
```
 ingrese el orden del filtro, cuyo coeficiente se va ha ajustar....5

orden =

     5

 ingrese el numero de antenas entre 1 y 5
 ingrese el numero de antenas entre 1,2,3,4 o 5....5
ingrese el voltaje1 (en v)..... voltaje1=0.0976
 ingrese la voltaje2 (en V).... voltaje2=0.2873
 ingrese la voltaje3 (en V).....voltaje3=0.3360
 ingrese la voltaje4 (en v)..... voltaje4=0.2210
 ingrese la voltaje5 (en v)..... voltaje5=0.0964
ingrese el ruido en la antena1.....ruido1=0.01
ingrese el ruido en la antena2.....ruido2=0.01
ingrese el ruido en la antena3.....ruido3=0.02
ingrese el ruido en la antena4.....ruido4=0.03
ingrese el ruido en la antena5.....ruido5=0.01
si desea que el ruido adicional vaya a la señal a la entrada digite 1
de lo contrario se considerara que el ruido adicional se agregara a la señal de referencia
opcion_ruido=1
ingrese el error del sistema=0.0000001
```

Pantalla 3.33 Potencias obtenidas con el programa problema2_lms_e

```
potencia =

    0.2778


potencia1 =

    0.0146


potencia2 =

    0.0918


potencia3 =

    0.1303


potencia4 =

    0.0663


potencia5 =

    0.0143
```

Pantalla 3.34 Ejemplo problema2_lms_e. Comparación de los pesos estimados

Ejemplo problema2_lms_e
Comparacion de los pesos estimados de una señal gaussiana con los pesos reales o actuales de una señal gaussiana utilizando un algoritmo lms, orden=5, numero de pesos=5

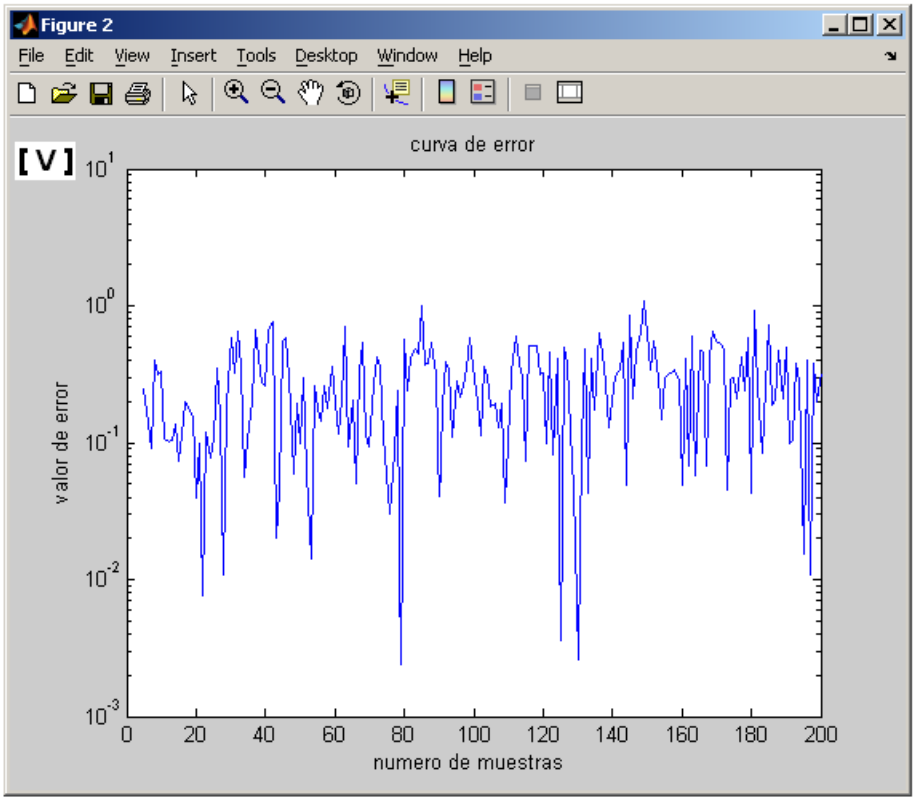


Eje de la x constante, en el eje de las y amplitud de la señal en voltios. Amplitud_señal vs Valor constante.

Se nota un pequeño aumento del error al disminuir el número de muestras. Con lo cual se cumple lo que dice la teoría del algoritmo LMS.

Pantalla 3.35 Ejemplo problema2_lms_e. Curva de error de una señal gaussiana en un algoritmo lms.

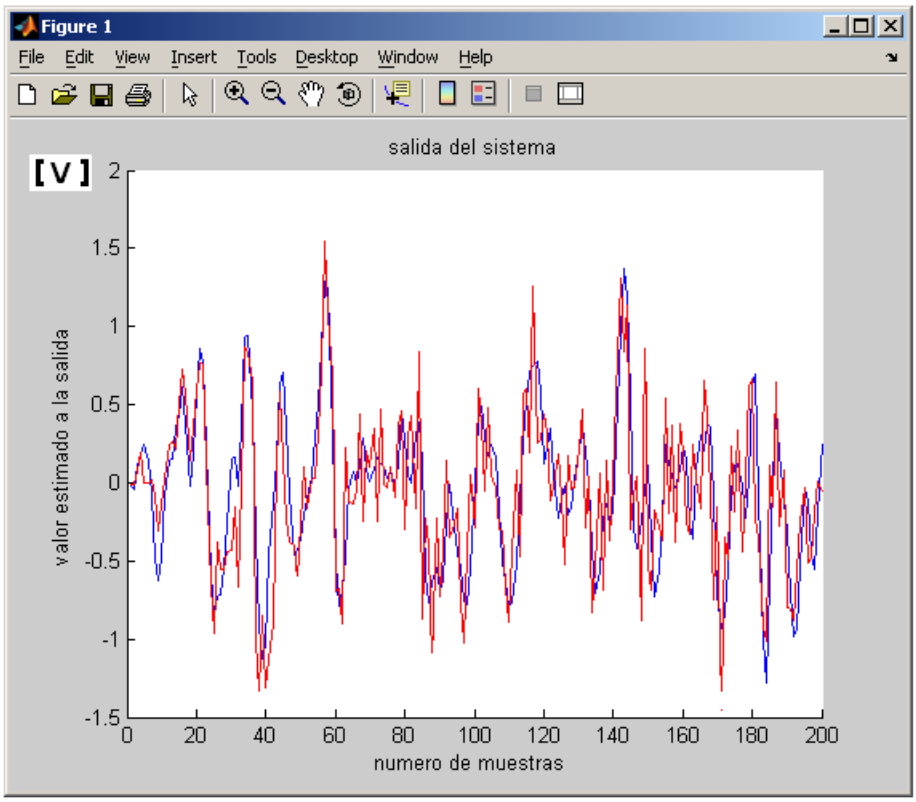
Ejemplo problema2_lms_e



Amplitud de la señal (voltios) vs numero de muestras

Pantalla 3.36 Ejemplo problema2_lms_e. Comparación entre la salida del sistema de una señal gaussiana (lms) con la señal de referencia del sistema.

Ejemplo problema2_lms_e



Amplitud de la señal (voltios) vs numero de muestras
señal de referencia (color azul)
señal obtenida por el algoritmo lms (color rojo)

Problema2_lms_f

Pantalla 3.37 Problema2_lms_f

```
   * PROBLEMA LMS2.- Determinar los pesos, el numero de iteraciones y la potencia*.
   * de un grupo de antenas inteligentes utilizando el algoritmo LMS (algoritmo  *.
   * minimo cuadrado, las antenas estan distribuidas en forma de celdas para lo  *.
   * cual requeriremos los siguientes datos:                                      *.
   *                                                                             *.
   *  • la señal aplicada a la entrada del filtro,   ej  Xn=sin((2*pi*k)/N)      *.
   *  • el numero de antenas  x                                                  *.
   *  • la señal de referencia, ejemplo d(k)=cos((2*pi*k)/N);                    *.
   *  • el ruido de las diferentes antenas, (hay que tener presente que el       *.
   *    algoritmo LMS en su estructura internamente produce un ruido del 10      *.
   *    por ciento aparte de este ruido puede haber otros ruidos.                *.
   *  • el orden del filtro FIR cuyos coeficientes se van a ajustar, orden       *.
   *  • el número de iteraciones en el aprendizaje, N (este número se hará       *.
   *    hasta que cumpla el error)                                               *.
   *  • la constante de ajuste   u                                               *.
   *  • el vector de pesos inicial. Los voltajes de las antenas V(f)             *.
   *  • el error deseado (puede ser ingresado por teclado) error                 *.
   *                                                                             *.
   *  A su salida devolverá:                                                     *.
   *  • la salida en cada iteración durante el proceso de ajuste, y(t)           *.
   *  • la curva de aprendizaje (grafico de la curva)                            *.
   *  • los vectores de pesos obtenidos durante el proceso. W(t)                 *.
   *  • la Potencia de la antena P(k)                                            *.
   *  • el numero de iteraciones en que converge la curva y se cumple el valor   *.
   *    del error. j                                                             *.
   * alumno: Juan Francisco Alvarez Alvarado                                     *.
   * alumno: Maribel del Rosario Chuez Gonzales                                  *.
   *                                                                             *.
   * Profesor :Ing. Pedro Vargas                                                 *.
   *                                                                             *.
   *******************************************************************************.
teclee enter para proseguir
```

Pantalla 3.38 Datos del programa problema2_lms_f

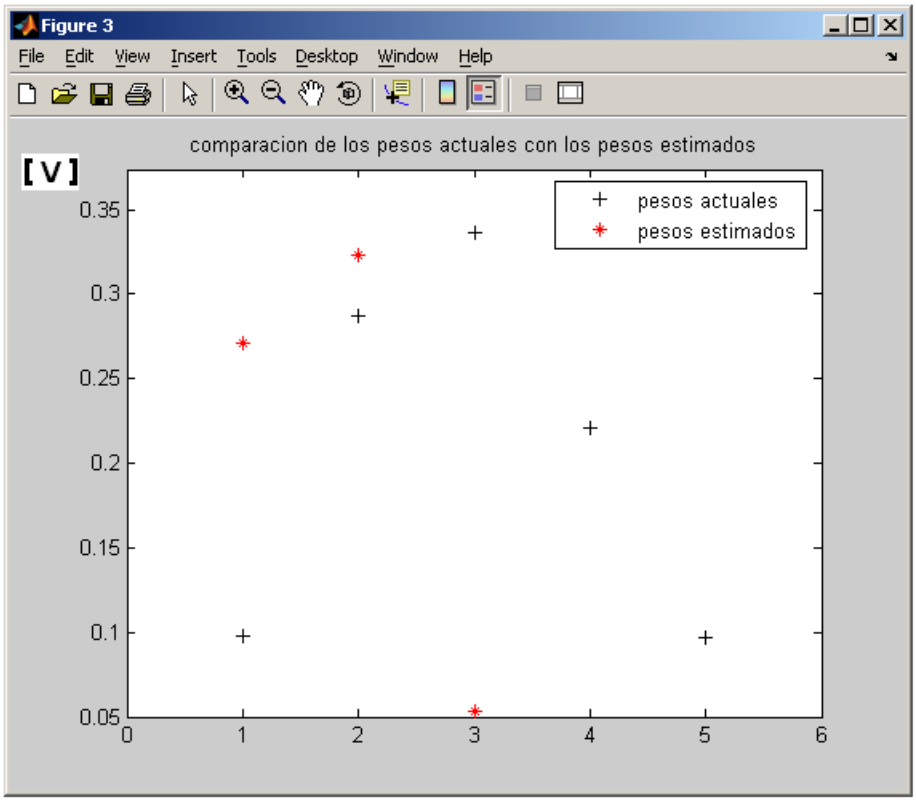
```
 ingrese el orden del filtro, cuyo coeficiente se va ha ajustar....5

orden =

     5

 ingrese el numero de antenas entre 1 y 5
 ingrese el numero de antenas entre 1,2,3,4 o 5....4
ingrese el voltaje1 (en v)..... voltaje1=0.0976
 ingrese la voltaje2 (en V).... voltaje2=0.2873
 ingrese la voltaje3 (en V).....voltaje3=0.3360
 ingrese la voltaje4 (en v)..... voltaje4=0.2210
ingrese el ruido en la antena1.....ruido1=0.01
ingrese el ruido en la antena2.....ruido2=0.01
ingrese el ruido en la antena3.....ruido3=0.02
ingrese el ruido en la antena4.....ruido4=0.03
si desea que el ruido adicional vaya a la señal a la entrada digite 1
de lo contrario se considerara que el ruido adicional se agregara a la señal de referencia
opcion_ruido=1
ingrese el error del sistema=0.0000001
```

Pantalla 3.39 Potencias obtenidas con el programa problema2_lms_f

```
potencia =

    0.2725


potencia1 =

    0.0141


potencia1 =

    0.0905


potencia3 =

    0.1217


potencia4 =

    0.0652
```

Pantalla 3.40 Ejemplo problema2_lms_f. Comparación de los pesos estimados

Ejemplo problema2_lms_f
Comparacion de los pesos estimados de una señal gaussiana con los pesos reales o actuales de una señal gaussiana utilizando un algoritmo lms, orden=5, numero de pesos=4



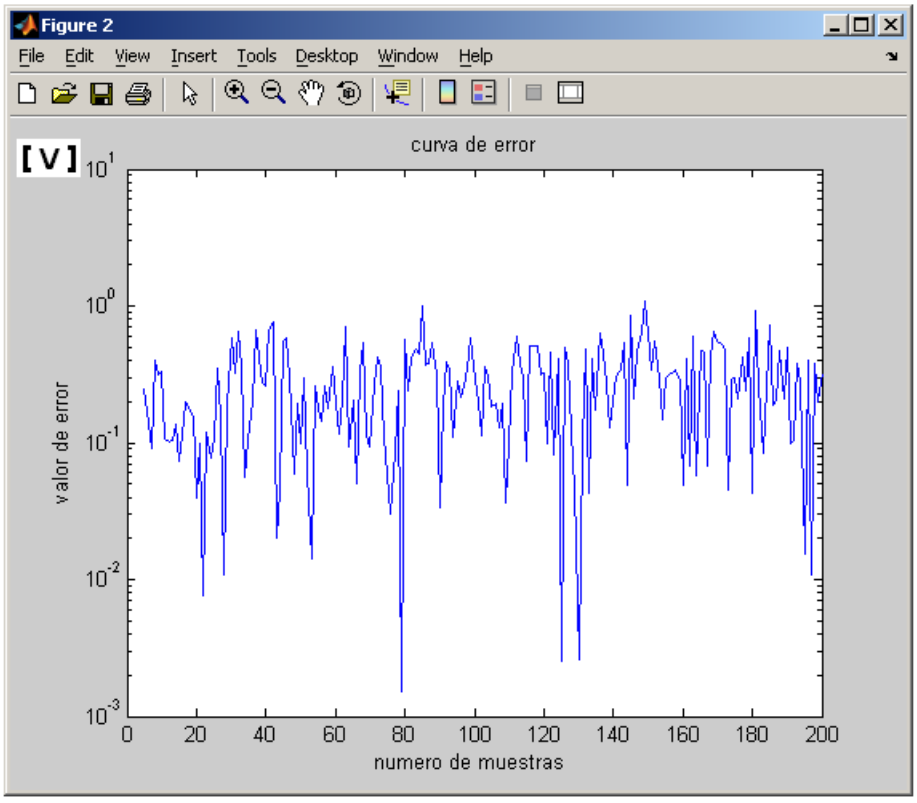
Eje de la x constante, en el eje de las y amplitud de la señal en voltios. Amplitud_señal vs Valor constante.

Vemos que el error aumento considerablemente para mu=0.30 y 0.50.

Sería conveniente hacer una tabla de valores de mu, y ver hasta que valores el error es pequeño en este caso se ve que si se aumentara solamente hasta mu=1 todos los pesos estimados serian diferentes de los pesos reales.

Pantalla 3.41 Ejemplo problema2_lms_f. Curva de error de una señal gaussiana en un algoritmo lms

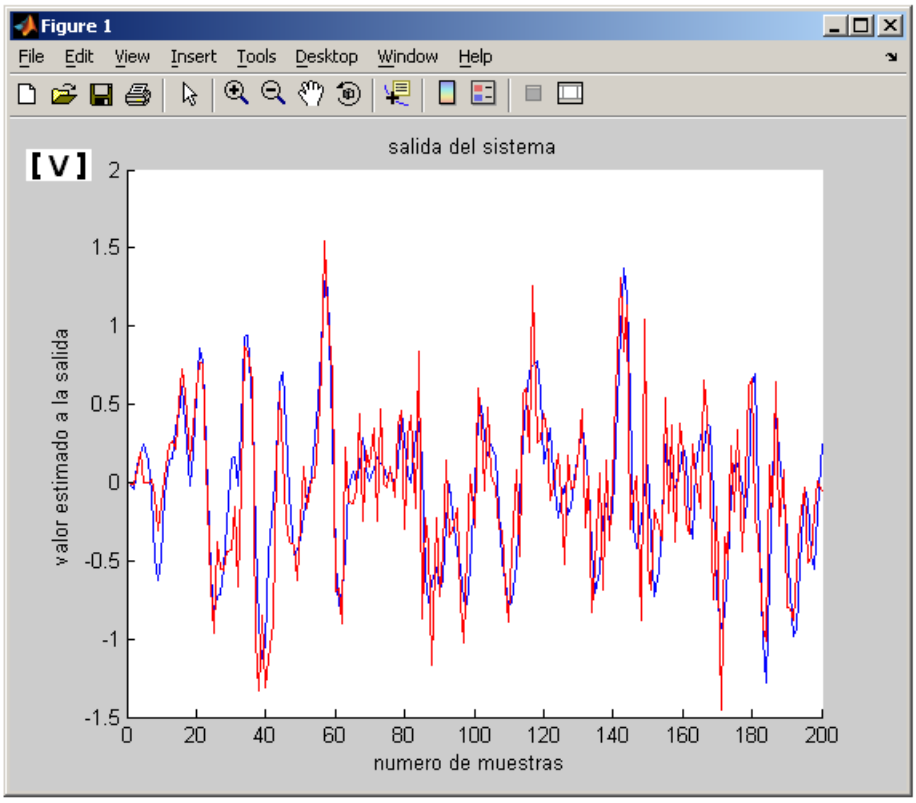
Ejemplo problema2_lms_f



Amplitud de la señal (voltios) vs numero de muestras

Pantalla 3.42 Ejemplo problema2_lms_f. Comparación entre la salida del sistema de una señal gaussiana (lms) con la señal de referencia del sistema.

Ejemplo problema2_lms_f



Amplitud de la señal (voltios) vs numero de muestras
señal de referencia (color azul)
señal obtenida por el algoritmo lms (color rojo)

Problema2_lms_g

Pantalla 3.43 Problema2_lms_g

```
   * PROBLEMA LMS2.- Determinar los pesos, el numero de iteraciones y la potencia*.
   * de un grupo de antenas inteligentes utilizando el algoritmo LMS (algoritmo  *.
   * minimo cuadrado, las antenas estan distribuidas en forma de celdas para lo  *.
   * cual requeriremos los siguientes datos:                                      *.
   *                                                                             *.
   *  • la señal aplicada a la entrada del filtro,   ej  Xn=sin((2*pi*k)/N)      *.
   *  • el numero de antenas  x                                                  *.
   *  • la señal de referencia, ejemplo d(k)=cos((2*pi*k)/N);                    *.
   *  • el ruido de las diferentes antenas, (hay que tener presente que el       *.
   *    algoritmo LMS en su estructura internamente produce un ruido del 10      *.
   *    por ciento aparte de este ruido puede haber otros ruidos.                *.
   *  • el orden del filtro FIR cuyos coeficientes se van a ajustar, orden       *.
   *  • el número de iteraciones en el aprendizaje, N (este número se hará       *.
   *    hasta que cumpla el error)                                               *.
   *  • la constante de ajuste   u                                               *.
   *  • el vector de pesos inicial. Los voltajes de las antenas V(f)             *.
   *  • el error deseado (puede ser ingresado por teclado) error                 *.
   *                                                                             *.
   *  A su salida devolverá:                                                     *.
   *  • la salida en cada iteración durante el proceso de ajuste, y(t)           *.
   *  • la curva de aprendizaje (grafico de la curva)                            *.
   *  • los vectores de pesos obtenidos durante el proceso. W(t)                 *.
   *  • la Potencia de la antena P(k)                                            *.
   *  • el numero de iteraciones en que converge la curva y se cumple el valor   *.
   *    del error. j                                                             *.
   * alumno: Juan Francisco Alvarez Alvarado                                     *.
   * alumno: Maribel del Rosario Chuez Gonzales                                  *.
   *                                                                             *.
   * Profesor :Ing. Pedro Vargas                                                 *.
   *                                                                             *.
   *******************************************************************************.
teclee enter para proseguir
```

Pantalla 3.44 Datos del programa problema2_lms_g

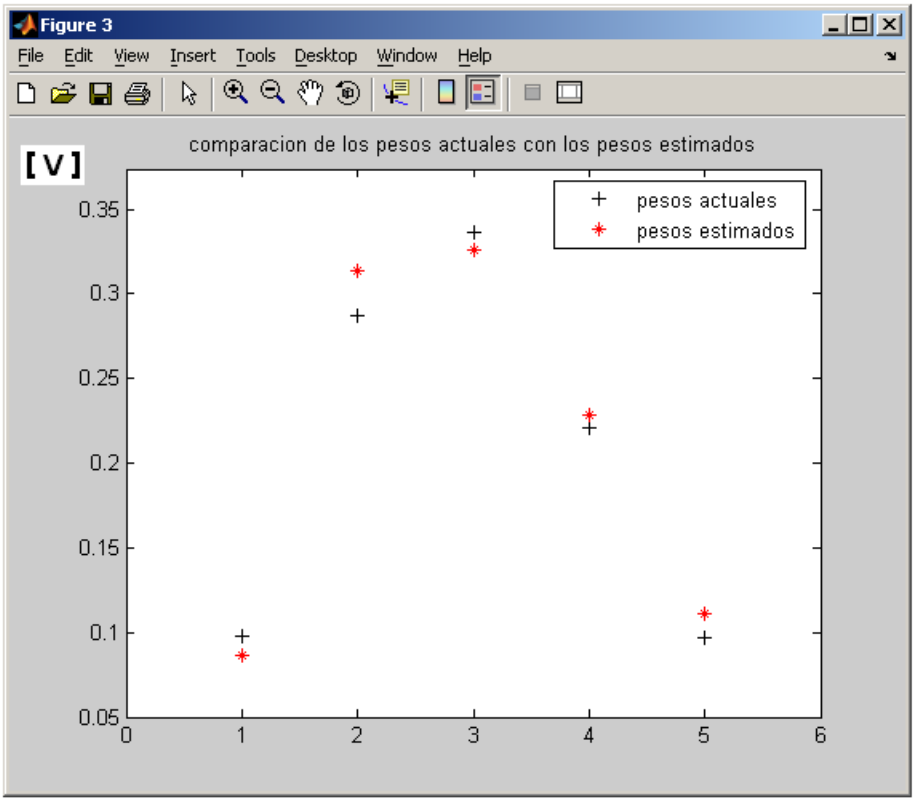
```
 ingrese el orden del filtro, cuyo coeficiente se va ha ajustar....5

orden =

     5

 ingrese el numero de antenas entre 1 y 5
 ingrese el numero de antenas entre 1,2,3,4 o 5....4
ingrese el voltaje1 (en v)..... voltaje1=0.0976
 ingrese la voltaje2 (en V).... voltaje2=0.2873
 ingrese la voltaje3 (en V).....voltaje3=0.3360
 ingrese la voltaje4 (en v)..... voltaje4=0.2210
ingrese el ruido en la antena1.....ruido1=0.01
ingrese el ruido en la antena2.....ruido2=0.01
ingrese el ruido en la antena3.....ruido3=0.02
ingrese el ruido en la antena4.....ruido4=0.03
si desea que el ruido adicional vaya a la señal a la entrada digite 1
de lo contrario se considerara que el ruido adicional se agregara a la señal de referencia
opcion_ruido=1
ingrese el error del sistema=0.0000001
```

Pantalla 3.45 Potencias obtenidas con el programa problema2_lms_g

```
potencia =

    0.2725


potencia1 =

    0.0141


potencia1 =

    0.0905


potencia3 =

    0.1217


potencia4 =

    0.0652
```

Pantalla 3.46 Ejemplo problema2_lms_g. Comparación de los pesos estimados

Ejemplo problema2_lms_g
Comparacion de los pesos estimados de una señal gaussiana con los pesos reales o actuales de una señal gaussiana utilizando un algoritmo lms, orden=5, numero de pesos=4

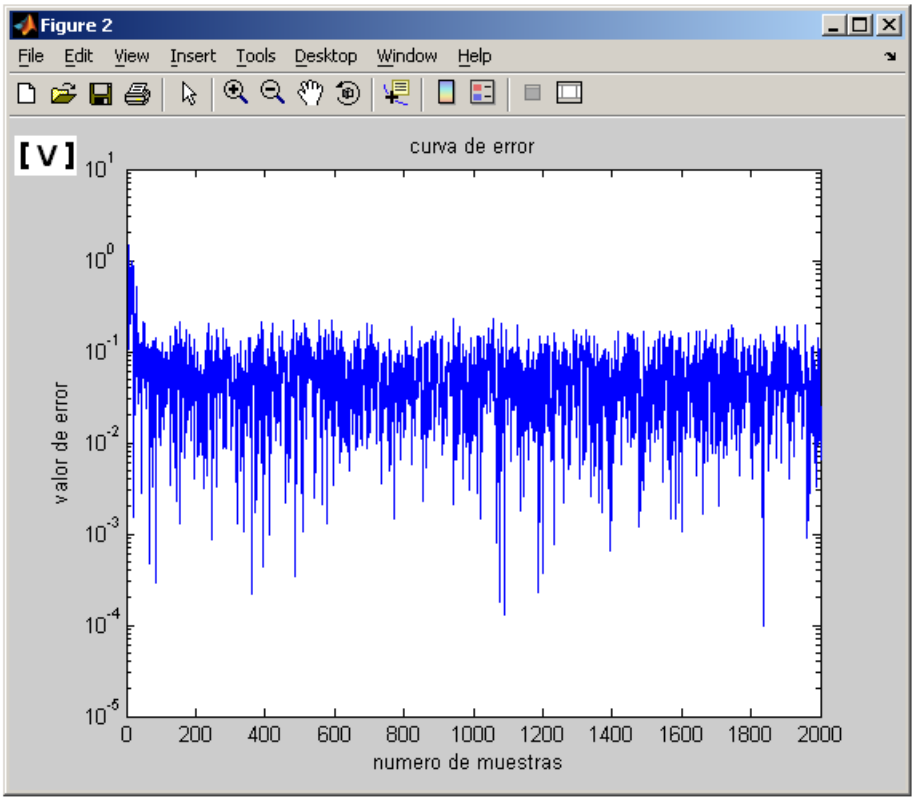


Eje de la x constante, en el eje de las y amplitud de la señal en voltios. Amplitud_señal vs Valor constante.

Se ve que el error es casi igual a los valores que obtenemos con un valor de mu=0.32 y mu=0.25 recomendados por la central matlab

Pantalla 3.47 Ejemplo problema2_lms_g. Curva de error de una señal gaussiana en un algoritmo lms

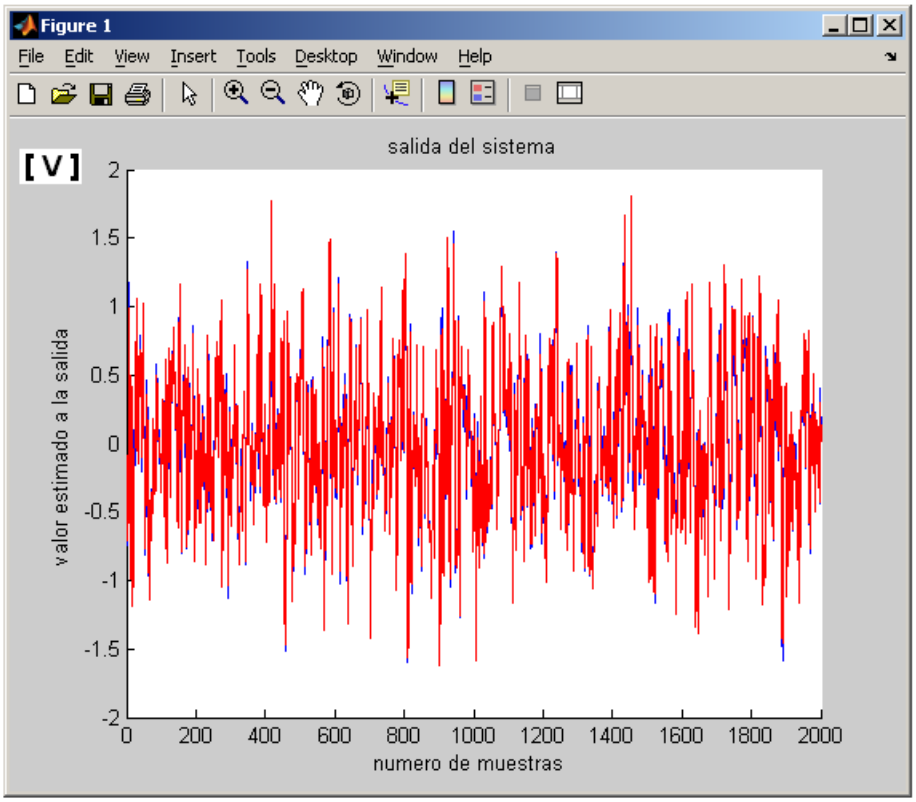
Ejemplo problema2_lms_g



Amplitud de la señal (voltios) vs numero de muestras

Pantalla 3.48 Ejemplo problema2_lms_g. Comparación entre la salida del sistema de una señal gaussiana (lms) con la señal de referencia del sistema

Ejemplo problema2_lms_g



Amplitud de la señal (voltios) vs numero de muestras
señal de referencia (color azul)
señal obtenida por el algoritmo lms (color rojo)

Problema2_lms_h

Pantalla 3.49 Problema2_lms_h

```
   * PROBLEMA LMS2.- Determinar los pesos, el numero de iteraciones y la potencia*.
   * de un grupo de antenas inteligentes utilizando el algoritmo LMS (algoritmo  *.
   * minimo cuadrado, las antenas estan distribuidas en forma de celdas para lo  *.
   * cual requeriremos los siguientes datos:                                      *.
   *                                                                             *.
   *  • la señal aplicada a la entrada del filtro,   ej  Xn=sin((2*pi*k)/N)      *.
   *  • el numero de antenas  x                                                  *.
   *  • la señal de referencia, ejemplo d(k)=cos((2*pi*k)/N);                    *.
   *  • el ruido de las diferentes antenas, (hay que tener presente que el       *.
   *    algoritmo LMS en su estructura internamente produce un ruido del 10      *.
   *    por ciento aparte de este ruido puede haber otros ruidos.                *.
   *  • el orden del filtro FIR cuyos coeficientes se van a ajustar, orden       *.
   *  • el número de iteraciones en el aprendizaje, N (este número se hará       *.
   *    hasta que cumpla el error)                                               *.
   *  • la constante de ajuste   u                                               *.
   *  • el vector de pesos inicial. Los voltajes de las antenas V(f)             *.
   *  • el error deseado (puede ser ingresado por teclado) error                 *.
   *                                                                             *.
   *  A su salida devolverá:                                                     *.
   *  • la salida en cada iteración durante el proceso de ajuste, y(t)           *.
   *  • la curva de aprendizaje (grafico de la curva)                            *.
   *  • los vectores de pesos obtenidos durante el proceso. W(t)                 *.
   *  • la Potencia de la antena P(k)                                            *.
   *  • el numero de iteraciones en que converge la curva y se cumple el valor   *.
   *    del error. j                                                             *.
   * alumno: Juan Francisco Alvarez Alvarado                                     *.
   * alumno: Maribel del Rosario Chuez Gonzales                                  *.
   *                                                                             *.
   * Profesor :Ing. Pedro Vargas                                                 *.
   *                                                                             *.
   *******************************************************************************.
teclee enter para proseguir
```

Pantalla 3.50 Datos del programa problema2_lms_h

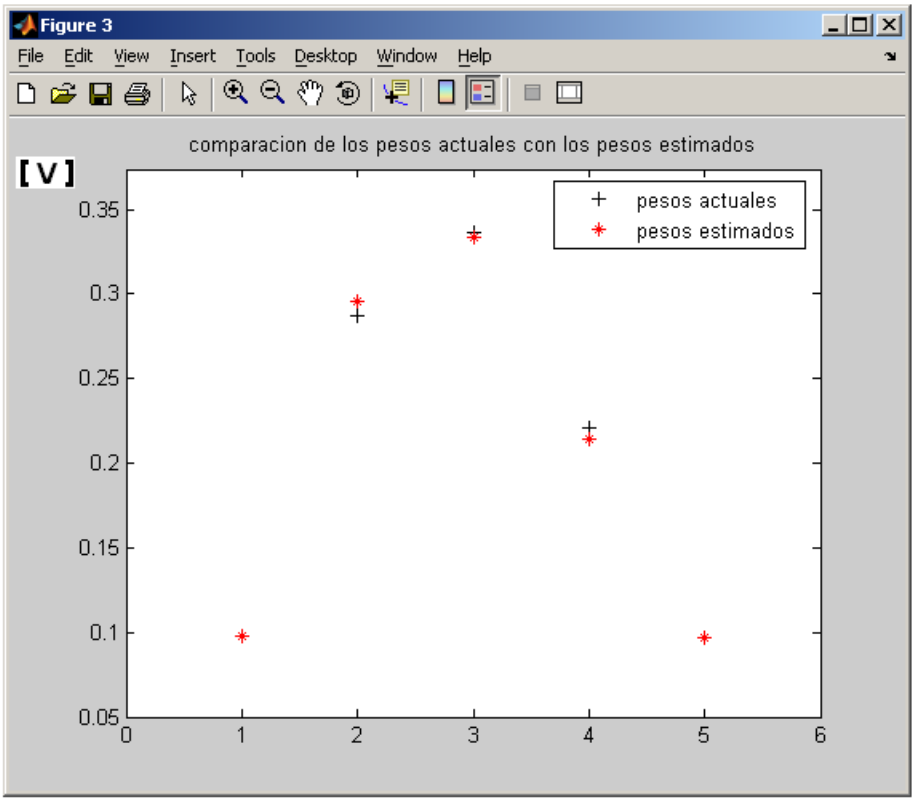
```
 ingrese el orden del filtro, cuyo coeficiente se va ha ajustar....5

orden =

     5

 ingrese el numero de antenas entre 1 y 5
 ingrese el numero de antenas entre 1,2,3,4 o 5....5
ingrese el voltaje1 (en v)..... voltaje1=0.0976
 ingrese la voltaje2 (en V).... voltaje2=0.2873
 ingrese la voltaje3 (en V).....voltaje3=0.3360
 ingrese la voltaje4 (en v)..... voltaje4=0.2210
 ingrese la voltaje5 (en v)..... voltaje5=0.0964
ingrese el ruido en la antena1.....ruido1=0.01
ingrese el ruido en la antena2.....ruido2=0.01
ingrese el ruido en la antena3.....ruido3=0.02
ingrese el ruido en la antena4.....ruido4=0.03
ingrese el ruido en la antena5.....ruido5=0.01
si desea que el ruido adicional vaya a la señal a la entrada digite 1
de lo contrario se considerara que el ruido adicional se agregara a la señal de referencia
opcion_ruido=2
ingrese el error del sistema=0.0000001
```

Pantalla 3.51 Potencias obtenidas con el programa problema2_lms_h

```
potencia =

    0.2778


potencia1 =

    0.0146


potencia2 =

    0.0920


potencia3 =

    0.1303


potencia4 =

    0.0663


potencia5 =

    0.0143
```

Pantalla 3.52 Ejemplo problema2_lms_h. Comparación de los pesos estimados

Ejemplo problema2_lms_h
Comparacion de los pesos estimados de una señal gaussiana con los pesos reales o actuales de una señal gaussiana utilizando un algoritmo lms, orden=5, numero de pesos=5



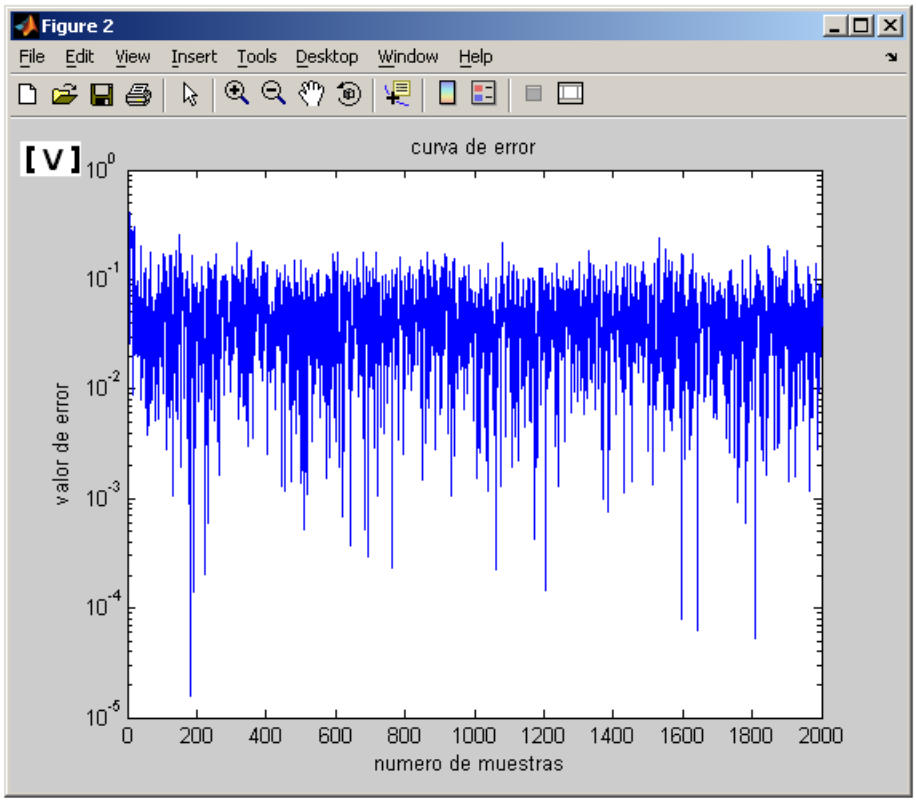
Eje de las x constante, en el eje de la y amplitud de la señal en voltios. Amplitud_señal vs Valor constante.

Vemos un aumento considerable del error para el primer valor y para el quinto, pero casi exacto para los valores de 2, 3 y 4. Parece ser que el algoritmo LMS acepta mejor los ruidos adicionales a la entrada del filtro y no tanto a su salida. Esto tiene sentido porque al ingresar el ruido adicional a la entrada del filtro,

este valor es filtrado y debe disminuir el error considerablemente. Además por la teoría del algoritmo LMS se puede disminuir este error un poco con un mayor número de muestras, y con valores de la constante de ajuste mu que le permitan ser más exacto, pero sería bastante cansado buscar para que valores de mu salen valores más exactos.

Pantalla 3.53 Ejemplo problema2_lms_h. Curva de error de una señal gaussiana en un algoritmo lms

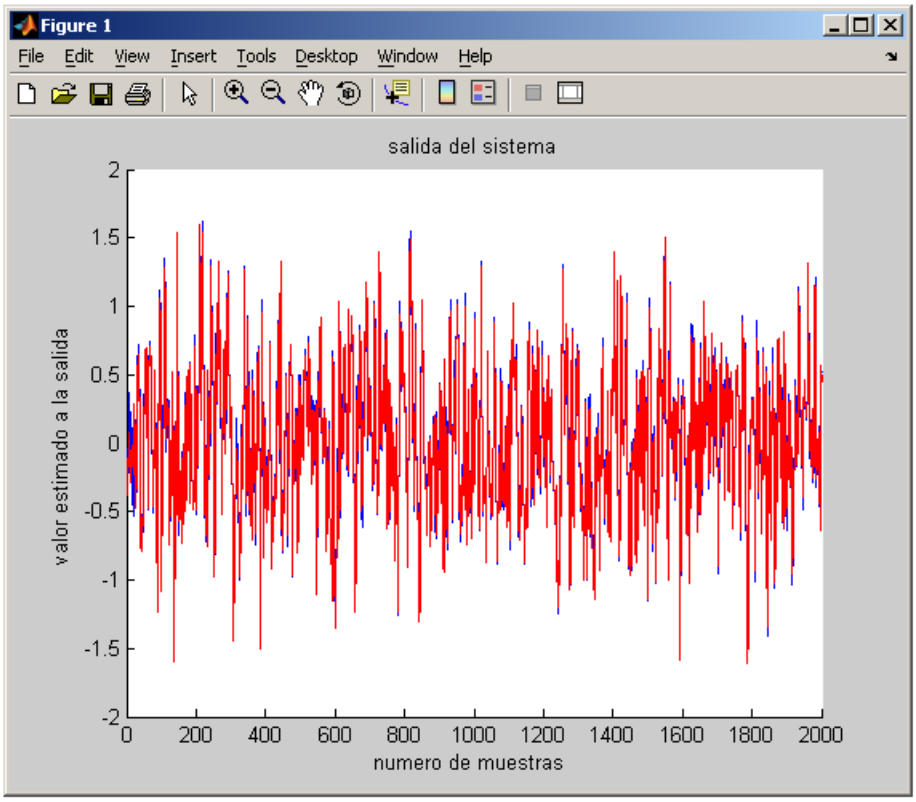
Ejemplo problema2_lms_h



Amplitud de la señal (voltios) vs numero de muestras

Pantalla 3.54 Ejemplo problema2_lms_h. Comparación entre la salida del sistema de una señal gaussiana (lms) con la señal de referencia del sistema

Ejemplo problema2_lms_h



Amplitud de la señal (voltios) vs numero de muestras
señal de referencia (color azul)
señal obtenida por el algoritmo lms (color rojo)

Problema1_lms_a

Pantalla 3.55 Problema1_lms_a

```
* PROBLEMA1_lms.- Determinar los pesos, el numero de iteraciones y la potencia*.
* de un grupo de antenas inteligentes utilizando el algoritmo LMS (algoritmo  *.
* minimo cuadrado, las antenas estan distribuidas en forma de celdas para lo  *.
* cual requeriremos los siguientes datos:                                      *.
*                                                                              *.
*  • la señal aplicada a la entrada del filtro,   ej  Xn=2*sin((2*pi*k)/N)     *.
*  • el numero de antenas  x                                                   *.
*  • la señal de referencia, d(k)=2*cos((2*pi*k)/N);                           *.
*  • el ruido de las diferentes antenas, (hay que tener presente que el        *.
*    algoritmo LMS en su estructura internamente produce un ruido del 10       *.
*    aparte de este ruido puede haber otros.  n(f)                             *.
*  • el orden del filtro FIR cuyos coeficientes se van a ajustar, orden        *.
*  • el número de iteraciones en el aprendizaje, N (este número se hará        *.
*    hasta que cumpla el error)                                                *.
*  • la constante de ajuste   u                                                *.
*  • el vector de pesos inicial. Los voltajes de las antenas V(f)              *.
*  • el error deseado (puede ser ingresado por teclado) error                  *.
*                                                                              *.
*  A su salida devolverá:                                                      *.
*  • la salida en cada iteración durante el proceso de ajuste, y(t)            *.
*  • la curva de aprendizaje (grafico de la curva)                             *.
*  • los vectores de pesos obtenidos durante el proceso. W(t)                  *.
*  • la Potencia de la antena P(k)                                             *.
*  • el numero de iteraciones en que converge la curva y se cumple el valor    *.
*    del error.  j                                                             *.
* alumno: Juan Francisco Alvarez Alvarado                                      *.
* alumno: Maribel del Rosario Chuez Gonzales                                   *.
*                                                                              *.
* Profesor :Ing. Pedro Vargas                                                  *.
```

Pantalla 3.56 Datos del programa problema1_lms_a

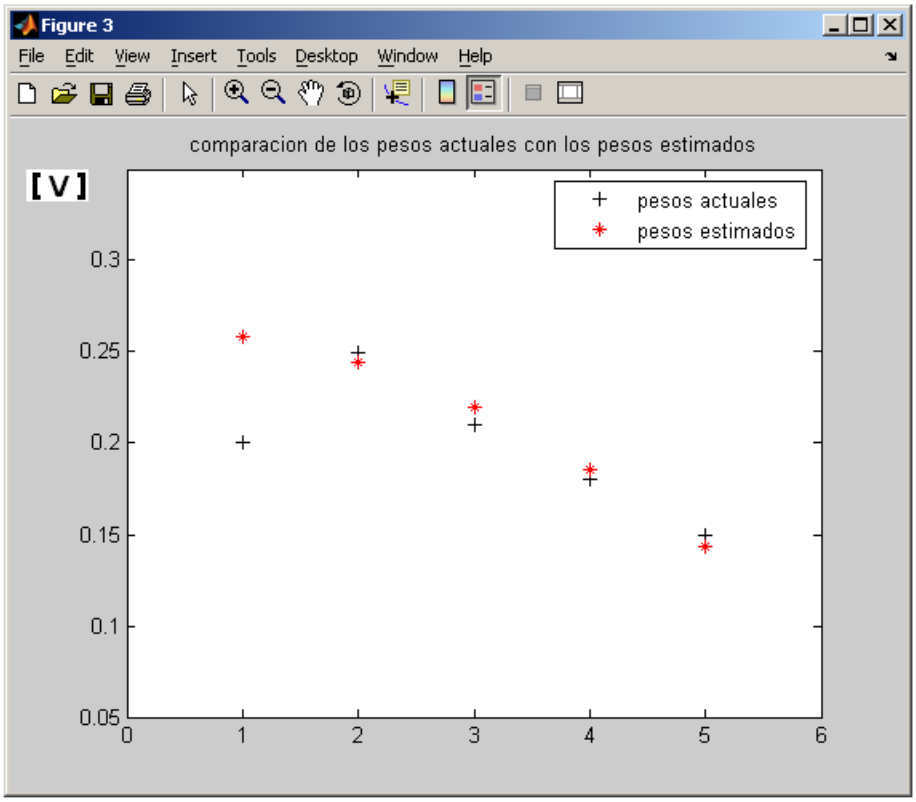
```
 ingrese el orden del filtro, cuyo coeficiente se va ha ajustar....5

orden =

     5

 ingrese el numero de antenas entre 1 y 5
 ingrese el numero de antenas entre 1,2,3,4 o 5....5
ingrese el voltaje1 (en v)..... voltaje1=0.20
 ingrese la voltaje2 (en V).... voltaje2=0.25
 ingrese la voltaje3 (en V).....voltaje3=0.21
 ingrese la voltaje4 (en v)..... voltaje4=0.18
 ingrese la voltaje5 (en v)..... voltaje5=0.15
ingrese el ruido en la antena1.....ruido1=0.01
ingrese el ruido en la antena2.....ruido2=0.01
ingrese el ruido en la antena3.....ruido3=0.01
ingrese el ruido en la antena4.....ruido4=0.01
ingrese el ruido en la antena5.....ruido5=0.01
si desea que el ruido adicional vaya a la señal a la entrada digite 1
 de lo contrario se considerara que el ruido adicional se agregara a la señal de referencia
opcion_ruido=1
ingrese el error del sistema=0.0000001
```

Pantalla 3.57 Potencias obtenidas con el programa problema1_lms_a

```
totallength =

   200


potencia =

    0.4943


potencia1 =

    0.0484


potencia2 =

    0.0718


potencia3 =

    0.0527


potencia4 =

    0.0405


potencia5 =

    0.0301
```

Pantalla 3.58 Ejemplo problema1_lms_a. Comparación de los pesos estimados

Ejemplo problema1_lms_a
Comparacion de los pesos estimados de una señal senosoidal con los pesos reales o actuales de una señal senosoidal utilizando un algoritmo lms, orden=5, numero de pesos=5.

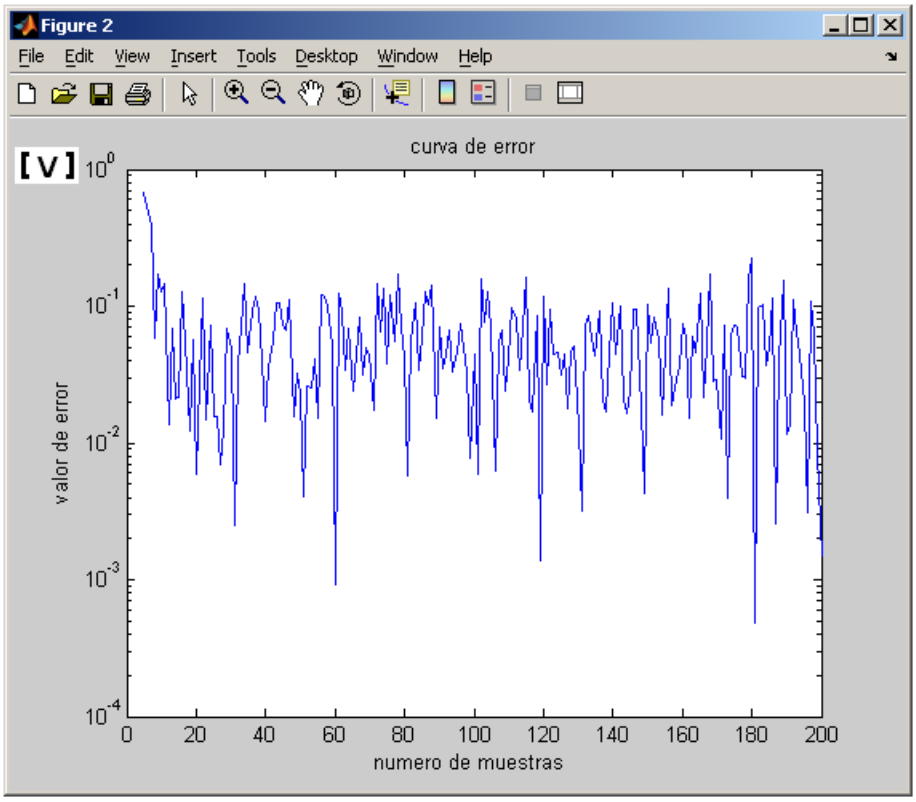


Eje de las x constante, en el eje de las y amplitud de la señal en voltios. Amplitud_señal vs Valor constante.

Vemos un pequeño error para el primer valor ingresado pero para los siguientes valores el error es casi inexistente.

Pantalla 3.59 Ejemplo problema1_lms_a. Curva de error de una señal senosoidal en un algoritmo lms

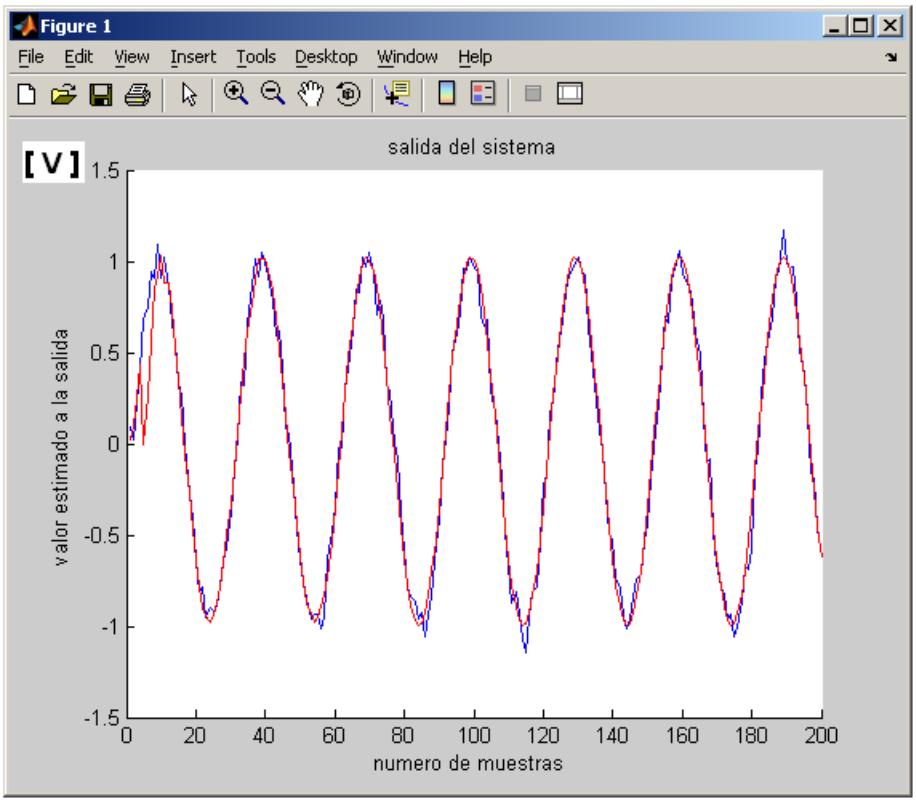
Ejemplo problema1_lms_a



Amplitud de la señal (voltios) vs numero de muestras

Pantalla 3.60 Ejemplo problema1_lms_a. Comparación entre la salida del sistema de una señal senosoidal (lms) con la señal de referencia del sistema

Ejemplo problema1_lms_a



Amplitud de la señal (voltios) vs numero de muestras
señal de referencia (color azul)
señal obtenida por el algoritmo lms (color rojo)

Ahora utilizaremos el mismo ejemplo con la misma señal de entrada como señal de referencia con un pequeño ruido adicional, pero con un numero de muestras de N=400 en teoría con el mayor numero de muestras el error debe de disminuir un poco.

Problema1_lms_b

Pantalla 3.61 Problema1_lms_b

```
* PROBLEMA1_lms.- Determinar los pesos, el numero de iteraciones y la potencia*.
* de un grupo de antenas inteligentes utilizando el algoritmo LMS (algoritmo  *.
* minimo cuadrado, las antenas estan distribuidas en forma de celdas para lo  *.
* cual requeriremos los siguientes datos:                                      *.
*                                                                              *.
*  • la señal aplicada a la entrada del filtro,   ej  Xn=2*sin((2*pi*k)/N)     *.
*  • el numero de antenas  x                                                   *.
*  • la señal de referencia, d(k)=2*cos((2*pi*k)/N);                           *.
*  • el ruido de las diferentes antenas, (hay que tener presente que el        *.
*    algoritmo LMS en su estructura internamente produce un ruido del 10       *.
*    aparte de este ruido puede haber otros.  n(f)                             *.
*  • el orden del filtro FIR cuyos coeficientes se van a ajustar, orden        *.
*  • el número de iteraciones en el aprendizaje, N (este número se hará        *.
*    hasta que cumpla el error)                                                *.
*  • la constante de ajuste   u                                                *.
*  • el vector de pesos inicial. Los voltajes de las antenas V(f)              *.
*  • el error deseado (puede ser ingresado por teclado) error                  *.
*                                                                              *.
*  A su salida devolverá:                                                      *.
*  • la salida en cada iteración durante el proceso de ajuste, y(t)            *.
*  • la curva de aprendizaje (grafico de la curva)                             *.
*  • los vectores de pesos obtenidos durante el proceso. W(t)                  *.
*  • la Potencia de la antena P(k)                                             *.
*  • el numero de iteraciones en que converge la curva y se cumple el valor    *.
*    del error.  j                                                             *.
* alumno: Juan Francisco Alvarez Alvarado                                      *.
* alumno: Maribel del Rosario Chuez Gonzales                                   *.
*                                                                              *.
* Profesor :Ing. Pedro Vargas                                                  *.
```

Pantalla 3.62 Datos del programa problema1_lms_b

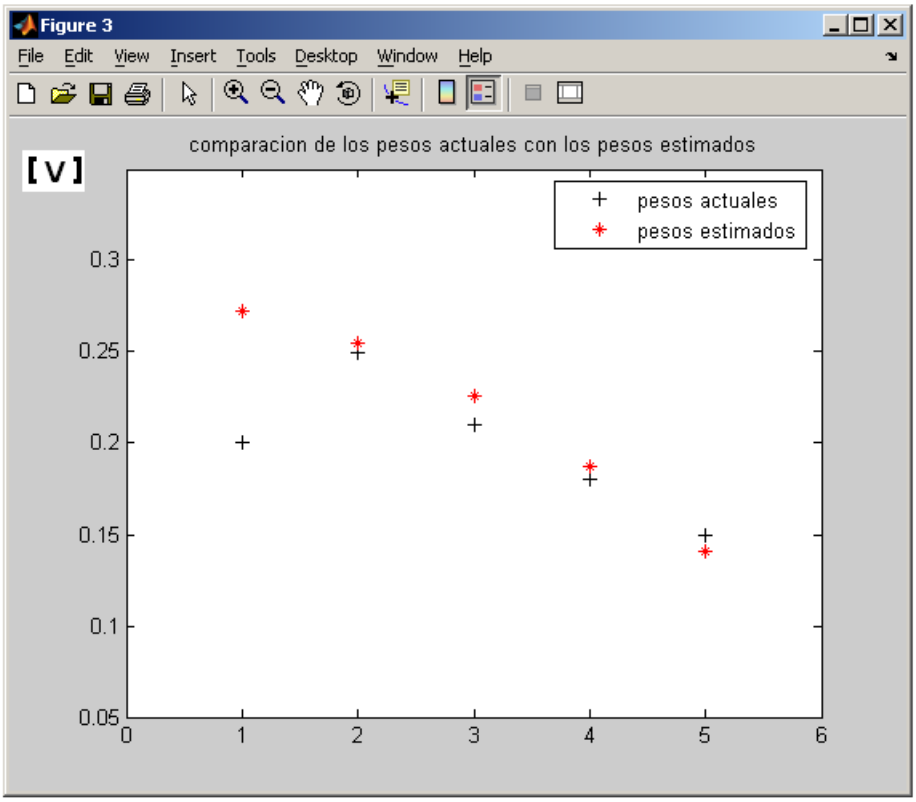
```
 ingrese el orden del filtro, cuyo coeficiente se va ha ajustar....5

orden =

     5

 ingrese el numero de antenas entre 1 y 5
 ingrese el numero de antenas entre 1,2,3,4 o 5....5
ingrese el voltaje1 (en v)..... voltaje1=0.20
 ingrese la voltaje2 (en V).... voltaje2=0.25
 ingrese la voltaje3 (en V).....voltaje3=0.21
 ingrese la voltaje4 (en v)..... voltaje4=0.18
 ingrese la voltaje5 (en v)..... voltaje5=0.15
ingrese el ruido en la antena1.....ruido1=0.01
ingrese el ruido en la antena2.....ruido2=0.01
ingrese el ruido en la antena3.....ruido3=0.01
ingrese el ruido en la antena4.....ruido4=0.01
ingrese el ruido en la antena5.....ruido5=0.01
si desea que el ruido adicional vaya a la señal a la entrada digite 1
 de lo contrario se considerara que el ruido adicional se agregara a la señal de referencia
opcion_ruido=1
ingrese el error del sistema=0.0000001
```

Pantalla 3.63 Potencias obtenidas con el programa problema1_lms_b

```
 totallength =

     400


 potencia =

     0.4943


 potencia1 =

     0.0484


 potencia2 =

     0.0718

potencia3 =

    0.0527


potencia4 =

    0.0405


potencia5 =

    0.0301
```

Pantalla 3.64 Ejemplo problema1_lms_b. Comparación de los pesos estimados

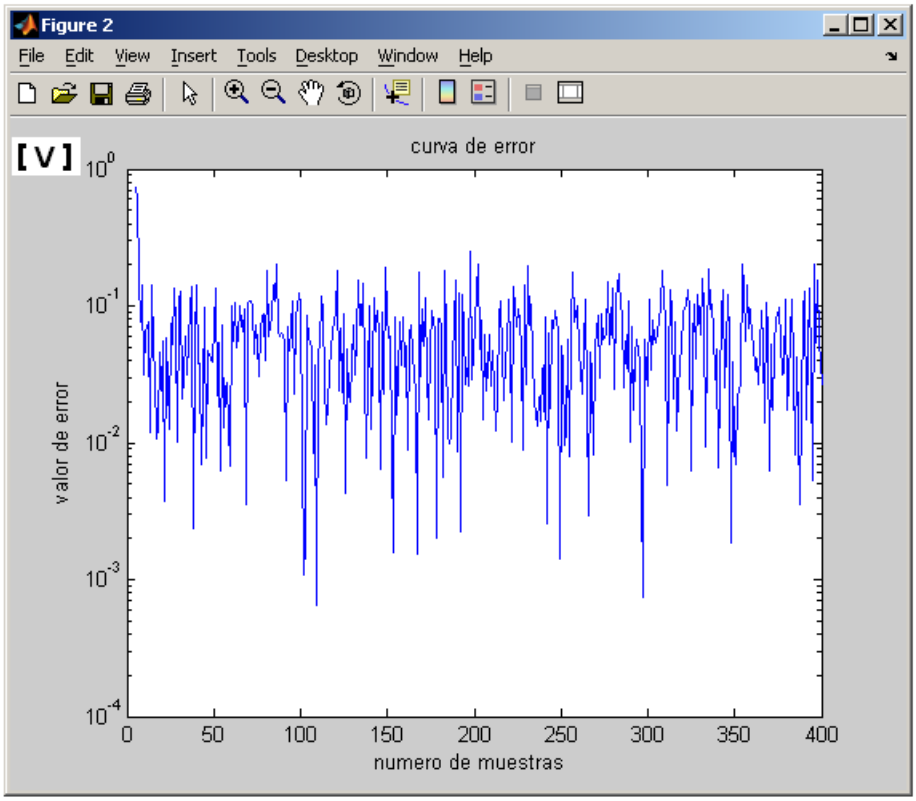
Ejemplo problema1_lms_b
Comparacion de los pesos estimados de una señal senosoidal con los pesos reales o actuales de una señal senosoidal utilizando un algoritmo lms, orden=5, numero de pesos=5.



Eje de las x constante, en el eje de las y amplitud de la señal en voltios. Amplitud_señal vs Valor constante.

Pantalla 3.65 Ejemplo problema1_lms_b. Curva de error de una señal senosoidal en un algoritmo lms

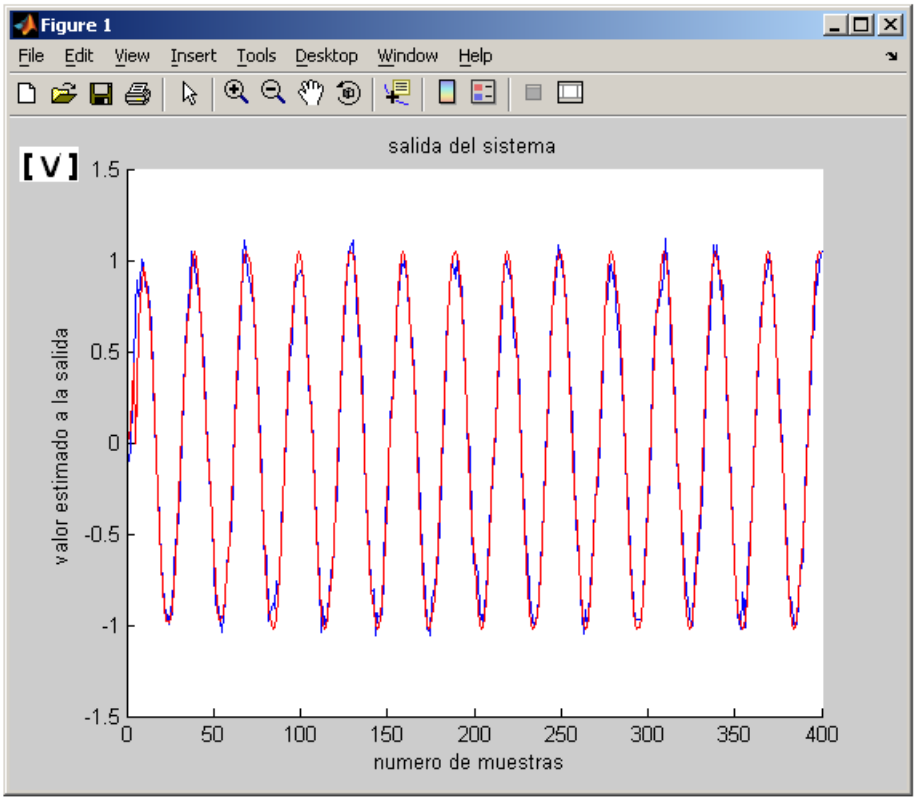
Ejemplo problema1_lms_b



Amplitud de la señal (voltios) vs numero de muestras.

Pantalla 3.66 Ejemplo problema1_lms_b. Comparación entre la salida del sistema de una señal senosoidal (lms) con la señal de referencia del sistema

Ejemplo problema1_lms_b



Amplitud de la señal (voltios) vs numero de muestras
señal de referencia (color azul)
señal obtenida por el algoritmo lms (color rojo)

Problema1_lms_c

Pantalla 3.67 Problema1_lms_c

```
* PROBLEMA1_lms.- Determinar los pesos, el numero de iteraciones y la potencia*.
* de un grupo de antenas inteligentes utilizando el algoritmo LMS (algoritmo  *.
* minimo cuadrado, las antenas estan distribuidas en forma de celdas para lo  *.
* cual requeriremos los siguientes datos:                                      *.
*                                                                              *.
*  • la señal aplicada a la entrada del filtro,   ej  Xn=2*sin((2*pi*k)/N)     *.
*  • el numero de antenas  x                                                   *.
*  • la señal de referencia, d(k)=2*cos((2*pi*k)/N);                           *.
*  • el ruido de las diferentes antenas, (hay que tener presente que el        *.
*    algoritmo LMS en su estructura internamente produce un ruido del 10       *.
*    aparte de este ruido puede haber otros.  n(f)                             *.
*  • el orden del filtro FIR cuyos coeficientes se van a ajustar, orden        *.
*  • el número de iteraciones en el aprendizaje, N (este número se hará        *.
*    hasta que cumpla el error)                                                *.
*  • la constante de ajuste   u                                                *.
*  • el vector de pesos inicial. Los voltajes de las antenas V(f)              *.
*  • el error deseado (puede ser ingresado por teclado) error                  *.
*                                                                              *.
*  A su salida devolverá:                                                      *.
*  • la salida en cada iteración durante el proceso de ajuste, y(t)            *.
*  • la curva de aprendizaje (grafico de la curva)                             *.
*  • los vectores de pesos obtenidos durante el proceso. W(t)                  *.
*  • la Potencia de la antena P(k)                                             *.
*  • el numero de iteraciones en que converge la curva y se cumple el valor    *.
*    del error.  j                                                             *.
* alumno: Juan Francisco Alvarez Alvarado                                      *.
* alumno: Maribel del Rosario Chuez Gonzales                                   *.
*                                                                              *.
* Profesor :Ing. Pedro Vargas                                                  *.
```

Pantalla 3.68 Datos del programa problema1_lms_c

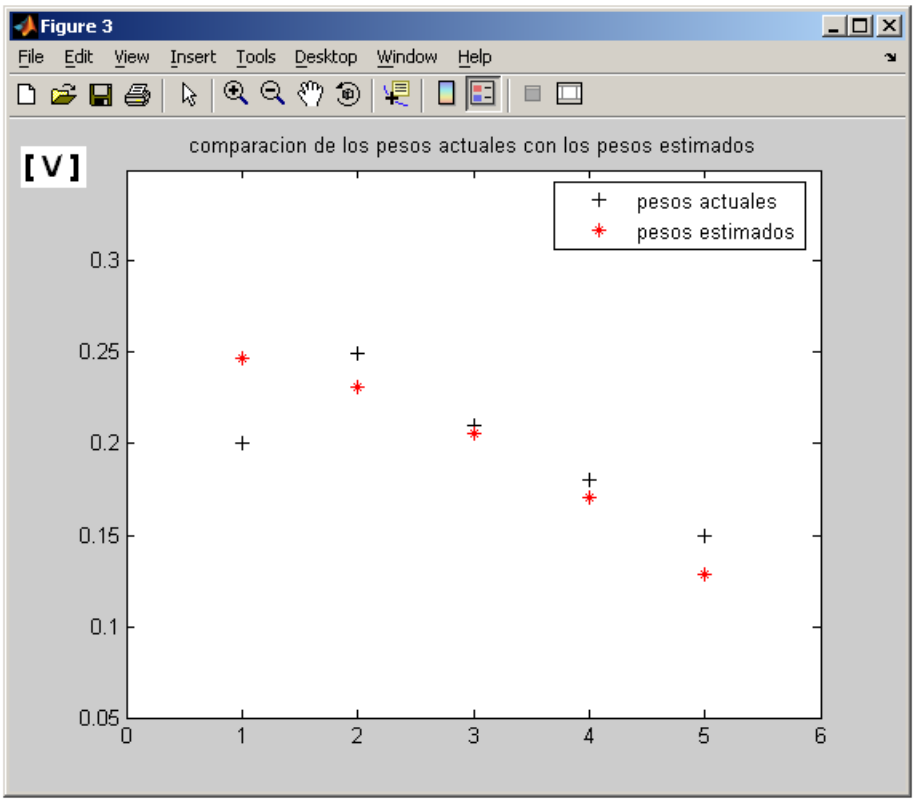
```
 ingrese el orden del filtro, cuyo coeficiente se va ha ajustar....5

orden =

     5

 ingrese el numero de antenas entre 1 y 5
 ingrese el numero de antenas entre 1,2,3,4 o 5....5
ingrese el voltaje1 (en v)..... voltaje1=0.20
 ingrese la voltaje2 (en V).... voltaje2=0.25
 ingrese la voltaje3 (en V).....voltaje3=0.21
 ingrese la voltaje4 (en v)..... voltaje4=0.18
 ingrese la voltaje5 (en v)..... voltaje5=0.15
ingrese el ruido en la antena1.....ruido1=0.01
ingrese el ruido en la antena2.....ruido2=0.01
ingrese el ruido en la antena3.....ruido3=0.01
ingrese el ruido en la antena4.....ruido4=0.01
ingrese el ruido en la antena5.....ruido5=0.01
si desea que el ruido adicional vaya a la señal a la entrada digite 1
 de lo contrario se considerara que el ruido adicional se agregara a la señal de referencia
opcion_ruido=1
ingrese el error del sistema=0.0000001
```

Pantalla 3.69 Potencias obtenidas con el programa problema1_lms_c

```
potencia =

    0.5025


potencia1 =

    0.0478


potencia2 =

    0.0710


potencia3 =

    0.0520


potencia4 =

    0.0399


potencia5 =

    0.0296
```

Pantalla 3.70 Ejemplo problema1_lms_c. Comparación de los pesos estimados

Ejemplo problema1_lms_c
Comparacion de los pesos estimados de una señal senosoidal con los pesos reales o actuales de una señal senosoidal utilizando un algoritmo lms, orden=5, numero de pesos=5.

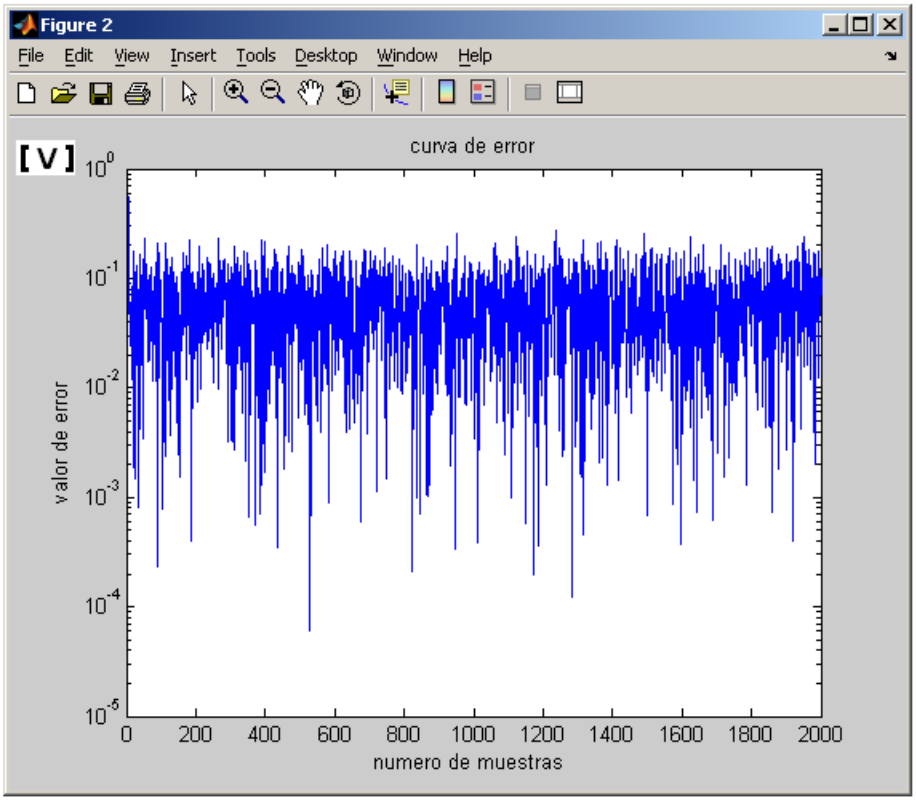


Eje de las x constante, en el eje de las y amplitud de la señal en voltios. Amplitud_señal vs Valor constante.

Parece que el error disminuyo levemente para el primer valor pero aumento levemente para los pesos 2 y 5, en fin no hay mucha diferencia en el numero de muestras. Aunque el programa tardo más tiempo en evaluar para 2000 muestras. Parece ser que el aumento de la frecuencia aumento el error en los pesos 2 y 5.

Pantalla 3.71 Ejemplo problema1_lms_c. Curva de error de una señal senosoidal en un algoritmo lms

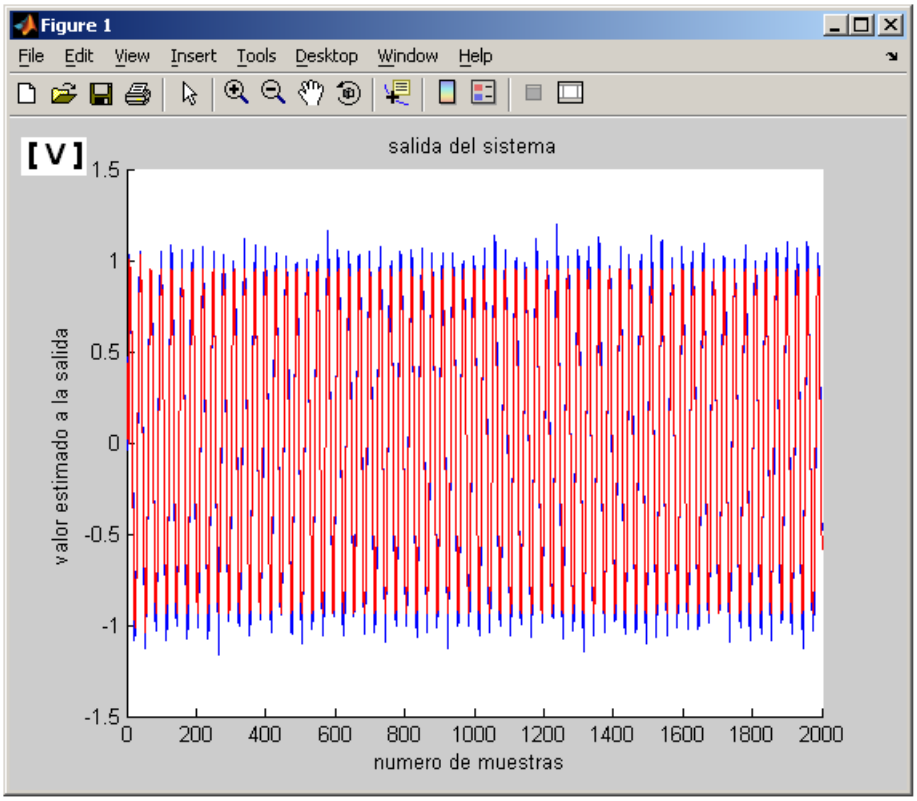
Ejemplo problema1_lms_c



Amplitud de la señal (voltios) vs numero de muestras.

Pantalla 3.72 Ejemplo problema1_lms_c. Comparación entre la salida del sistema de una señal senosoidal (lms) con la señal de referencia del sistema

Ejemplo problema1_lms_c



Amplitud de la señal (voltios) vs numero de muestras
señal de referencia (color azul)
señal obtenida por el algoritmo lms (color rojo)

Problema1_lms_d

Pantalla 3.73 Problema1_lms_d

```
* PROBLEMA1_lms.- Determinar los pesos, el numero de iteraciones y la potencia*.
* de un grupo de antenas inteligentes utilizando el algoritmo LMS (algoritmo  *.
* minimo cuadrado, las antenas estan distribuidas en forma de celdas para lo  *.
* cual requeriremos los siguientes datos:                                      *.
*                                                                              *.
*  • la señal aplicada a la entrada del filtro,   ej  Xn=2*sin((2*pi*k)/N)     *.
*  • el numero de antenas  x                                                   *.
*  • la señal de referencia, d(k)=2*cos((2*pi*k)/N);                           *.
*  • el ruido de las diferentes antenas, (hay que tener presente que el        *.
*    algoritmo LMS en su estructura internamente produce un ruido del 10       *.
*    aparte de este ruido puede haber otros.  n(f)                             *.
*  • el orden del filtro FIR cuyos coeficientes se van a ajustar, orden        *.
*  • el número de iteraciones en el aprendizaje, N (este número se hará        *.
*    hasta que cumpla el error)                                                *.
*  • la constante de ajuste   u                                                *.
*  • el vector de pesos inicial. Los voltajes de las antenas V(f)              *.
*  • el error deseado (puede ser ingresado por teclado) error                  *.
*                                                                              *.
*  A su salida devolverá:                                                      *.
*  • la salida en cada iteración durante el proceso de ajuste, y(t)            *.
*  • la curva de aprendizaje (grafico de la curva)                             *.
*  • los vectores de pesos obtenidos durante el proceso. W(t)                  *.
*  • la Potencia de la antena P(k)                                             *.
*  • el numero de iteraciones en que converge la curva y se cumple el valor    *.
*    del error.  j                                                             *.
* alumno: Juan Francisco Alvarez Alvarado                                      *.
* alumno: Maribel del Rosario Chuez Gonzales                                   *.
*                                                                              *.
* Profesor :Ing. Pedro Vargas                                                  *.
```

Pantalla 3.74 Datos del programa problema1_lms_d

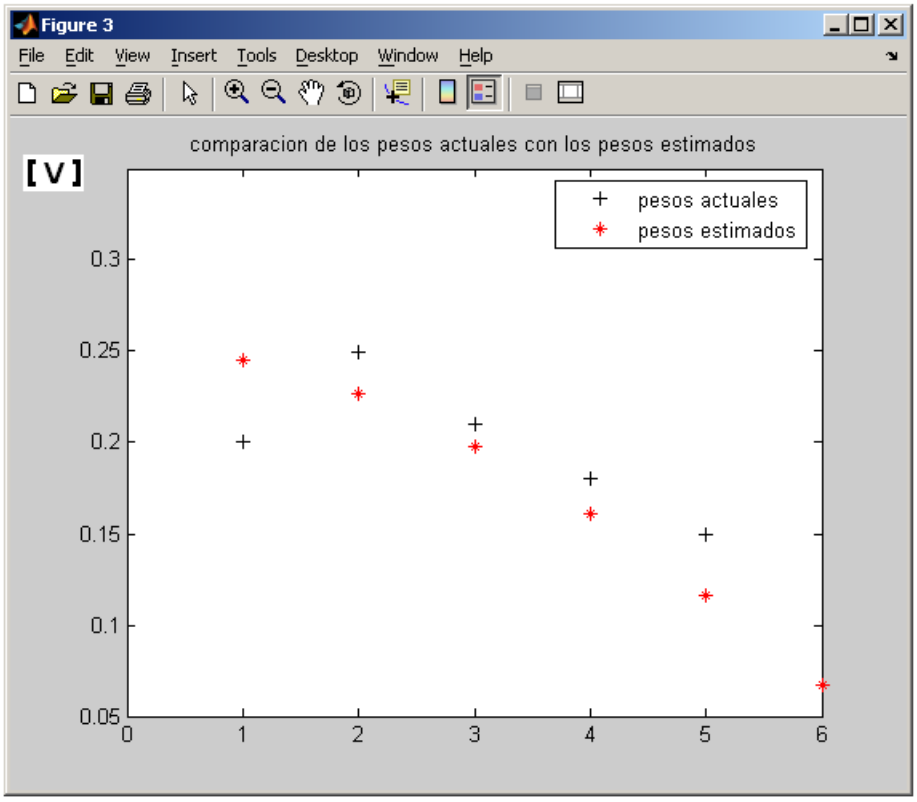
```
 ingrese el orden del filtro, cuyo coeficiente se va ha ajustar....8

orden =

     8

 ingrese el numero de antenas entre 1 y 5
 ingrese el numero de antenas entre 1,2,3,4 o 5....5
ingrese el voltaje1 (en v)..... voltaje1=0.20
 ingrese la voltaje2 (en V).... voltaje2=0.25
 ingrese la voltaje3 (en V).....voltaje3=0.21
 ingrese la voltaje4 (en v)..... voltaje4=0.18
 ingrese la voltaje5 (en v)..... voltaje5=0.15
ingrese el ruido en la antena1.....ruido1=0.01
ingrese el ruido en la antena2.....ruido2=0.01
ingrese el ruido en la antena3.....ruido3=0.01
ingrese el ruido en la antena4.....ruido4=0.01
ingrese el ruido en la antena5.....ruido5=0.01
si desea que el ruido adicional vaya a la señal a la entrada digite 1
 de lo contrario se considerara que el ruido adicional se agregara a la señal de referencia
opcion_ruido=1
ingrese el error del sistema=0.0000001
```

Pantalla 3.75 Potencias obtenidas con el programa problema1_lms_d

```
 totallength =

     400


 potencia =

     0.4943


 potencia1 =

     0.0484


 potencia2 =

     0.0718

potencia3 =

    0.0527


potencia4 =

    0.0405


potencia5 =

    0.0301
```

Pantalla 3.76 Ejemplo problema1_lms_d. Comparación de los pesos estimados

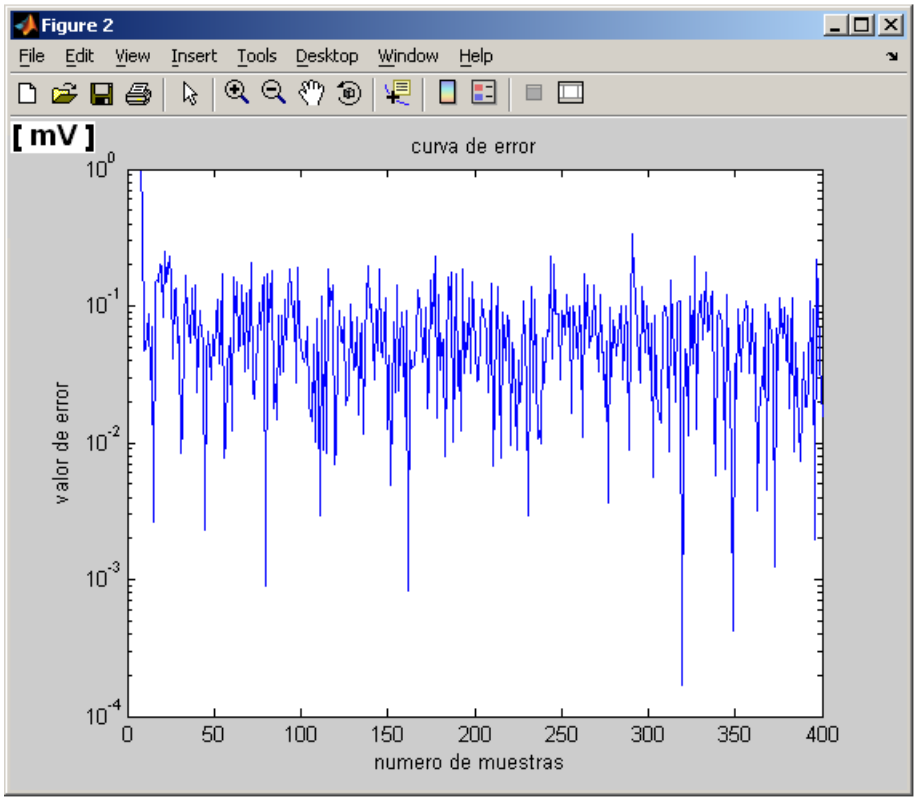
Ejemplo problema1_lms_d
Comparacion de los pesos estimados de una señal senosoidal con los pesos reales o actuales de una señal senosoidal utilizando un algoritmo lms, orden=8, numero de pesos=5.



Eje de las x constatante, en el eje de las y amplitud de la señal en voltios. Amplitud_señal vs Valor constante.

Pantalla 3.77 Ejemplo problema1_lms_d. Curva de error de una señal senosoidal en un algoritmo lms

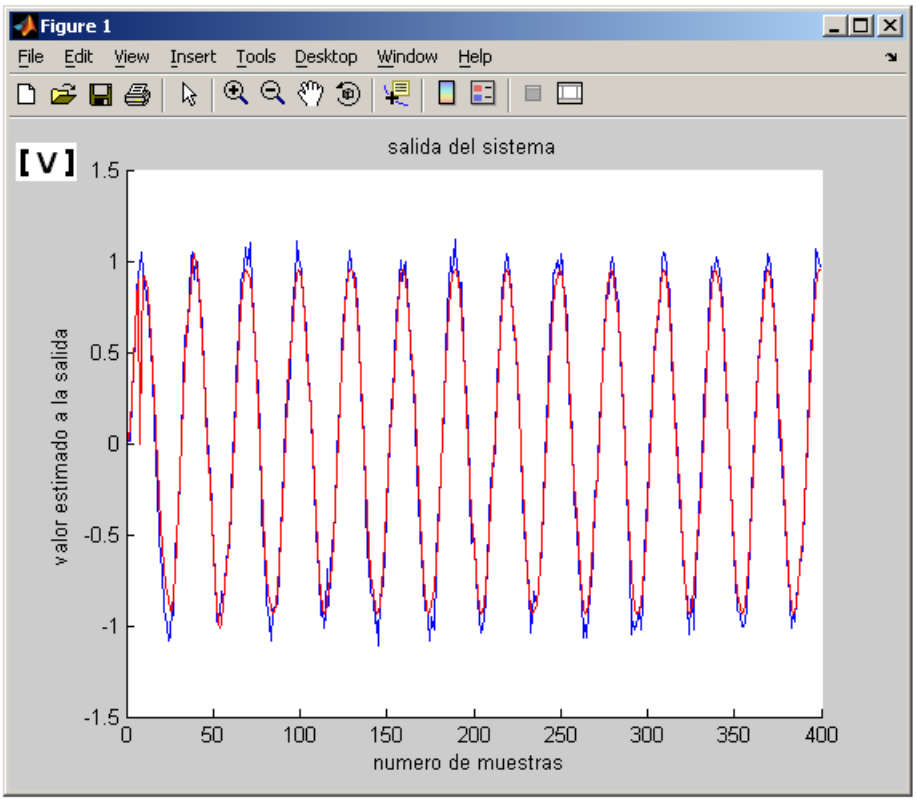
Ejemplo problema1_lms_d



Amplitud de la señal (voltios) vs numero de muestras.

Pantalla 3.78 Ejemplo problema1_lms_d. Comparación entre la salida del sistema de una señal senosoidal (lms) con la señal de referencia del sistema

Ejemplo problema1_lms_d



Amplitud de la señal (voltios) vs numero de muestras
señal de referencia (color azul)
señal obtenida por el algoritmo lms (color rojo)

Problema1_lms_e

Pantalla 3.79 Problema1_lms_e

```
* PROBLEMA1_lms.- Determinar los pesos, el numero de iteraciones y la potencia*.
* de un grupo de antenas inteligentes utilizando el algoritmo LMS (algoritmo  *.
* minimo cuadrado, las antenas estan distribuidas en forma de celdas para lo  *.
* cual requeriremos los siguientes datos:                                      *.
*                                                                              *.
*  • la señal aplicada a la entrada del filtro,   ej  Xn=2*sin((2*pi*k)/N)     *.
*  • el numero de antenas  x                                                   *.
*  • la señal de referencia, d(k)=2*cos((2*pi*k)/N);                           *.
*  • el ruido de las diferentes antenas, (hay que tener presente que el        *.
*    algoritmo LMS en su estructura internamente produce un ruido del 10       *.
*    aparte de este ruido puede haber otros.  n(f)                             *.
*  • el orden del filtro FIR cuyos coeficientes se van a ajustar, orden        *.
*  • el número de iteraciones en el aprendizaje, N (este número se hará        *.
*    hasta que cumpla el error)                                                *.
*  • la constante de ajuste   u                                               *.
*  • el vector de pesos inicial. Los voltajes de las antenas V(f)              *.
*  • el error deseado (puede ser ingresado por teclado) error                  *.
*                                                                              *.
*  A su salida devolverá:                                                      *.
*  • la salida en cada iteración durante el proceso de ajuste, y(t)            *.
*  • la curva de aprendizaje (grafico de la curva)                             *.
*  • los vectores de pesos obtenidos durante el proceso. W(t)                  *.
*  • la Potencia de la antena P(k)                                             *.
*  • el numero de iteraciones en que converge la curva y se cumple el valor    *.
*    del error.  j                                                             *.
* alumno: Juan Francisco Alvarez Alvarado                                      *.
* alumno: Maribel del Rosario Chuez Gonzales                                   *.
*                                                                              *.
* Profesor :Ing. Pedro Vargas                                                  *.
```

Pantalla 3.80 Datos del programa problema1_lms_e

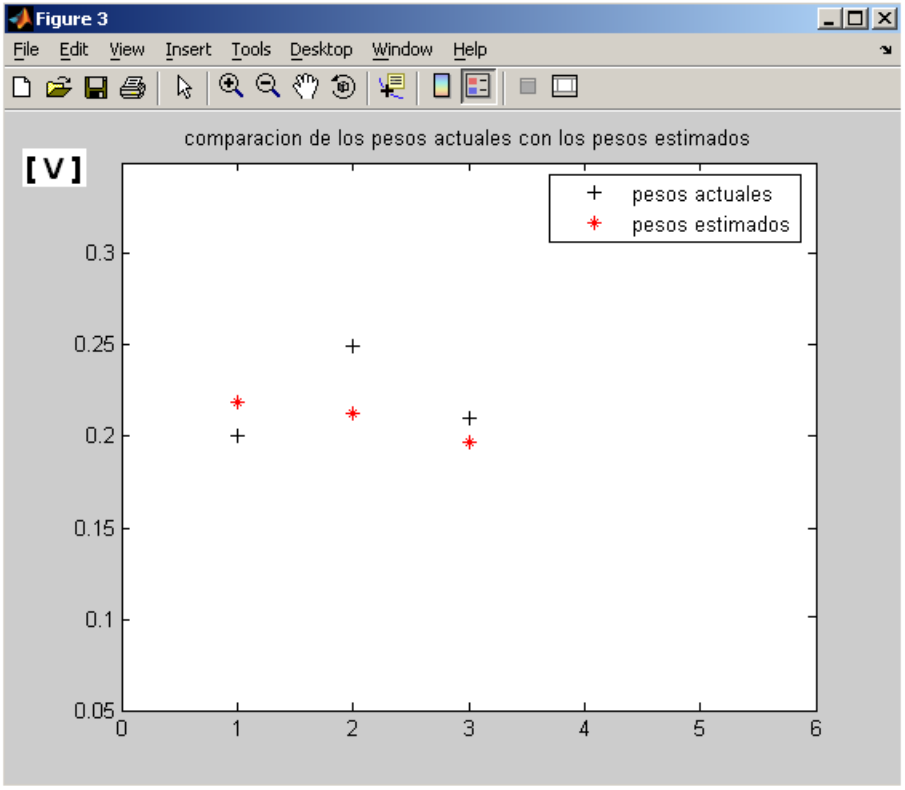
```
 ingrese el orden del filtro, cuyo coeficiente se va ha ajustar....10

orden =

    10

 ingrese el numero de antenas entre 1 y 5
 ingrese el numero de antenas entre 1,2,3,4 o 5....3
ingrese el voltaje1 (en v)..... voltaje1=0.20
 ingrese la voltaje2 (en V).... voltaje2=0.25
 ingrese la voltaje3 (en V).....voltaje3=0.21
ingrese el ruido en la antena1.....ruido1=0.01
ingrese el ruido en la antena2.....ruido2=0.01
ingrese el ruido en la antena3.....ruido3=0.01
si desea que el ruido adicional vaya a la señal a la entrada digite 1
 de lo contrario se considerara que el ruido adicional se agregara a la señal de referencia
opcion_ruido=1
ingrese el error del sistema=0.0000001
```

Pantalla 3.81 Potencias obtenidas con el programa problema1_lms_e

```
totallength =

   400


potencia =

    0.5047


potencia1 =

    0.0468


potencia1 =

    0.0697


potencia3 =

    0.0510
```

Pantalla 3.82 Ejemplo problema1_lms_e. Comparación de los pesos estimados

Ejemplo problema1_lms_e
Comparacion de los pesos estimados de una señal senosoidal con los pesos reales o actuales de una señal senosoidal utilizando un algoritmo lms, orden=10, numero de pesos=3.

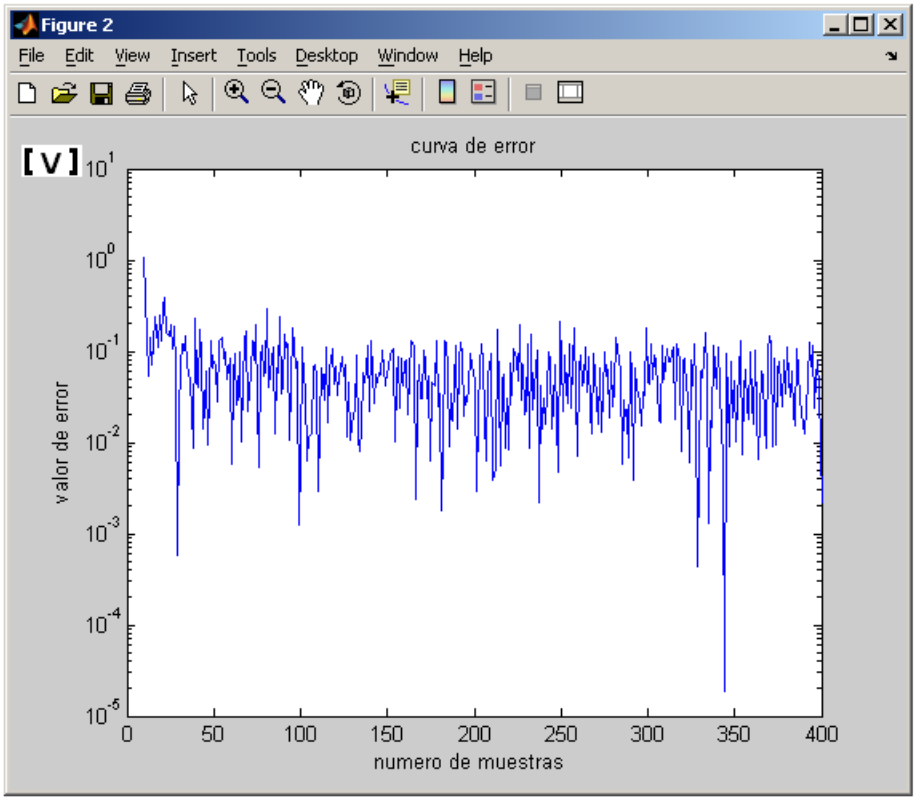


Eje de las x constante, en el eje de las y amplitud de la señal en voltios. Amplitud_señal vs Valor constante.

La diferencia entre el valor El error parece ser casi igual no hay un gran aumento del error, con orden=10. Después será conveniente ver qué sucede con otra curva en la misma circunstancia.

Pantalla 3.83 Ejemplo problema1_lms_e. Curva de error de una señal senosoidal en un algoritmo lms

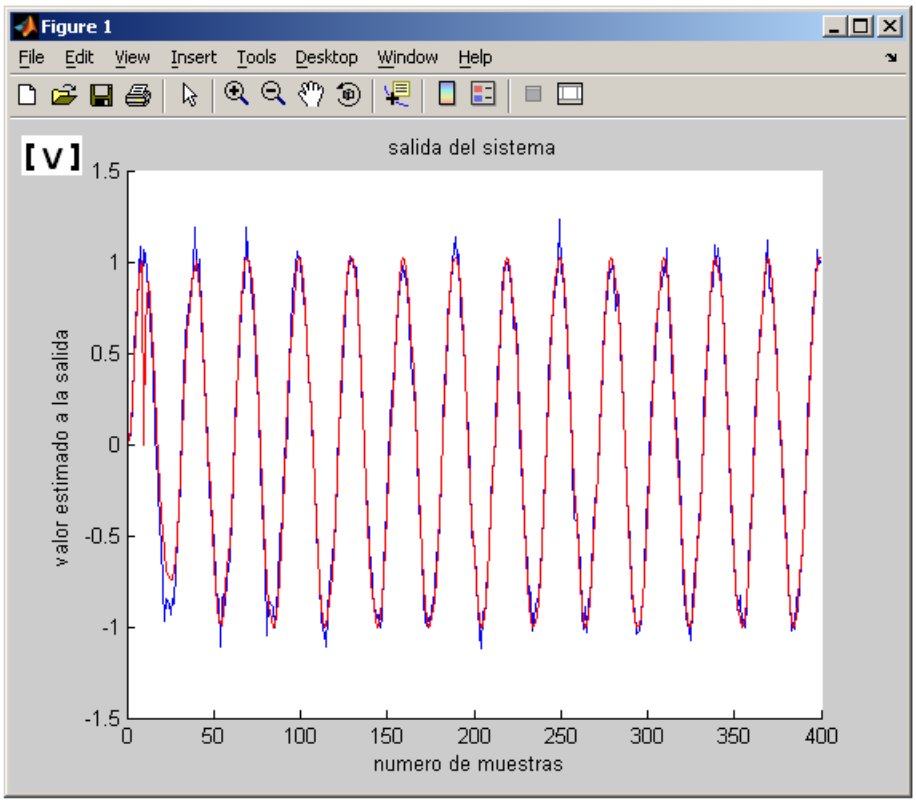
Ejemplo problema1_lms_e



Amplitud de la señal (voltios) vs numero de muestras

Pantalla 3.84 Ejemplo problema1_lms_e. Comparación entre la salida del sistema de una señal senosoidal (lms) con la señal de referencia del sistema

Ejemplo problema1_lms_e
Comparacion entre la salida del sistema de una señal senosoidal (lms) con la señal de referencia del sistema



Amplitud de la señal (voltios) vs numero de muestras
señal de referencia (color azul)
señal obtenida por el algoritmo lms (color rojo)

Problema1_lms_f

Pantalla 3.85 Problema1_lms_f

```
* PROBLEMA1_lms.- Determinar los pesos, el numero de iteraciones y la potencia*.
* de un grupo de antenas inteligentes utilizando el algoritmo LMS (algoritmo  *.
* minimo cuadrado, las antenas estan distribuidas en forma de celdas para lo  *.
* cual requeriremos los siguientes datos:                                      *.
*                                                                              *.
*  • la señal aplicada a la entrada del filtro,   ej  Xn=2*sin((2*pi*k)/N)     *.
*  • el numero de antenas  x                                                   *.
*  • la señal de referencia, d(k)=2*cos((2*pi*k)/N);                           *.
*  • el ruido de las diferentes antenas, (hay que tener presente que el        *.
*    algoritmo LMS en su estructura internamente produce un ruido del 10       *.
*    aparte de este ruido puede haber otros.  n(f)                             *.
*  • el orden del filtro FIR cuyos coeficientes se van a ajustar, orden        *.
*  • el número de iteraciones en el aprendizaje, N (este número se hará        *.
*    hasta que cumpla el error)                                                *.
*  • la constante de ajuste   u                                               *.
*  • el vector de pesos inicial. Los voltajes de las antenas V(f)              *.
*  • el error deseado (puede ser ingresado por teclado) error                  *.
*                                                                              *.
*  A su salida devolverá:                                                      *.
*  • la salida en cada iteración durante el proceso de ajuste, y(t)            *.
*  • la curva de aprendizaje (grafico de la curva)                             *.
*  • los vectores de pesos obtenidos durante el proceso. W(t)                  *.
*  • la Potencia de la antena P(k)                                             *.
*  • el numero de iteraciones en que converge la curva y se cumple el valor    *.
*    del error.  j                                                             *.
* alumno: Juan Francisco Alvarez Alvarado                                      *.
* alumno: Maribel del Rosario Chuez Gonzales                                   *.
*                                                                              *.
* Profesor :Ing. Pedro Vargas                                                  *.
```

Pantalla 3.86 Datos del programa problema1_lms_f

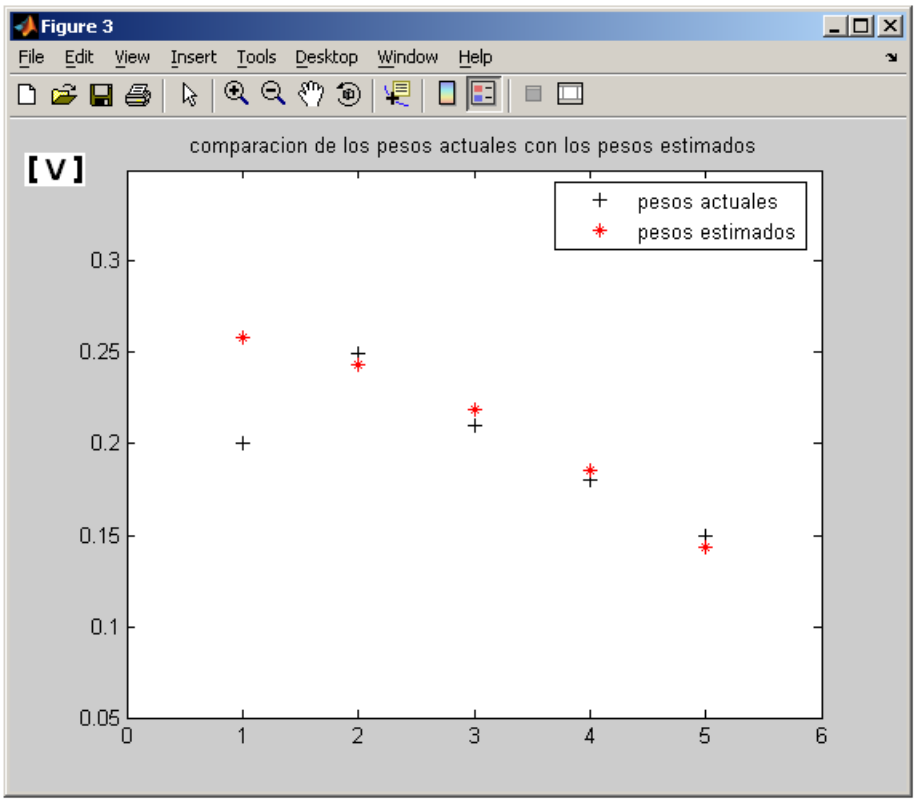
```
 ingrese el orden del filtro, cuyo coeficiente se va ha ajustar....5

orden =

     5

 ingrese el numero de antenas entre 1 y 5
 ingrese el numero de antenas entre 1,2,3,4 o 5....5
ingrese el voltaje1 (en v)..... voltaje1=0.20
 ingrese la voltaje2 (en V).... voltaje2=0.25
 ingrese la voltaje3 (en V).....voltaje3=0.21
 ingrese la voltaje4 (en v)..... voltaje4=0.18
 ingrese la voltaje5 (en v)..... voltaje5=0.15
ingrese el ruido en la antena1.....ruido1=0.01
ingrese el ruido en la antena2.....ruido2=0.02
ingrese el ruido en la antena3.....ruido3=0.02
ingrese el ruido en la antena4.....ruido4=0.03
ingrese el ruido en la antena5.....ruido5=0.04
si desea que el ruido adicional vaya a la señal a la entrada digite 1
 de lo contrario se considerara que el ruido adicional se agregara a la señal de referencia
opcion_ruido=1
ingrese el error del sistema=0.0000001
```

Pantalla 3.87 Potencias obtenidas con el programa problema1_lms_f

```
potencia =

    0.5033


potencia1 =

    0.0478


potencia2 =

    0.0762


potencia3 =

    0.0564


potencia4 =

    0.0478


potencia5 =

    0.0399
```

Pantalla 3.88 Ejemplo problema1_lms_f. Comparación de los pesos estimados

Ejemplo problema1_lms_f
Comparacion de los pesos estimados de una señal senosoidal con los pesos reales o actuales de una señal senosoidal utilizando un algoritmo lms, orden=5, numero de pesos=5.

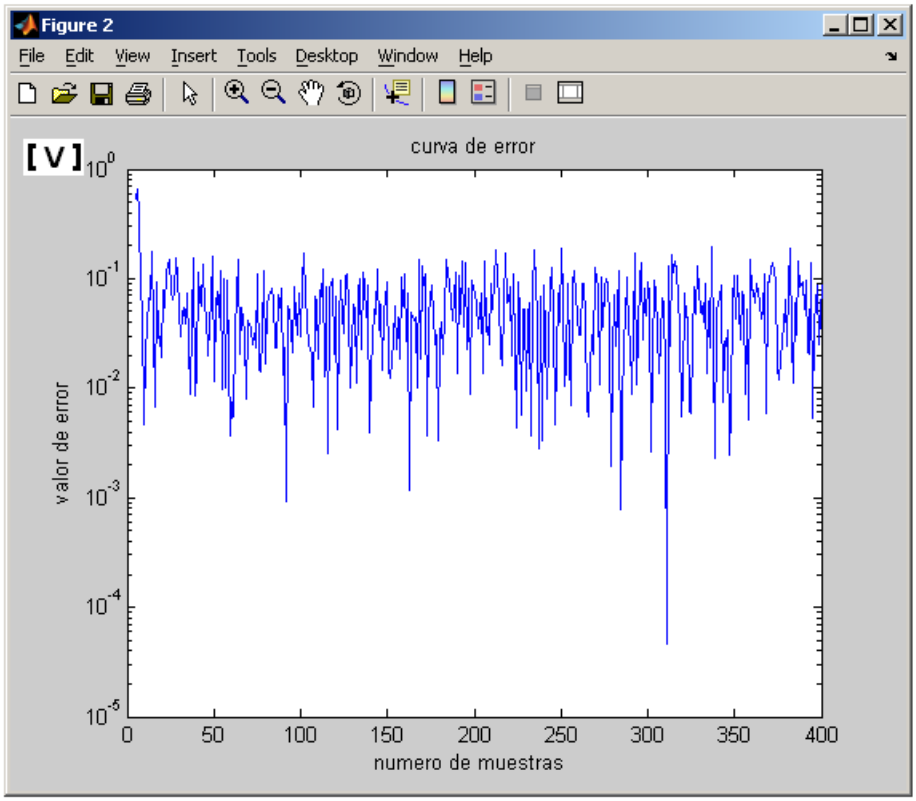


Eje de las x constante, en el eje de las y amplitud de la señal en voltios. Amplitud_señal vs Valor constante.

Se ve en el grafico que con valores variables de error 0.01, 0.02, 0.03 y 0.04 no hay mucha diferencias con el error. Para esta curva.

Pantalla 3.89 Ejemplo problema1_lms_f. Curva de error de una señal senosoidal en un algoritmo lms

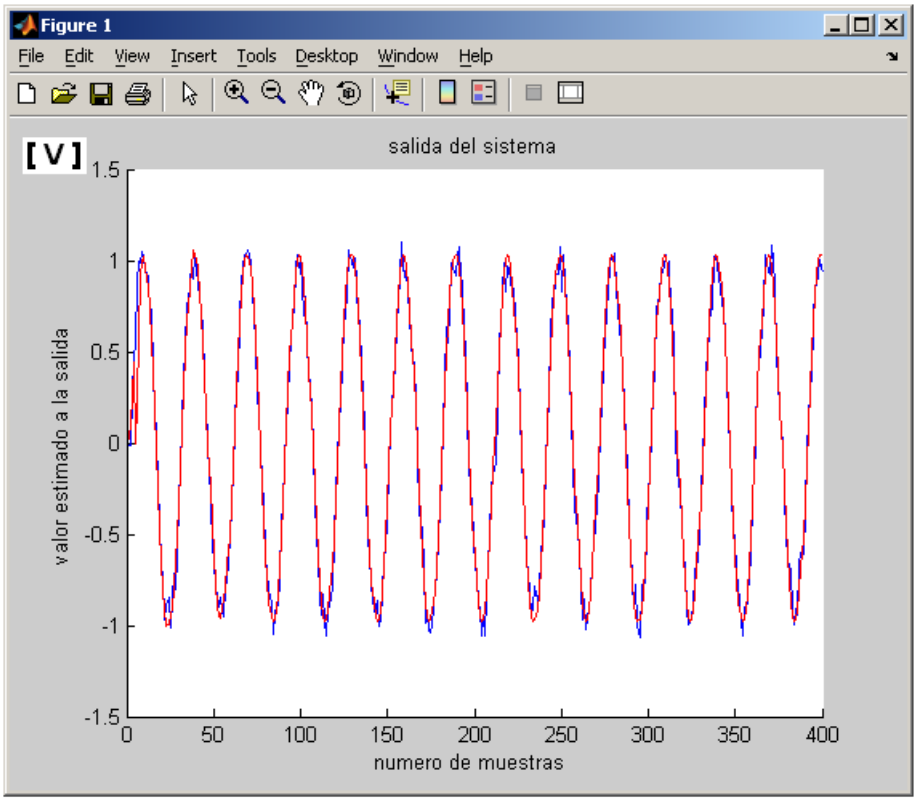
Ejemplo problema1_lms_f



Amplitud de la señal (voltios) vs numero de muestras

Pantalla 3.90 Ejemplo problema1_lms_f. Comparación entre la salida del sistema de una señal senosoidal (lms) con la señal de referencia del sistema

Ejemplo problema1_lms_f



Amplitud de la señal (voltios) vs numero de muestras
señal de referencia (color azul)
señal obtenida por el algoritmo lms (color rojo)

Problema1_rls_a

Pantalla 4.1 Problema1_rls_a

```
* PROBLEMA1_rls.- Determinar los pesos, el numero de iteraciones y la potencia*.
* de un grupo de antenas inteligentes utilizando el algoritmo RLS (algoritmo  *.
* minimo cuadrado recursivo, las antenas estan distribuidas en forma de celdas*.
* para lo cual requeriremos de los siguientes datos:                           *.
*                                                                              *.
*  • la señal aplicada a la entrada del filtro,   ej  Xn=sin((2*pi*k)/N)    *.
*  • el numero de antenas  x                                                   *.
*  • la señal de referencia, d(k)=cos((2*pi*k)/N);                          *.
*  • el ruido de las diferentes antenas, (hay que tener presente que el       *.
*    algoritmo LMS en su estructura internamente produce un ruido del 10      *.
*    aparte de este ruido puede haber otros.  n(f)                            *.
*  • el orden del filtro FIR cuyos coeficientes se van a ajustar, orden       *.
*  • el número de iteraciones en el aprendizaje, N (este número se hará       *.
*    hasta que cumpla el error)                                               *.
*  • la constante de ajuste   u                                               *.
*  • el vector de pesos inicial. Los voltajes de las antenas V(f)             *.
*  • el error deseado (puede ser ingresado por teclado) error                 *.
*                                                                              *.
*  A su salida devolverá:                                                      *.
*  • la salida en cada iteración durante el proceso de ajuste, y(t)           *.
*  • la curva de aprendizaje (grafico de la curva)                            *.
*  • los vectores de pesos obtenidos durante el proceso. W(t)                 *.
*  • la Potencia de la antena P(k)                                            *.
*  • el numero de iteraciones en que converge la curva y se cumple el valor   *.
*    del error.  j                                                            *.
* alumno: Juan Francisco Alvarez Alvarado                                      *.
* alumno: Maribel del Rosario Chuez Gonzales                                   *.
*                                                                              *.
* Profesor :Ing. Pedro Vargas                                                  *.
*                                                                              *.
********************************************************************************.
```

Pantalla 4.2 Datos del programa problema1_rls_a

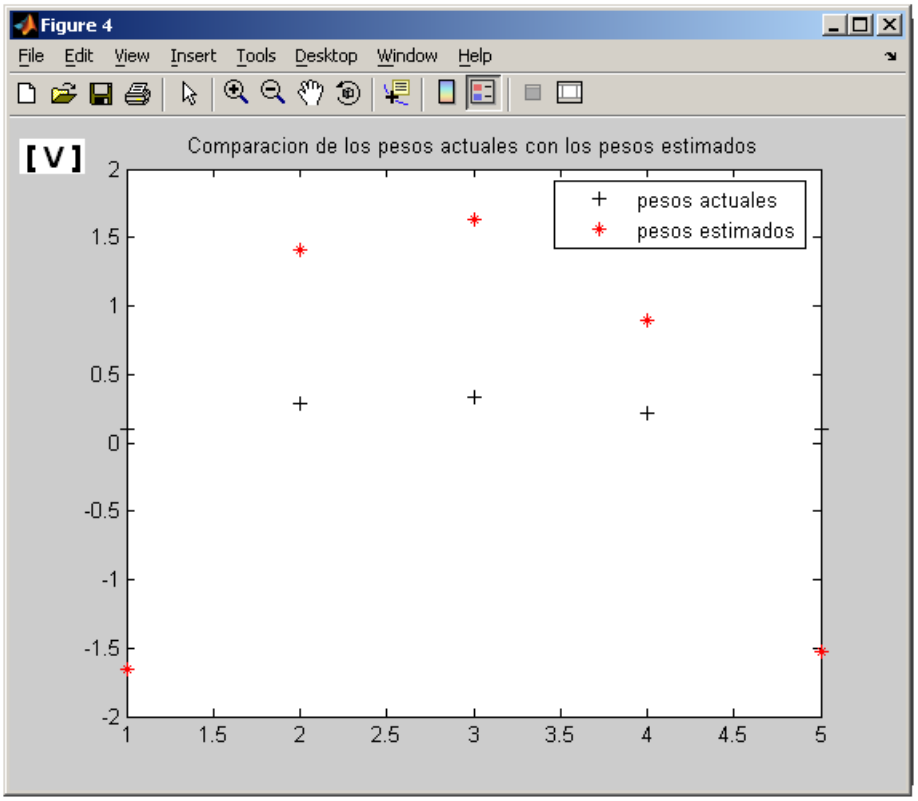
```
 ingrese el orden del filtro, cuyo coeficiente se va ha ajustar....5

orden =

     5

 ingrese el numero de antenas entre 1 y 15
 ingrese el numero de antenas entre 1,2,3,4,5,6,7,8,9,10,11,12,13,14 o 15....5
ingrese el voltaje1 (en v)..... voltaje1=0.097631
 ingrese la voltaje2 (en V).... voltaje2=0.287310
 ingrese la voltaje3 (en V).....voltaje3=0.335965
 ingrese la voltaje4 (en v)..... voltaje4=0.2209
 ingrese la voltaje5 (en v)..... voltaje5=0.0963
ingrese el ruido en la antena1.....ruido1=0.01
ingrese el ruido en la antena2.....ruido2=0.01
ingrese el ruido en la antena3.....ruido3=0.01
ingrese el ruido en la antena4.....ruido4=0.01
ingrese el ruido en la antena5.....ruido5=0.02
si desea que el ruido adicional vaya a la señal a la entrada digite 1
 de lo contrario se considerara que el ruido adicional se agregara a la señal de referencia
opcion_ruido=1
ingrese el error del sistema=0.0000000000001

h =

    0.0976
    0.2873
    0.3360
    0.2209
    0.0963
```

Pantalla 4.3 Potencias obtenidas con el programa problema1_rls_a

```
potencia =

    0.5006

potencia1 =

    0.0166

potencia2 =

    0.0934

potencia3 =

    0.1247

potencia4 =

    0.0583

potencia5 =

    0.0185
```

Pantalla 4.4 Ejemplo problema1_rls_a. Comparación de los pesos estimados

Ejemplo problema1_rls_a
comparacion de los pesos estimados de una señal senosoidal con los pesos reales o actuales de una señal senosoidal utilizando un algoritmo rls, orden=5, numero pesos=5



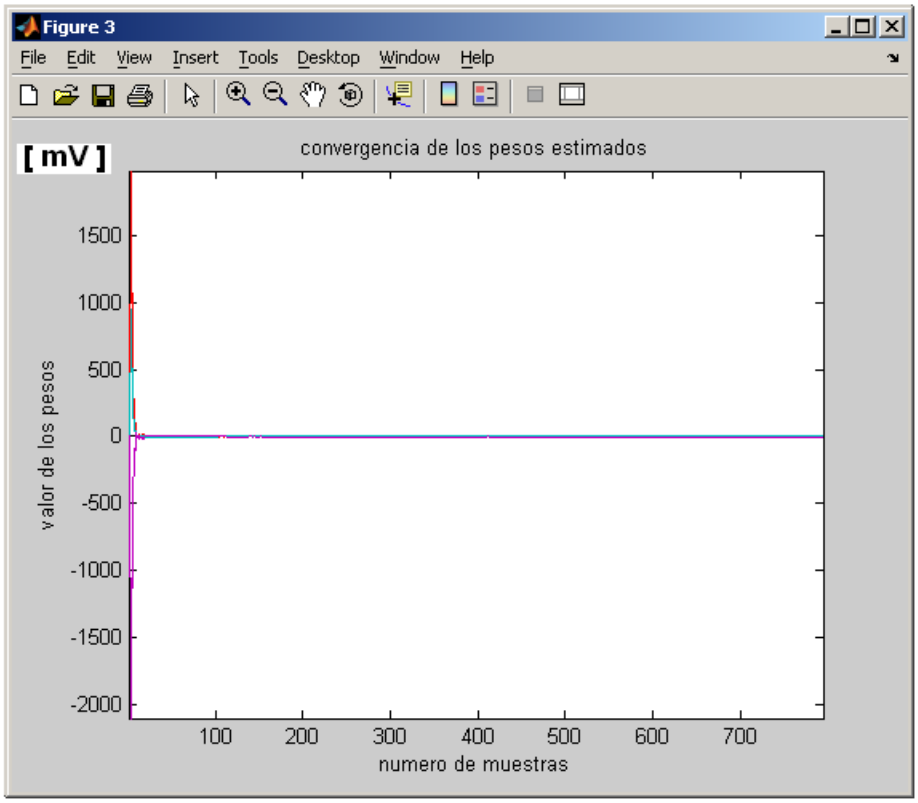
Eje de las x constante, eje de las y (amplitud de la señal en voltios). Amplitud_señal vs Valor constante

En este primer ejemplo utilizamos valores que se suelen utilizar en las funciones gaussianas, por eso se ha presentado un error muy alto al aplicarlo en una función senosoidal, pero aun así se ve que trato de converger aunque sea con un error muy alto. También se forma un nuevo grafico relación de la convergencia de los pesos estimados con los valores de los pesos, este grafico no existe para el algoritmo lms, y es otra de las

características que tiene de diferencia el algoritmo rls con el lms. Otra característica que tiene el algoritmo rls es que el numero de muestras de los pesos estimados puede ser distinto que el numero de muestras para el cálculo del error. Son independientes una de otra.

Pantalla 4.5 Ejemplo problema1_rls_a. Covergencia de los pesos estimados

Ejemplo problema1_rls_a
Grafico de la convergencia de los pesos estimados en una señal senosoidal utilizando un algoritmo rls, orden=5, numero de pesos=5

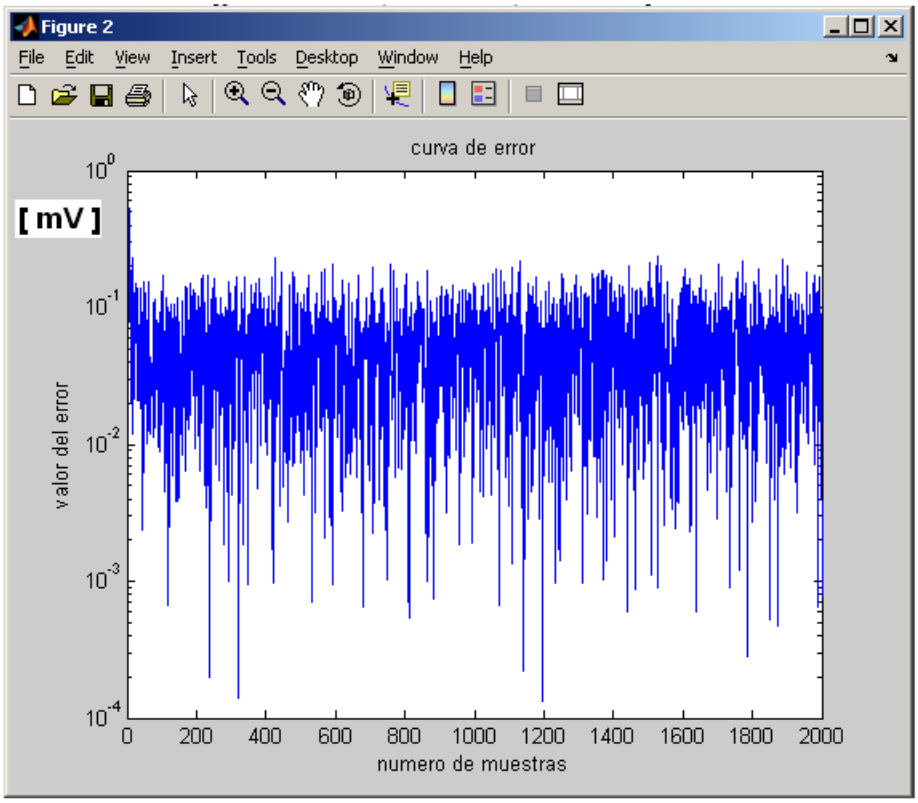


Valor de los pesos vs numero de muestras

Aquí vemos que con pocas muestras el sistema ya supone que ha alcanzado la convergencia, aunque en la realidad hay un error más alto del esperado, para reducir este error seria recomendable normalizar la ecuación y usar mejores valores para el filtro. En la mayoría de las simulaciones la curva se forma dentro de los 50 primeros valores de muestra.

Pantalla 4.6 Ejemplo problema1_rls_a. Curva de error de una señal senosoidal en un algoritmo rls

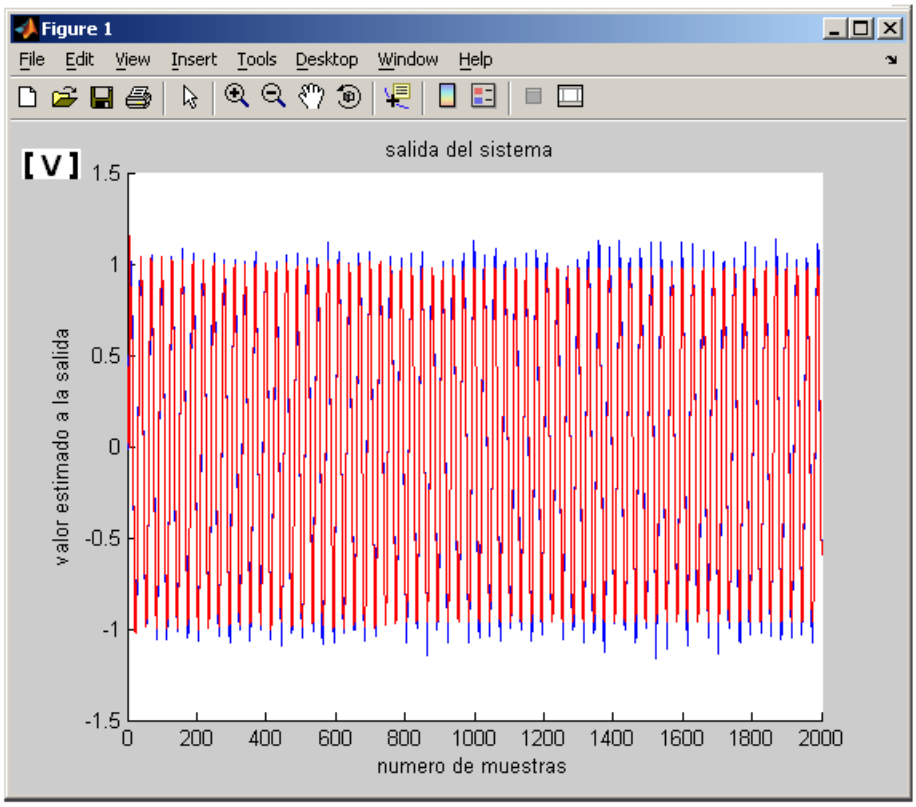
Ejemplo problema1_rls_a



Amplitud de la señal (voltios) vs numero de muestras

Pantalla 4.7 Ejemplo problema1_rls_a. Comparación entre la salida del sistema de una señal senosoidal (rls) con la señal de referencia del sistema

Ejemplo problema1_rls_a



Amplitud de la señal (voltios) vs numero de muestras
señal de referencia (color azul)
señal obtenida por el algoritmo rls (color rojo)

Para tratar de disminuir el error se intento dividir el peso que se obtenía en el programa para 1.5 siendo h+, los pesos originales y w* de los pesos obtenidos por el programa, al dividir estos pesos obtenidos para 10 se obtuvo una disminución significativa del error,

aunque el error aun sigue siendo cercano al 10 por ciento en 4 de los cinco pesos, en uno de ellos fue mucho mayor

Problema1_rls_b

Pantalla 4.8 Problema1_rls_b

```
* PROBLEMA1_rls.- Determinar los pesos, el numero de iteraciones y la potencia*.
* de un grupo de antenas inteligentes utilizando el algoritmo RLS (algoritmo  *.
* minimo cuadrado recursivo, las antenas estan distribuidas en forma de celdas*.
* para lo cual requeriremos de los siguientes datos:                          *.
*                                                                             *.
*  • la señal aplicada a la entrada del filtro,   ej  Xn=sin((2*pi*k)/N)    *.
*  • el numero de antenas  x                                                  *.
*  • la señal de referencia, d(k)=cos((2*pi*k)/N);                          *.
*  • el ruido de las diferentes antenas, (hay que tener presente que el       *.
*    algoritmo LMS en su estructura internamente produce un ruido del 10      *.
*    aparte de este ruido puede haber otros.  n(f)                            *.
*  • el orden del filtro FIR cuyos coeficientes se van a ajustar, orden       *.
*  • el número de iteraciones en el aprendizaje, N (este número se hará       *.
*    hasta que cumpla el error)                                               *.
*  • la constante de ajuste   u                                               *.
*  • el vector de pesos inicial. Los voltajes de las antenas V(f)             *.
*  • el error deseado (puede ser ingresado por teclado) error                 *.
*                                                                             *.
*  A su salida devolverá:                                                     *.
*  • la salida en cada iteración durante el proceso de ajuste, y(t)           *.
*  • la curva de aprendizaje (grafico de la curva)                            *.
*  • los vectores de pesos obtenidos durante el proceso. W(t)                 *.
*  • la Potencia de la antena P(k)                                            *.
*  • el numero de iteraciones en que converge la curva y se cumple el valor   *.
*    del error.  j                                                            *.
* alumno: Juan Francisco Alvarez Alvarado                                     *.
* alumno: Maribel del Rosario Chuez Gonzales                                  *.
*                                                                             *.
* Profesor :Ing. Pedro Vargas                                                 *.
*                                                                             *.
*******************************************************************************.
```

Pantalla 4.9 Datos del programa problema1_rls_b

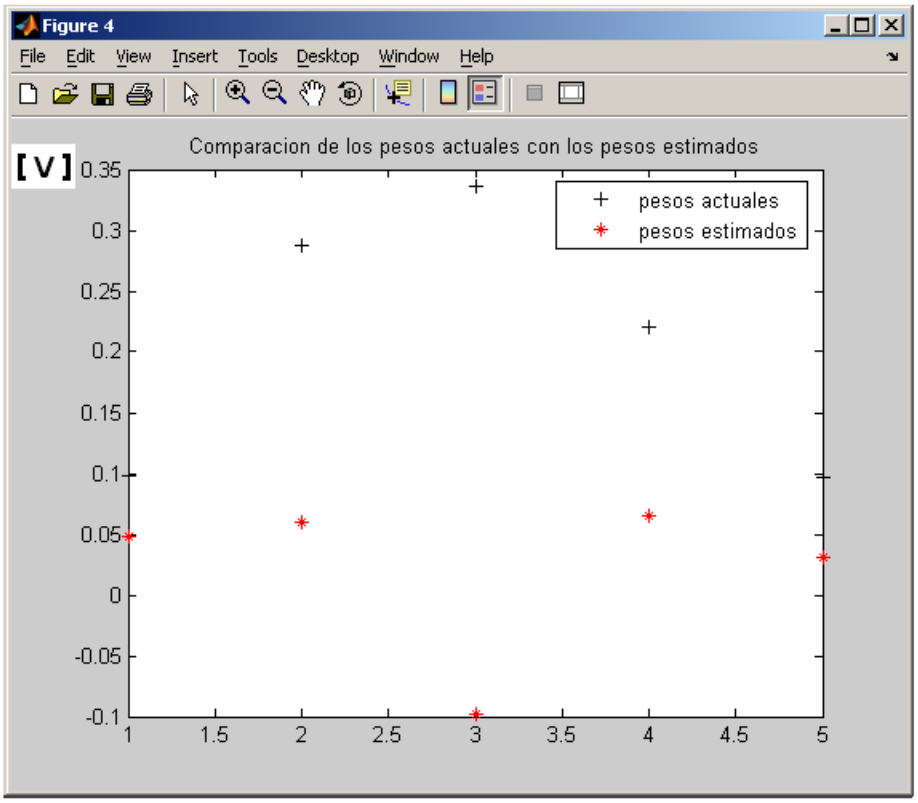
```
 ingrese el orden del filtro, cuyo coeficiente se va ha ajustar....5

orden =

     5

 ingrese el numero de antenas entre 1 y 15
 ingrese el numero de antenas entre 1,2,3,4,5,6,7,8,9,10,11,12,13,14 o 15....5
ingrese el voltaje1 (en v)..... voltaje1=0.097631
 ingrese la voltaje2 (en V).... voltaje2=0.287310
 ingrese la voltaje3 (en V).....voltaje3=0.335965
 ingrese la voltaje4 (en v)..... voltaje4=0.2209
 ingrese la voltaje5 (en v)..... voltaje5=0.0963
ingrese el ruido en la antena1.....ruido1=0.01
ingrese el ruido en la antena2.....ruido2=0.01
ingrese el ruido en la antena3.....ruido3=0.01
ingrese el ruido en la antena4.....ruido4=0.01
ingrese el ruido en la antena5.....ruido5=0.02
si desea que el ruido adicional vaya a la señal a la entrada digite 1
 de lo contrario se considerara que el ruido adicional se agregara a la señal de referencia
opcion_ruido=1
ingrese el error del sistema=0.0000000000001

h =

    0.0976
    0.2873
    0.3360
    0.2209
    0.0963
```

Pantalla 4.10 Potencias obtenidas con el programa problema1_rls_b

```
potencia =

    0.5006

potencia1 =

    0.0166

potencia2 =

    0.0934

potencia3 =

    0.1247

potencia4 =

    0.0583

potencia5 =

    0.0185
```

Pantalla 4.11 Ejemplo problema1_rls_b. Comparación de los pesos estimados

Ejemplo problema1_rls_b
comparacion de los pesos estimados de una señal senosoidal con los pesos reales o actuales de una señal senosoidal utilizando un algoritmo rls, orden=5, numero pesos=5

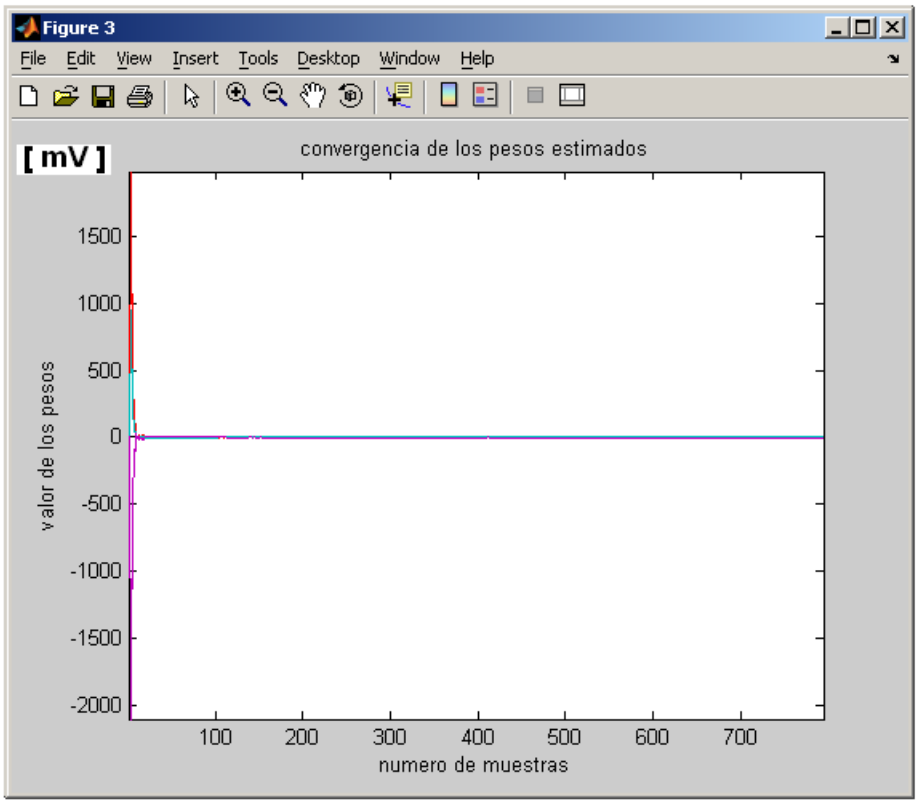


Eje de las x constante, eje de las y (amplitud de la señal en voltios). Amplitud_señal vs Valor constante

Pantalla 4.12 Ejemplo problema1_rls_b. Covergencia de los pesos estimados

Ejemplo problema1_rls_b

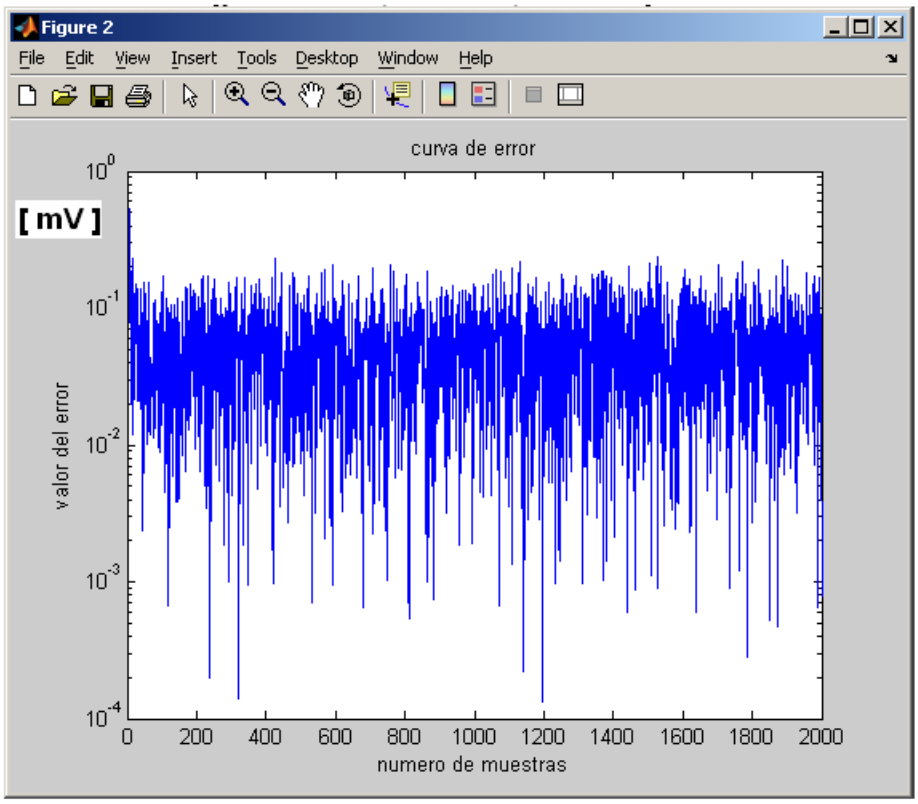
Grafico de la convergencia de los pesos estimados en una señal senosoidal utilizando un algoritmo rls, orden=5, numero de pesos= 5



Valor de los pesos vs numero de muestras

Pantalla 4.13 Ejemplo problema1_rls_b. Curva de error de una señal senosoidal en un algoritmo rls

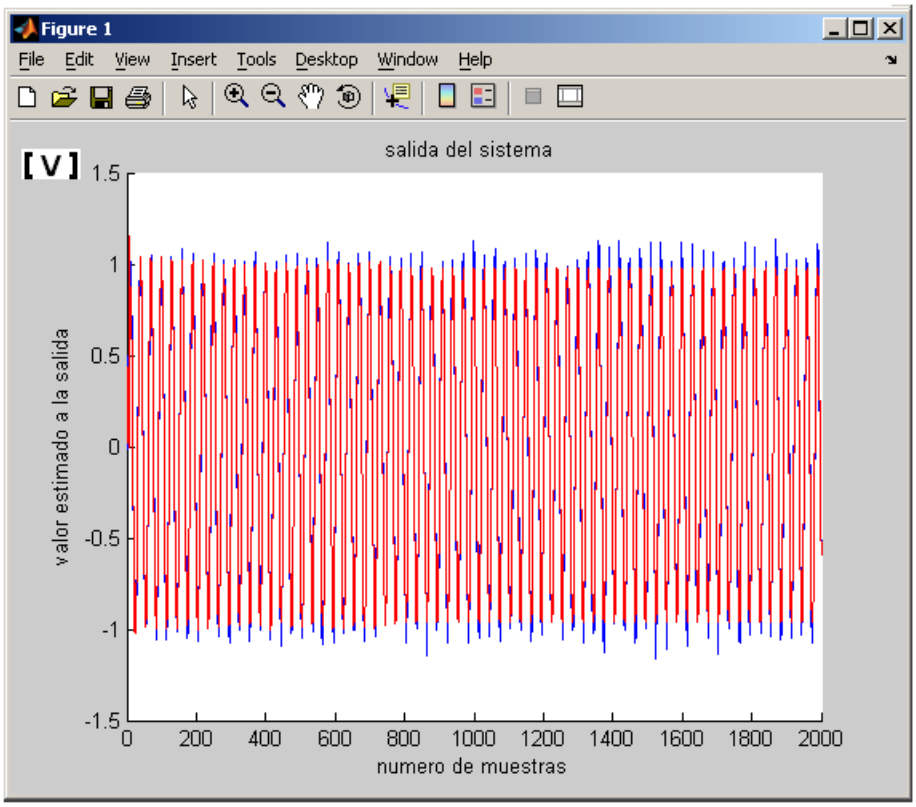
Ejemplo problema1_rls_b



Amplitud de la señal (voltios) vs numero de muestras

Pantalla 4.14 Ejemplo problema1_rls_b. Comparación entre la salida del sistema de una señal senosoidal (rls) con la señal de referencia del sistema

Ejemplo problema1_rls_b



Amplitud de la señal (voltios) vs numero de muestras
señal de referencia (color azul)
señal obtenida por el algoritmo rls (color rojo)

Problema1_rls_c

Pantalla 4.15 Problema1_rls_c

```
* PROBLEMA1_rls.- Determinar los pesos, el numero de iteraciones y la potencia*.
* de un grupo de antenas inteligentes utilizando el algoritmo RLS (algoritmo  *.
* minimo cuadrado recursivo, las antenas estan distribuidas en forma de celdas*.
* para lo cual requeriremos de los siguientes datos:                           *.
*                                                                              *.
*  • la señal aplicada a la entrada del filtro,   ej  Xn=sin((2*pi*k)/N)    *.
*  • el numero de antenas  x                                                   *.
*  • la señal de referencia, d(k)=cos((2*pi*k)/N);                           *.
*  • el ruido de las diferentes antenas, (hay que tener presente que el       *.
*    algoritmo LMS en su estructura internamente produce un ruido del 10      *.
*    aparte de este ruido puede haber otros.  n(f)                            *.
*  • el orden del filtro FIR cuyos coeficientes se van a ajustar, orden       *.
*  • el número de iteraciones en el aprendizaje, N (este número se hará       *.
*    hasta que cumpla el error)                                               *.
*  • la constante de ajuste   u                                               *.
*  • el vector de pesos inicial. Los voltajes de las antenas V(f)             *.
*  • el error deseado (puede ser ingresado por teclado) error                 *.
*                                                                              *.
*  A su salida devolverá:                                                      *.
*  • la salida en cada iteración durante el proceso de ajuste, y(t)           *.
*  • la curva de aprendizaje (grafico de la curva)                            *.
*  • los vectores de pesos obtenidos durante el proceso. W(t)                 *.
*  • la Potencia de la antena P(k)                                            *.
*  • el numero de iteraciones en que converge la curva y se cumple el valor   *.
*    del error.  j                                                            *.
* alumno: Juan Francisco Alvarez Alvarado                                      *.
* alumno: Maribel del Rosario Chuez Gonzales                                   *.
*                                                                              *.
* Profesor :Ing. Pedro Vargas                                                  *.
*                                                                              *.
********************************************************************************.
```

Pantalla 4.16 Datos del programa problema1_rls_c

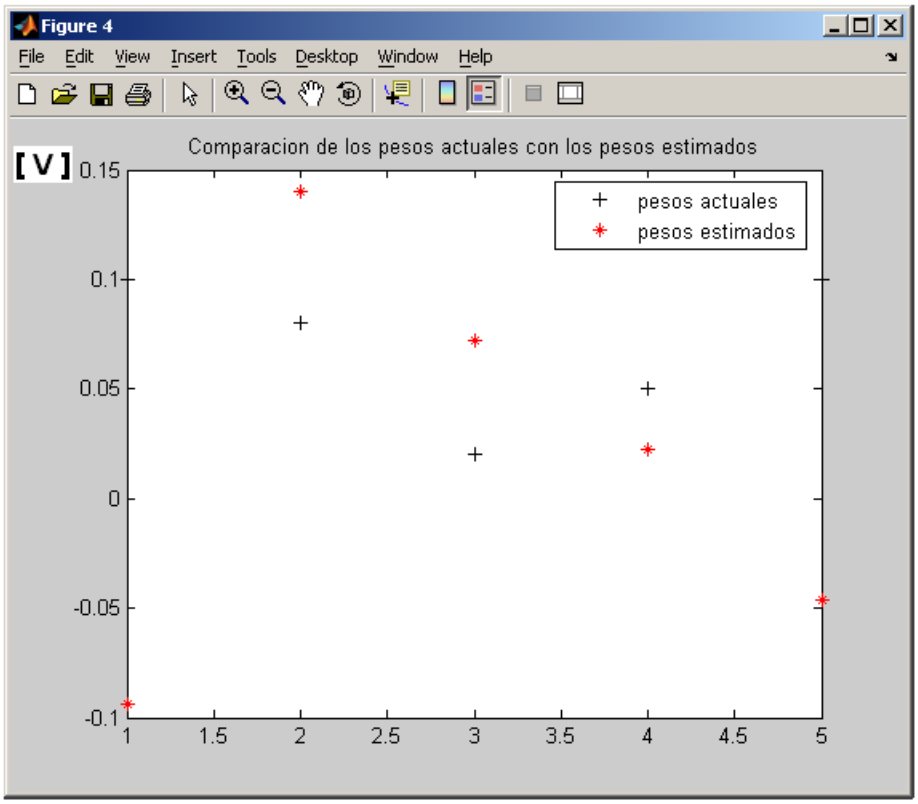
```
 ingrese el orden del filtro, cuyo coeficiente se va ha ajustar....5

orden =

     5

 ingrese el numero de antenas entre 1 y 15
 ingrese el numero de antenas entre 1,2,3,4,5,6,7,8,9,10,11,12,13,14 o 15....5
ingrese el voltaje1 (en v)..... voltaje1=0.10
 ingrese la voltaje2 (en V).... voltaje2=0.08
 ingrese la voltaje3 (en V).....voltaje3=0.02
 ingrese la voltaje4 (en v)..... voltaje4=0.05
 ingrese la voltaje5 (en v)..... voltaje5=0.10
ingrese el ruido en la antena1.....ruido1=0.01
ingrese el ruido en la antena2.....ruido2=0.01
ingrese el ruido en la antena3.....ruido3=0.01
ingrese el ruido en la antena4.....ruido4=0.02
ingrese el ruido en la antena5.....ruido5=0.02
si desea que el ruido adicional vaya a la señal a la entrada digite 1
 de lo contrario se considerara que el ruido adicional se agregara a la señal de referencia
opcion_ruido=1
ingrese el error del sistema=0.0000000000001

h =

    0.1000
    0.0800
    0.0200
    0.0500
    0.1000
```

Pantalla 4.17 Potencias obtenidas con el programa problema1_rls_c

```
potencia =

    0.4998

potencia1 =

    0.0170

potencia2 =

    0.0130

potencia3 =

    0.0058

potencia4 =

    0.0098

potencia5 =

    0.0193
```

Pantalla 4.18 Ejemplo problema1_rls_c. Comparación de los pesos estimados

Ejemplo problema1_rls_c
comparacion de los pesos estimados de una señal senosoidal con los pesos reales o actuales de una señal senosoidal utilizando un algoritmo rls, orden=5, numero pesos=5



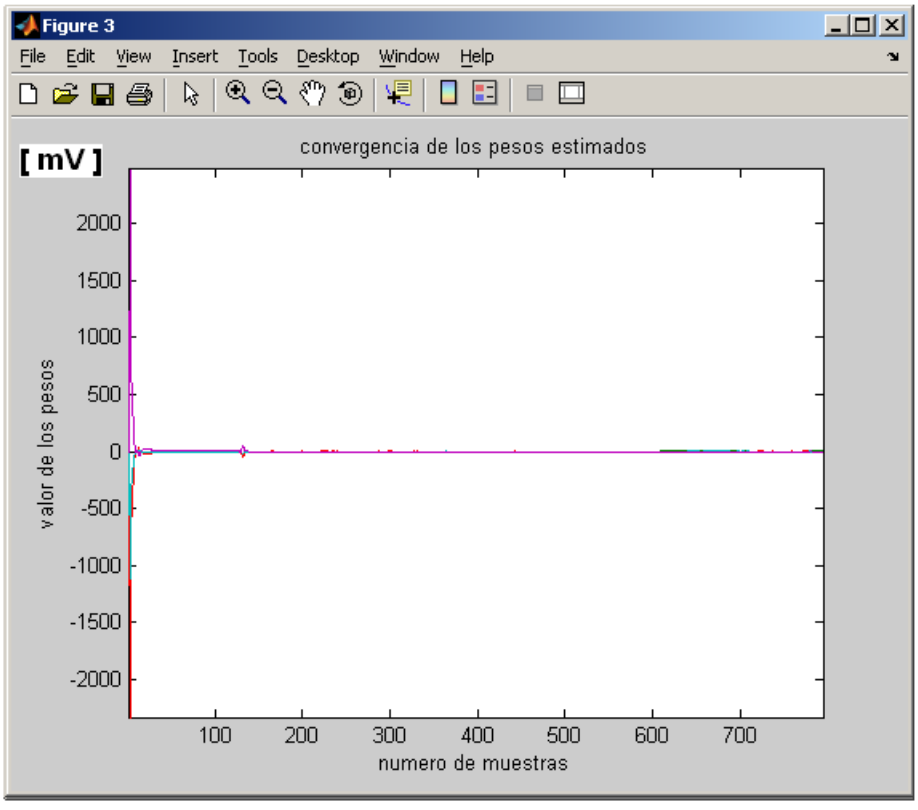
Eje de las x constante, eje de las y (amplitud de la señal en voltios). Amplitud_señal vs Valor constante

Considerando que las señales senosoidales de la señal es de 1 y los puntos escogidos son del orden de 0.10. 0.20 o 0.30 el error está entre 5 y 20 %. Dándonos un error muy cercano al producido al lms, pero superior al esperado, de 5 valores 2 tenian un error de casi 20%, 2 valores un error de casi 10% lo cual es un error igual al del algoritmo lms y

solo un valor dio un error menor a 5%. Revisando la literatura del algoritmo rls dice que si se presenta un porcentaje de error mayor al esperado se recomienda normalizar la ecuación , y también que se use valores más exactos para el filtro rls.

Pantalla 4.19 Ejemplo problema1_rls_c. Covergencia de los pesos estimados

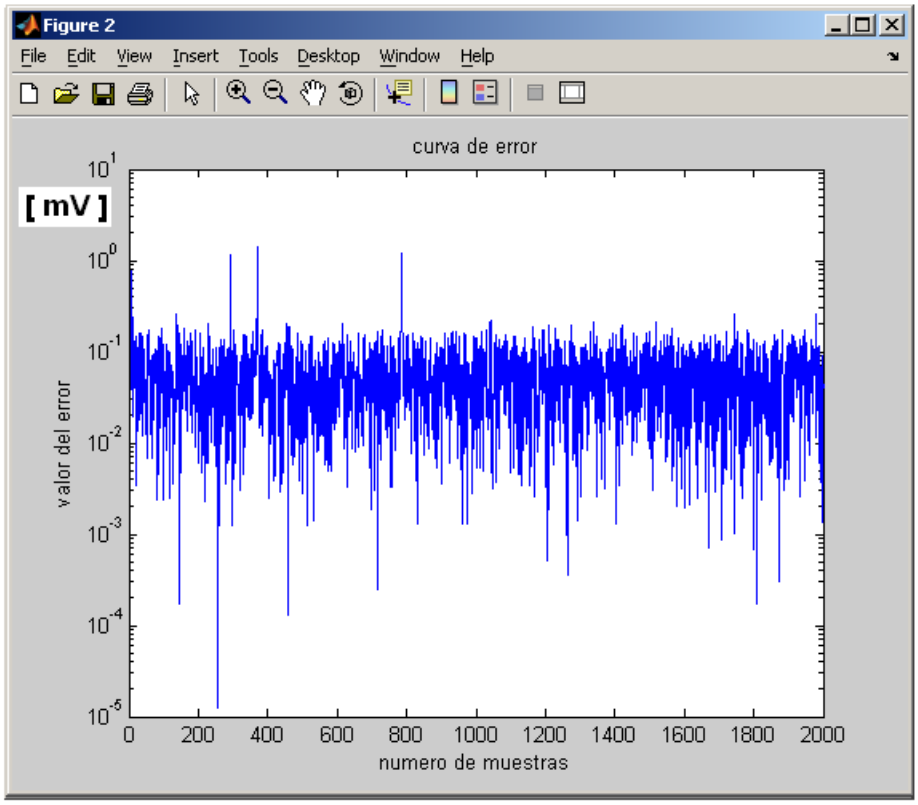
Ejemplo problema1_rls_c
Grafico de la convergencia de los pesos estimados en una señal senosoidal utilizando un algoritmo rls, orden=5, numero de pesos= 5



Valor de los pesos vs numero de muestras

Pantalla 4.20 Ejemplo problema1_rls_c. Curva de error de una señal senosoidal en un algoritmo rls

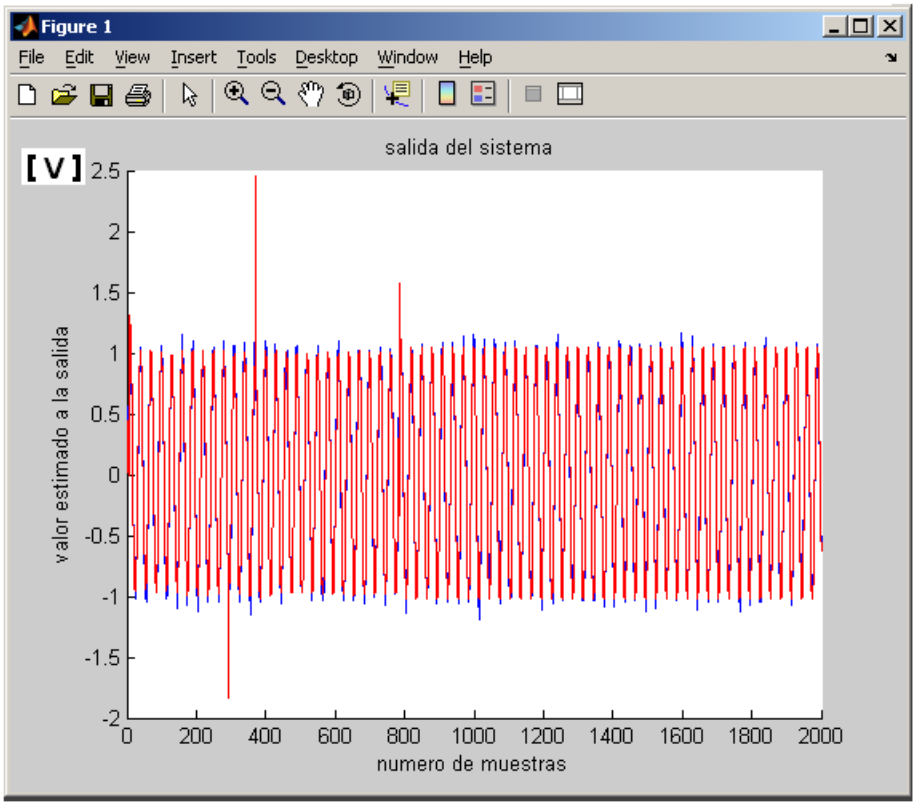
Ejemplo problema1_rls_c



Amplitud de la señal (voltios) vs numero de muestras

Pantalla 4.21 Ejemplo problema1_rls_c. Comparación entre la salida del sistema de una señal senosoidal (rls) con la señal de referencia del sistema

Ejemplo problema1_rls_c



Amplitud de la señal (voltios) vs numero de muestras
señal de referencia (color azul)
señal obtenida por el algoritmo rls (color rojo)

Problema2_rls_a

Pantalla 4.22 Problema2_rls_a

```
   * PROBLEMA2_rls.- Determinar los pesos, el numero de iteraciones y la potencia*.
   * de un grupo de antenas inteligentes utilizando el algoritmo RLS (algoritmo  *.
   * minimo cuadrado recursivo, las antenas estan distribuidas en forma de celdas*.
   * para lo cual requeriremos de los siguientes datos:                           *.
   *                                                                              *.
   *  • la señal aplicada a la entrada del filtro,   ej  Xn=sin((2*pi*k)/N)       *.
   *  • el numero de antenas  x                                                   *.
   *  • la señal de referencia, d(k)=cos((2*pi*k)/N);                           *.
   *  • el ruido de las diferentes antenas, (hay que tener presente que el        *.
   *    algoritmo LMS en su estructura internamente produce un ruido del 10       *.
   *    aparte de este ruido puede haber otros.  n(f)                             *.
   *  • el orden del filtro FIR cuyos coeficientes se van a ajustar, orden        *.
   *  • el número de iteraciones en el aprendizaje, N (este número se hará        *.
   *    hasta que cumpla el error)                                                *.
   *  • la constante de ajuste   u                                                *.
   *  • el vector de pesos inicial. Los voltajes de las antenas V(f)              *.
   *  • el error deseado (puede ser ingresado por teclado) error                  *.
   *                                                                              *.
   *  A su salida devolverá:                                                      *.
   *  • la salida en cada iteración durante el proceso de ajuste, y(t)            *.
   *  • la curva de aprendizaje (grafico de la curva)                             *.
   *  • los vectores de pesos obtenidos durante el proceso. W(t)                  *.
   *  • la Potencia de la antena P(k)                                             *.
   *  • el numero de iteraciones en que converge la curva y se cumple el valor    *.
   *    del error.  j                                                             *.
   * alumno: Juan Francisco Alvarez Alvarado                                      *.
   * alumno: Maribel del Rosario Chuez Gonzales                                   *.
   *                                                                              *.
   * Profesor :Ing. Pedro Vargas                                                  *.
   *                                                                              *.
   ********************************************************************************.
teclee enter para proseguir
```

Pantalla 4.23 Datos del programa problema2_rls_a

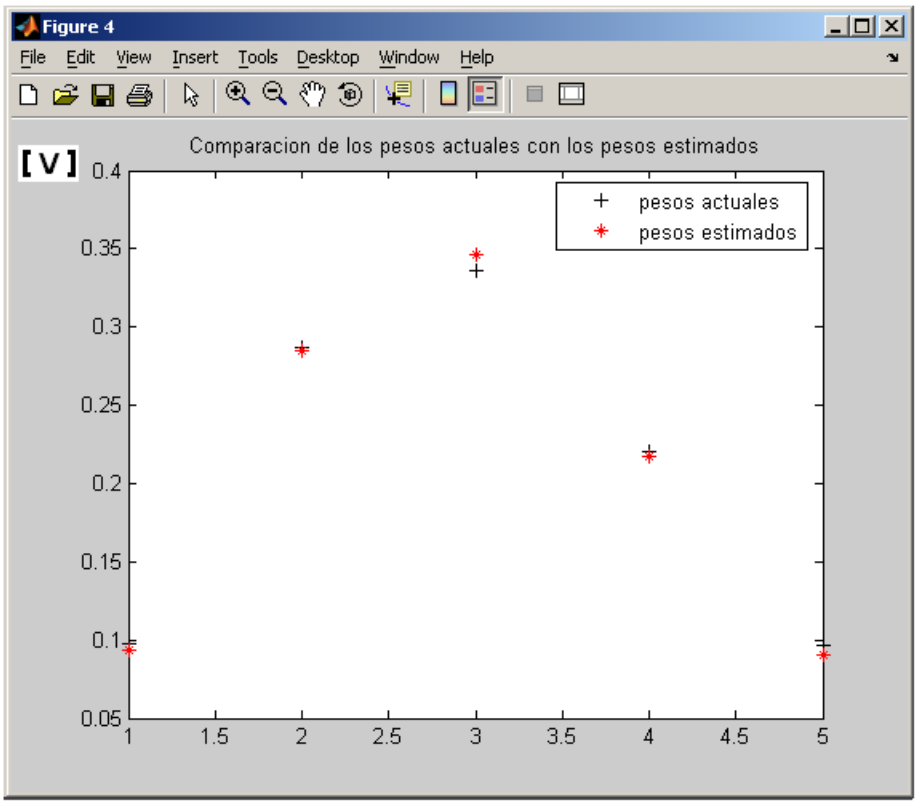
```
 ingrese el orden del filtro, cuyo coeficiente se va ha ajustar....5

orden =

     5

 ingrese el numero de antenas entre 1 y 15
 ingrese el numero de antenas entre 1,2,3,4,5,6,7,8,9,10,11,12,13,14 o 15....5
ingrese el voltaje1 (en v)..... voltaje1=0.09763
 ingrese la voltaje2 (en V).... voltaje2=0.28731
 ingrese la voltaje3 (en V).....voltaje3=0.335965
 ingrese la voltaje4 (en v)..... voltaje4=0.2209
 ingrese la voltaje5 (en v)..... voltaje5=0.0963
ingrese el ruido en la antena1.....ruido1=0.01
ingrese el ruido en la antena2.....ruido2=0.01
ingrese el ruido en la antena3.....ruido3=0.01
ingrese el ruido en la antena4.....ruido4=0.02
ingrese el ruido en la antena5.....ruido5=0.02
si desea que el ruido adicional vaya a la señal a la entrada digite 1
 de lo contrario se considerara que el ruido adicional se agregara a la señal de referencia
opcion_ruido=1
ingrese el error del sistema=0.0000001

h =

    0.0976
    0.2873
    0.3360
    0.2209
    0.0963
```

Pantalla 4.24 Potencias obtenidas con el programa problema2_rls_a

```
potencia =

    0.2484

potencia1 =

    0.0141

potencia2 =

    0.0910

potencia3 =

    0.1224

potencia4 =

    0.0606

potencia5 =

    0.0160
```

Pantalla 4.25 Ejemplo problema2_rls_a. Comparación de los pesos estimados

Ejemplo problema2_rls_a
comparacion de los pesos estimados de una señal gaussiana con los pesos reales o actuales de una señal gaussiana utilizando un algoritmo rls, orden=5, numero pesos=5



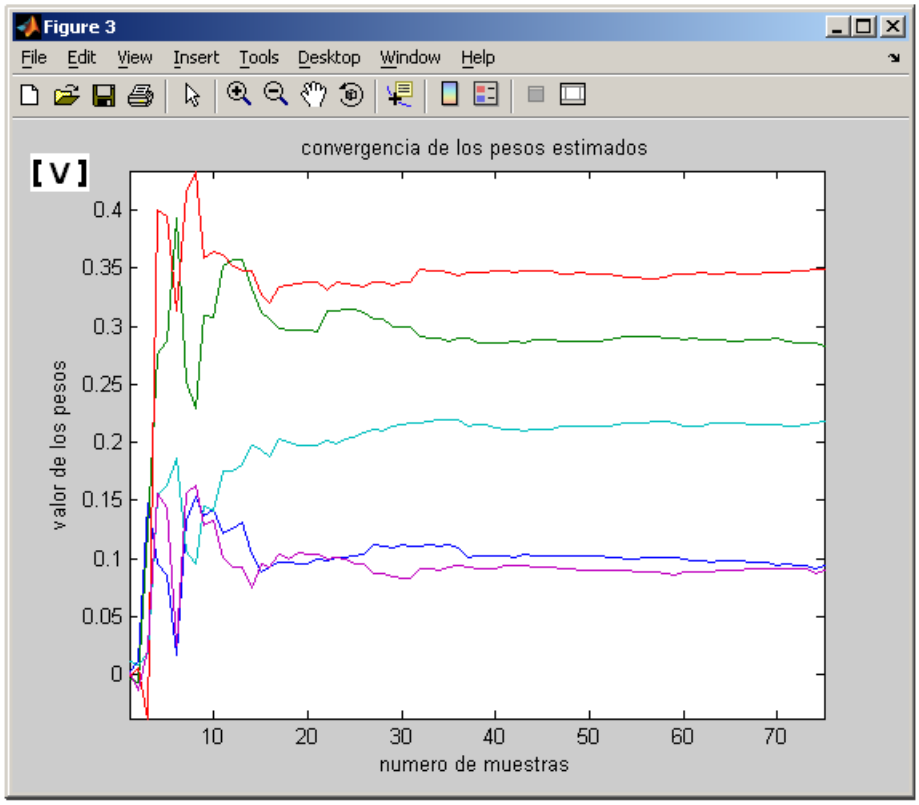
Eje de las x constante, eje de las y (amplitud de la señal en voltios). Amplitud_señal vs Valor constante

Al ingresar una señal gaussiana disminuyo considerablemente el error, utilizamos valores recomendados por un demo de matlab, al usar otros valores el error aumento un poco, pero no excesivamente.

Pantalla 4.26 Ejemplo problema2_rls_a. Covergencia de los pesos estimados

Ejemplo problema2_rls_a

Grafico de la convergencia de los pesos estimados en una señal gaussiana utilizando un algoritmo rls, orden=5, numero de pesos= 5

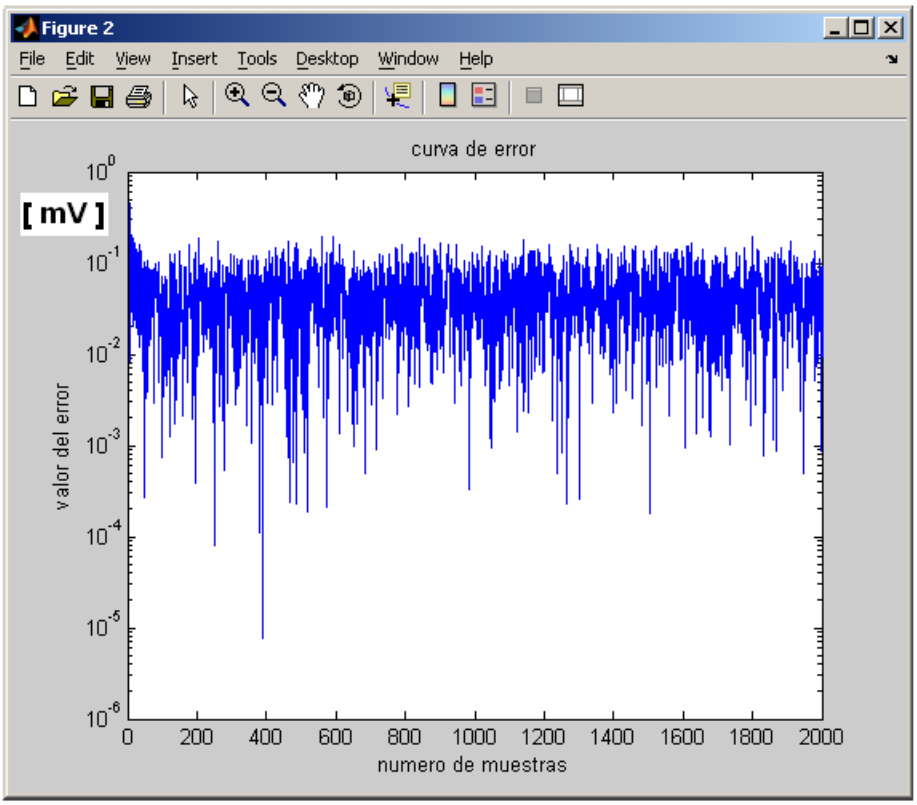


Valor de los pesos vs numero de muestras

Aquí observamos que la convergencia ocurre entre 10 y 20 muestras.

Pantalla 4.27 Ejemplo problema2_rls_a. Curva de error de una señal gaussiana en un algoritmo rls

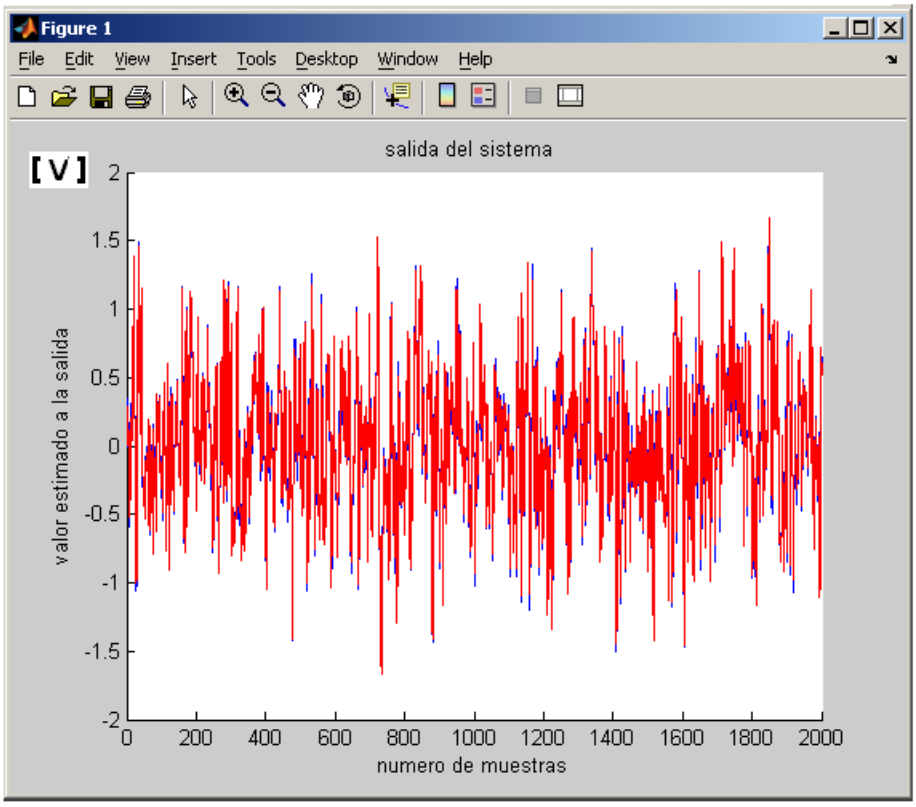
Ejemplo problema2_rls_a



Amplitud de la señal (voltios) vs numero de muestras

Pantalla 4.28 Ejemplo problema2_rls_a. Comparación entre la salida del sistema de una señal gaussiana (rls) con la señal de referencia del sistema

Ejemplo problema2_rls_a



Amplitud de la señal (voltios) vs numero de muestras
señal de referencia (color azul)
señal obtenida por el algoritmo rls (color rojo)

Problema2_rls_b

Pantalla 4.29 Problema2_rls_b

```
   * PROBLEMA2_rls.- Determinar los pesos, el numero de iteraciones y la potencia*.
   * de un grupo de antenas inteligentes utilizando el algoritmo RLS (algoritmo  *.
   * minimo cuadrado recursivo, las antenas estan distribuidas en forma de celdas*.
   * para lo cual requeriremos de los siguientes datos:                           *.
   *                                                                              *.
   *  • la señal aplicada a la entrada del filtro,   ej  Xn=sin((2*pi*k)/N)       *.
   *  • el numero de antenas  x                                                   *.
   *  • la señal de referencia, d(k)=cos((2*pi*k)/N);                           *.
   *  • el ruido de las diferentes antenas, (hay que tener presente que el        *.
   *    algoritmo LMS en su estructura internamente produce un ruido del 10       *.
   *    aparte de este ruido puede haber otros.  n(f)                             *.
   *  • el orden del filtro FIR cuyos coeficientes se van a ajustar, orden        *.
   *  • el número de iteraciones en el aprendizaje, N (este número se hará        *.
   *    hasta que cumpla el error)                                                *.
   *  • la constante de ajuste   u                                                *.
   *  • el vector de pesos inicial. Los voltajes de las antenas V(f)              *.
   *  • el error deseado (puede ser ingresado por teclado) error                  *.
   *                                                                              *.
   *  A su salida devolverá:                                                      *.
   *  • la salida en cada iteración durante el proceso de ajuste, y(t)            *.
   *  • la curva de aprendizaje (grafico de la curva)                             *.
   *  • los vectores de pesos obtenidos durante el proceso. W(t)                  *.
   *  • la Potencia de la antena P(k)                                             *.
   *  • el numero de iteraciones en que converge la curva y se cumple el valor    *.
   *    del error.  j                                                             *.
   * alumno: Juan Francisco Alvarez Alvarado                                      *.
   * alumno: Maribel del Rosario Chuez Gonzales                                   *.
   *                                                                              *.
   * Profesor :Ing. Pedro Vargas                                                  *.
   *                                                                              *.
   ********************************************************************************.
teclee enter para proseguir
```

Pantalla 4.30 Datos del programa problema2_rls_b

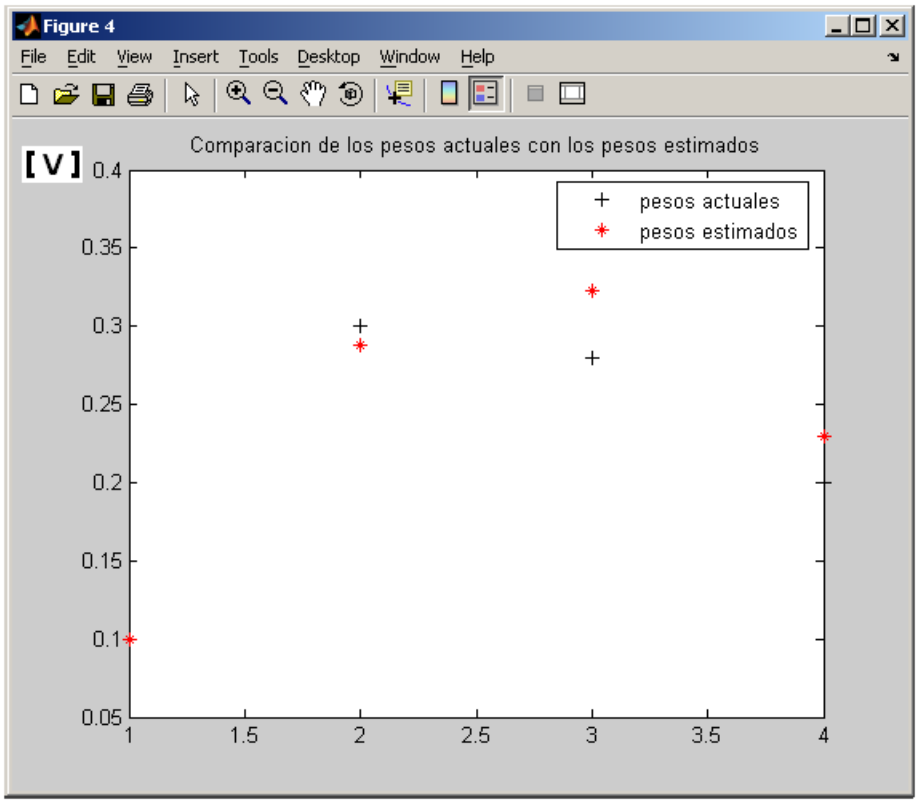
```
 ingrese el orden del filtro, cuyo coeficiente se va ha ajustar....4

orden =

     4

 ingrese el numero de antenas entre 1 y 15
 ingrese el numero de antenas entre 1,2,3,4,5,6,7,8,9,10,11,12,13,14 o 15....4
ingrese el voltaje1 (en v)..... voltaje1=0.10
 ingrese la voltaje2 (en V).... voltaje2=0.30
 ingrese la voltaje3 (en V).....voltaje3=0.28
 ingrese la voltaje4 (en v)..... voltaje4=0.20
ingrese el ruido en la antena1.....ruido1=0.01
ingrese el ruido en la antena2.....ruido2=0.01
ingrese el ruido en la antena3.....ruido3=0.02
ingrese el ruido en la antena4.....ruido4=0.02
si desea que el ruido adicional vaya a la señal a la entrada digite 1
 de lo contrario se considerara que el ruido adicional se agregara a la señal de referencia
opcion_ruido=2
ingrese el error del sistema=0.00000000000001

h =

    0.1000
    0.3000
    0.2800
    0.2000
```

Pantalla 4.31 Potencias obtenidas con el programa problema2_rls_b

```
potencia =

    0.2687

potencia1 =

    0.0146

potencia1 =

    0.0984

potencia3 =

    0.0923

potencia4 =

    0.0508
```

Pantalla 4.32 Ejemplo problema2_rls_b. Comparación de los pesos estimados

Ejemplo problema2_rls_b
comparacion de los pesos estimados de una señal gaussiana con los pesos reales o actuales de una señal gaussiana utilizando un algoritmo rls, orden=4, numero pesos=4



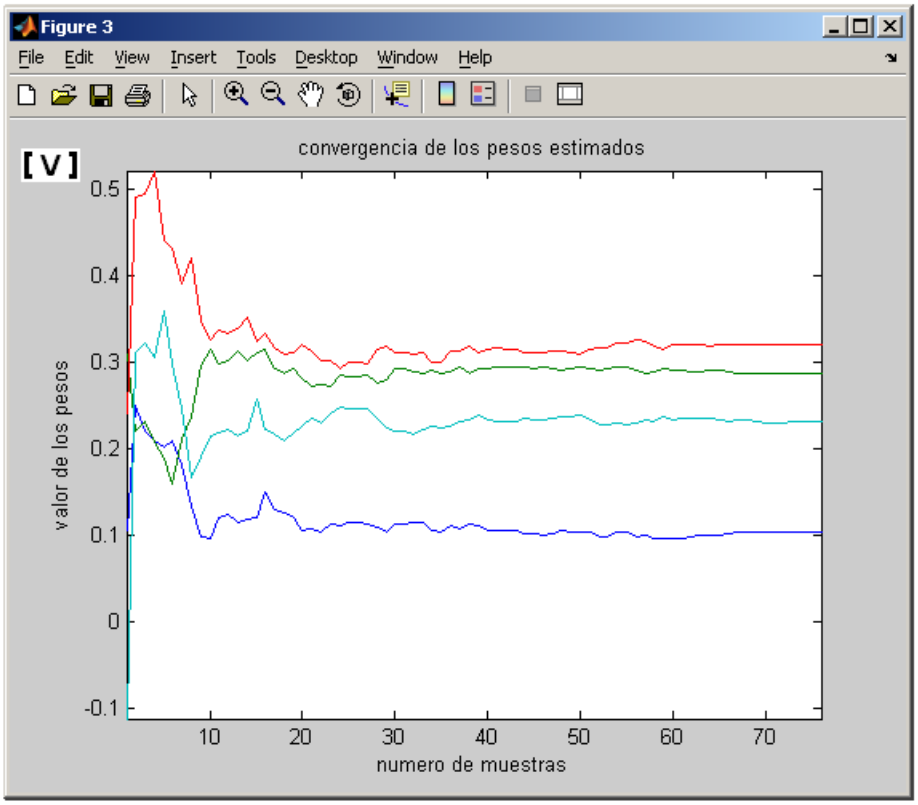
Eje de las x constante, eje de las y (amplitud de la señal en voltios). Amplitud_señal vs Valor constante

Aplicando una señal gaussiana aleatoria como señal de entrada, y el ruido adicional agregado a la salida del sistema, para un orden =4 y 4 valores de pesos originales se obtuvo un error bastante pequeño en 3 de los valores y 1 error un poco mayor en uno de los pesos.

Pantalla 4.33 Ejemplo problema2_rls_b. Covergencia de los pesos estimados

Ejemplo problema2_rls_b

Grafico de la convergencia de los pesos estimados en una señal gaussiana utilizando un algoritmo rls, orden=4, numero de pesos= 4

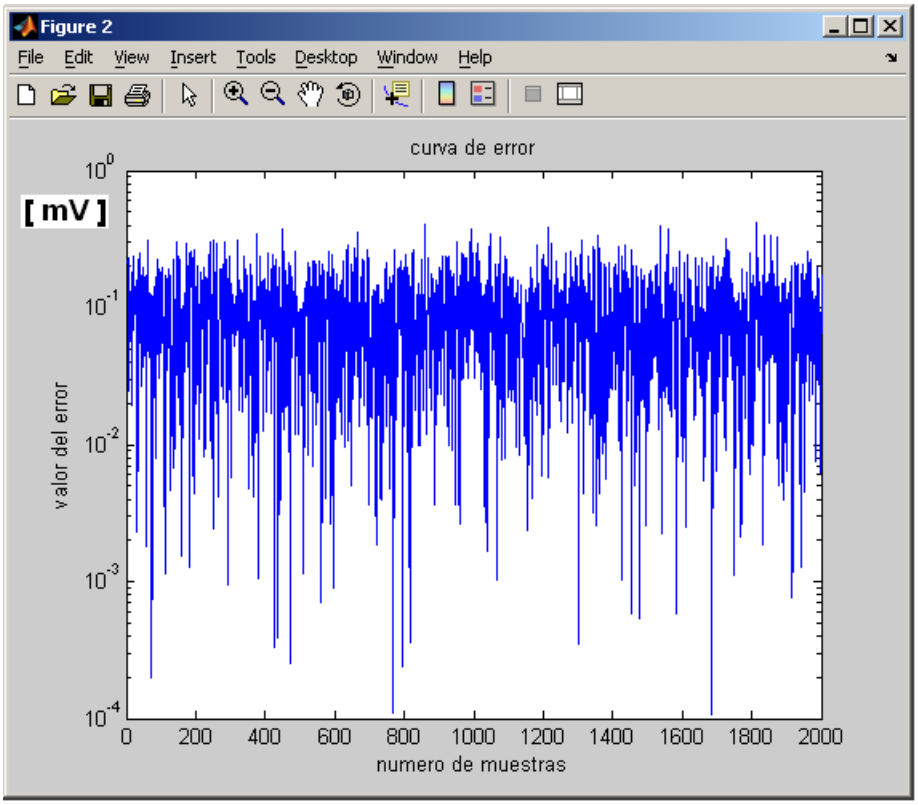


Valor de los pesos vs numero de muestras

La convergencia se logro entre 10 y 20 valores de muestras, lo cual es lo esperado para el algoritmo rls.

Pantalla 4.34 Ejemplo problema2_rls_b. Curva de error de una señal gaussiana en un algoritmo rls

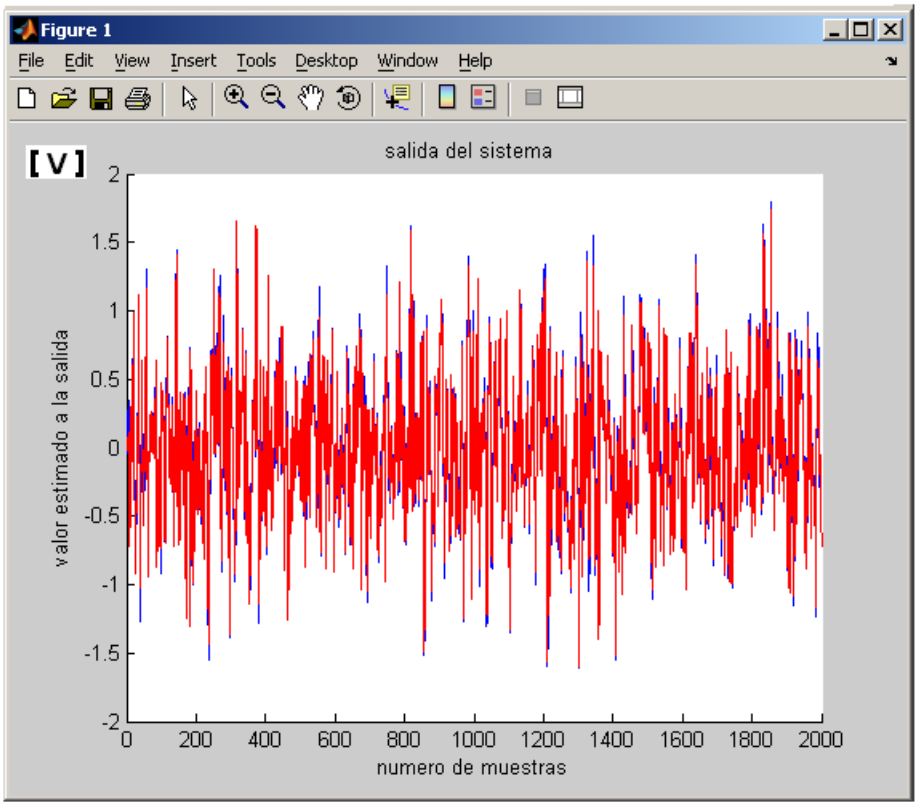
Ejemplo problema2_rls_b



Amplitud de la señal (voltios) vs numero de muestras

Pantalla 4.35 Ejemplo problema2_rls_b. Comparación entre la salida Del sistema de una señal gaussiana (rls) con la señal de referencia del sistema

Ejemplo problema2_rls_b



Amplitud de la señal (voltios) vs numero de muestras
señal de referencia (color azul)
señal obtenida por el algoritmo rls (color rojo)

Problema3_rls_a

Pantalla 4.36 Problema3_rls_a

```
   * PROBLEMA3_rls.- Determinar los pesos, el numero de iteraciones y la potencia*.
   * de un grupo de antenas inteligentes utilizando el algoritmo RLS (algoritmo  *.
   * minimo cuadrado recursivo, las antenas estan distribuidas en forma de celdas*.
   * para lo cual requeriremos de los siguientes datos:                           *.
   *                                                                              *.
   *  • la señal aplicada a la entrada del filtro,   ej  Xn=sin((2*pi*k)/N)       *.
   *  • el numero de antenas  x                                                   *.
   *  • la señal de referencia, d(k)=cos((2*pi*k)/N);                           *.
   *  • el ruido de las diferentes antenas, (hay que tener presente que el        *.
   *    algoritmo LMS en su estructura internamente produce un ruido del 10       *.
   *    aparte de este ruido puede haber otros.  n(f)                             *.
   *  • el orden del filtro FIR cuyos coeficientes se van a ajustar, orden        *.
   *  • el número de iteraciones en el aprendizaje, N (este número se hará        *.
   *    hasta que cumpla el error)                                                *.
   *  • la constante de ajuste   u                                                *.
   *  • el vector de pesos inicial. Los voltajes de las antenas V(f)              *.
   *  • el error deseado (puede ser ingresado por teclado) error                  *.
   *                                                                              *.
   *  A su salida devolverá:                                                      *.
   *  • la salida en cada iteración durante el proceso de ajuste, y(t)            *.
   *  • la curva de aprendizaje (grafico de la curva)                             *.
   *  • los vectores de pesos obtenidos durante el proceso. W(t)                  *.
   *  • la Potencia de la antena P(k)                                             *.
   *  • el numero de iteraciones en que converge la curva y se cumple el valor    *.
   *    del error.  j                                                             *.
   * alumno: Juan Francisco Alvarez Alvarado                                      *.
   * alumno: Maribel del Rosario Chuez Gonzales                                   *.
   *                                                                              *.
   * Profesor :Ing. Pedro Vargas                                                  *.
   *                                                                              *.
   ********************************************************************************.
teclee enter para proseguir
```

Pantalla 4.37 Datos del programa problema3_rls_a

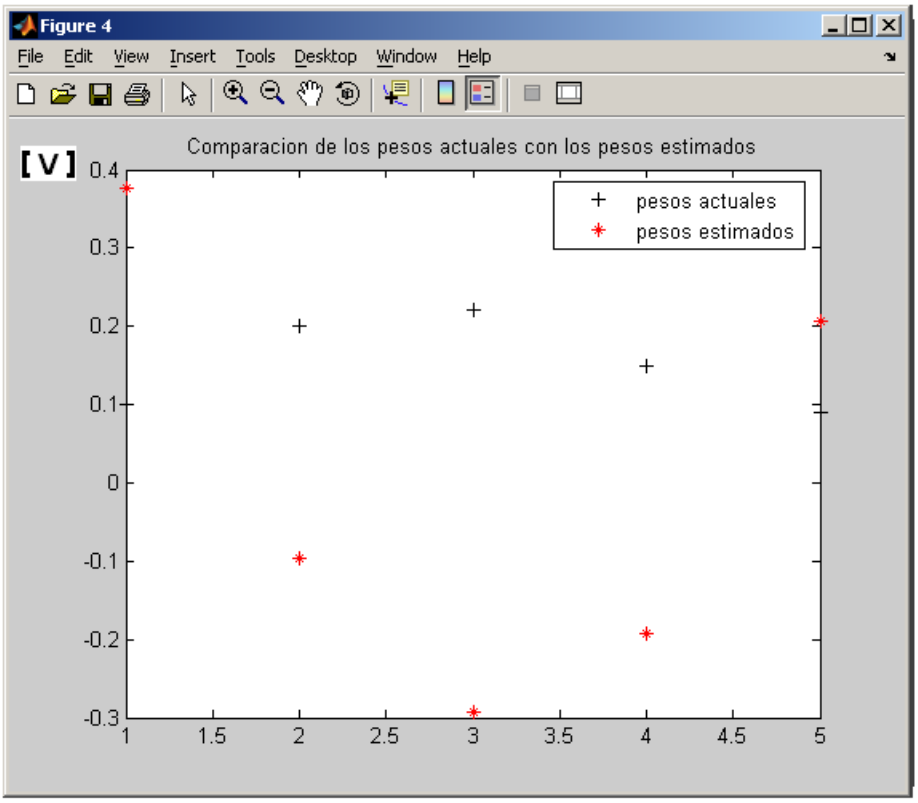
```
 ingrese el orden del filtro, cuyo coeficiente se va ha ajustar....5

orden =

     5

 ingrese el numero de antenas entre 1 y 15
 ingrese el numero de antenas entre 1,2,3,4,5,6,7,8,9,10,11,12,13,14 o 15....5
ingrese el voltaje1 (en v)..... voltaje1=0.10
 ingrese la voltaje2 (en V).... voltaje2=0.20
 ingrese la voltaje3 (en V).....voltaje3=0.22
 ingrese la voltaje4 (en v)..... voltaje4=0.15
 ingrese la voltaje5 (en v)..... voltaje5=0.09
ingrese el ruido en la antena1.....ruido1=0.01
ingrese el ruido en la antena2.....ruido2=0.01
ingrese el ruido en la antena3.....ruido3=0.01
ingrese el ruido en la antena4.....ruido4=0.02
ingrese el ruido en la antena5.....ruido5=0.02
si desea que el ruido adicional vaya a la señal a la entrada digite 1
 de lo contrario se considerara que el ruido adicional se agregara a la señal de referencia
opcion_ruido=1
ingrese el error del sistema=0.0000000000001
si desea que la señal de entrada se use como señal de referencia digite 1
de lo contrario se considerara otra señal diferente como señal de referencia
opcion_señal=2
```

Pantalla 4.38 Potencias obtenidas con el programa problema3_rls_a

```
potencia =

    0.5051

potencia1 =

    0.0179

potencia2 =

    0.0507

potencia3 =

    0.0597

potencia4 =

    0.0352

potencia5 =

    0.0179
```

Pantalla 4.39 Ejemplo problema3_rls_a. Comparación de los pesos estimados

Ejemplo problema3_rls_a
comparacion de los pesos estimados de una señal senosoidal con los pesos reales o actuales de una señal senosoidal utilizando un algoritmo rls, orden=5, numero pesos=5



Eje de las x constante, eje de las y (amplitud de la señal en voltios). Amplitud_señal vs Valor constante

Al aplicar la nueva señal de referencia se obtuvo un error del 10% en dos de la señales y un 20% en las otras tres señales. Es un error mayor al esperado, pero variando el valor de landa y delta se lo puede disminuir un poco.era de esperarse

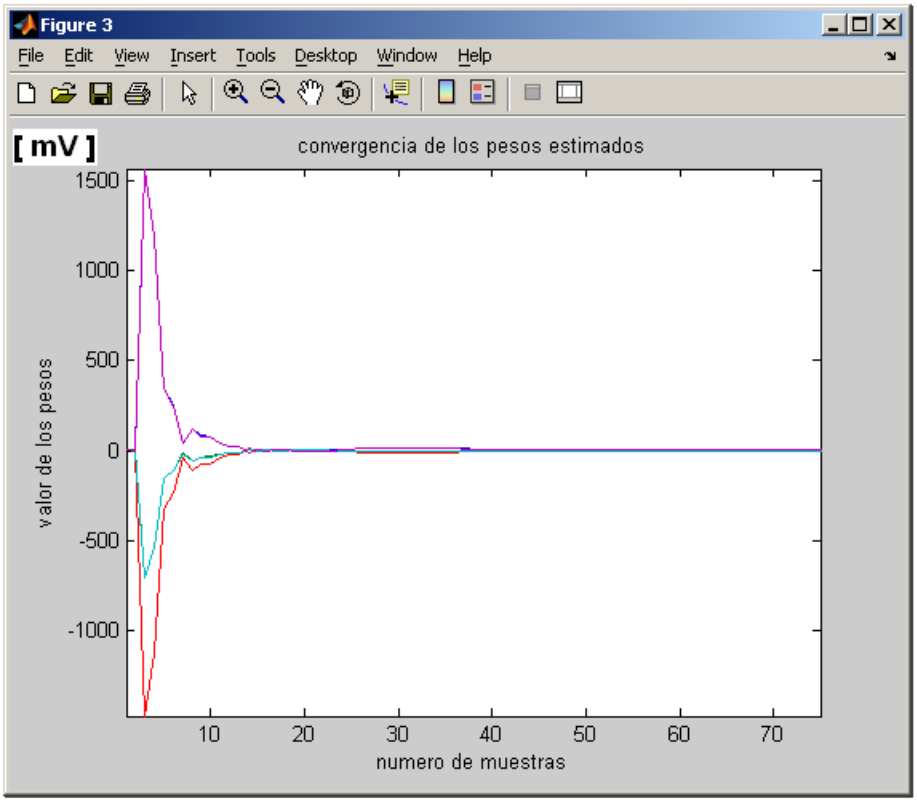
un aumento del error (nosotros suponíamos que llegaría al 10% igual al de algoritmo lms), pero fue un poco mayor.

Como comentario observamos en otras simulaciones que hicimos que se formaba una especie de campana invertida entre los valores reales y los calculados por el programa, pero solo sucedía ha veces, en la mayoría de los casos era paralela a la curva. Suponemos que debe de ocurrir esto por efecto de alguna de las propiedades del algoritmo rls

Pantalla 4.40 Ejemplo problema3_rls_a. Convergencia de los pesos estimados

Ejemplo problema3_rls_a

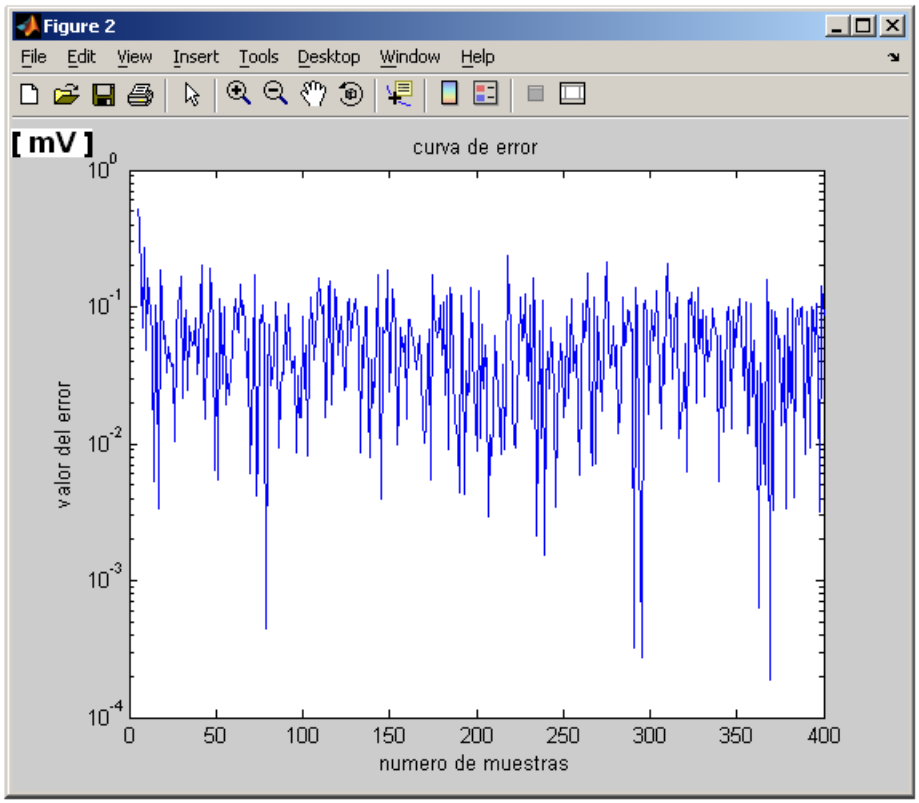
Grafico de la convergencia de los pesos estimados en una señal senosoidal utilizando un algoritmo rls, orden=5, numero de pesos=5



Valor de los pesos vs numero de muestras

Pantalla 4.41 Ejemplo problema3_rls_a. Curva de error de una señal senosoidal en un algoritmo rls

Ejemplo problema3_rls_a

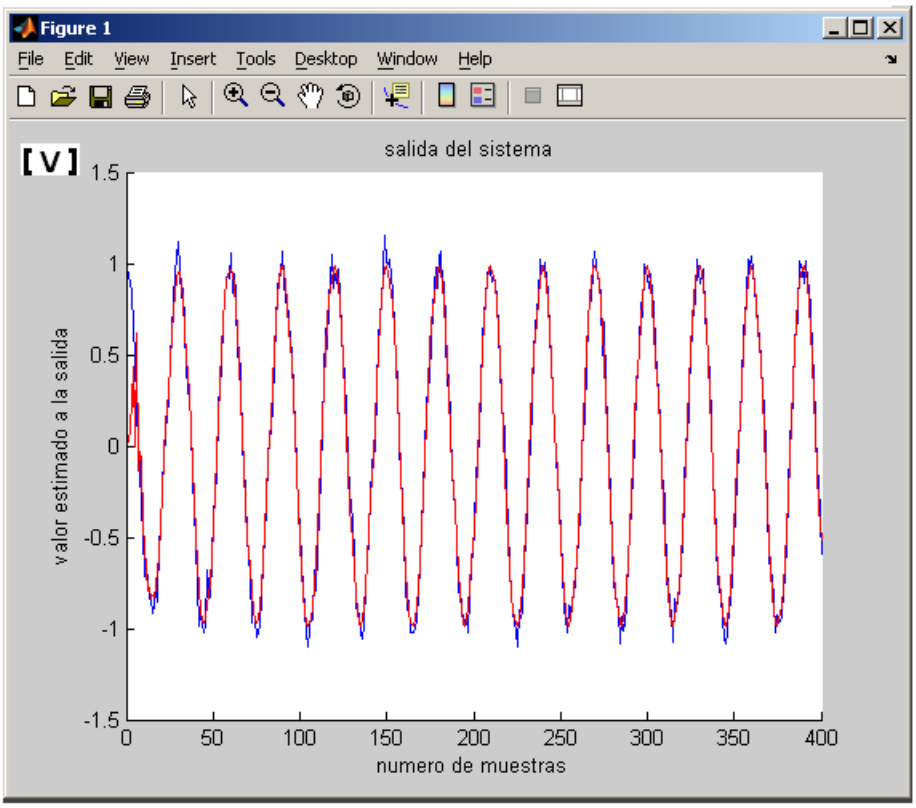


Amplitud de la señal (voltios) vs numero de muestras

Pantalla 4.42 Ejemplo problema3_rls_a.

Comparación entre la salida del sistema de una señal senosoidal (rls) con la señal de referencia del sistema

Ejemplo problema3_rls_a



Amplitud de la señal (voltios) vs numero de muestras
señal de referencia (color azul)
señal obtenida por el algoritmo rls (color rojo)

Problema3_rls_b

Pantalla 4.43 Problema3_rls_b

```
   * PROBLEMA3_rls.- Determinar los pesos, el numero de iteraciones y la potencia*.
   * de un grupo de antenas inteligentes utilizando el algoritmo RLS (algoritmo  *.
   * minimo cuadrado recursivo, las antenas estan distribuidas en forma de celdas*.
   * para lo cual requeriremos de los siguientes datos:                           *.
   *                                                                              *.
   *  • la señal aplicada a la entrada del filtro,   ej  Xn=sin((2*pi*k)/N)       *.
   *  • el numero de antenas  x                                                   *.
   *  • la señal de referencia, d(k)=cos((2*pi*k)/N);                           *.
   *  • el ruido de las diferentes antenas, (hay que tener presente que el        *.
   *    algoritmo LMS en su estructura internamente produce un ruido del 10       *.
   *    aparte de este ruido puede haber otros.  n(f)                             *.
   *  • el orden del filtro FIR cuyos coeficientes se van a ajustar, orden        *.
   *  • el número de iteraciones en el aprendizaje, N (este número se hará        *.
   *    hasta que cumpla el error)                                                *.
   *  • la constante de ajuste   u                                                *.
   *  • el vector de pesos inicial. Los voltajes de las antenas V(f)              *.
   *  • el error deseado (puede ser ingresado por teclado) error                  *.
   *                                                                              *.
   *  A su salida devolverá:                                                      *.
   *  • la salida en cada iteración durante el proceso de ajuste, y(t)            *.
   *  • la curva de aprendizaje (grafico de la curva)                             *.
   *  • los vectores de pesos obtenidos durante el proceso. W(t)                  *.
   *  • la Potencia de la antena P(k)                                             *.
   *  • el numero de iteraciones en que converge la curva y se cumple el valor    *.
   *    del error.  j                                                             *.
   * alumno: Juan Francisco Alvarez Alvarado                                      *.
   * alumno: Maribel del Rosario Chuez Gonzales                                   *.
   *                                                                              *.
   * Profesor :Ing. Pedro Vargas                                                  *.
   *                                                                              *.
   ********************************************************************************.
teclee enter para proseguir
```

Pantalla 4.44 Datos del programa problema3_rls_b

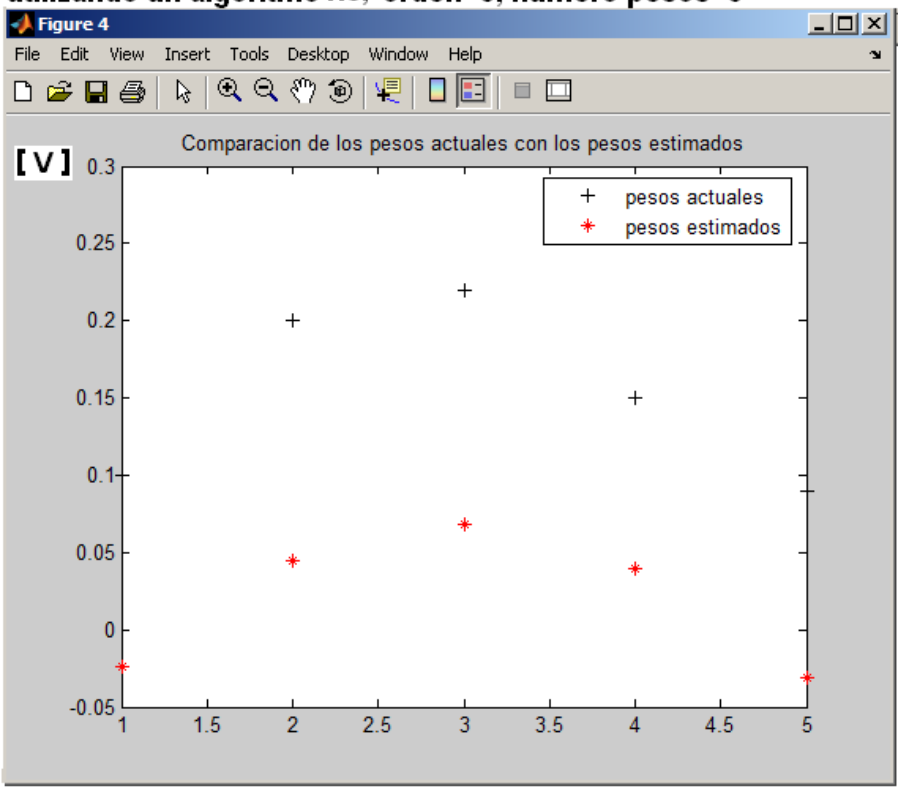
```
 ingrese el orden del filtro, cuyo coeficiente se va ha ajustar....5

orden =

     5

 ingrese el numero de antenas entre 1 y 15
 ingrese el numero de antenas entre 1,2,3,4,5,6,7,8,9,10,11,12,13,14 o 15....5
ingrese el voltaje1 (en v)..... voltaje1=0.10
 ingrese la voltaje2 (en V).... voltaje2=0.20
 ingrese la voltaje3 (en V).....voltaje3=0.22
 ingrese la voltaje4 (en v)..... voltaje4=0.15
 ingrese la voltaje5 (en v)..... voltaje5=0.09
ingrese el ruido en la antena1.....ruido1=0.01
ingrese el ruido en la antena2.....ruido2=0.01
ingrese el ruido en la antena3.....ruido3=0.02
ingrese el ruido en la antena4.....ruido4=0.02
ingrese el ruido en la antena5.....ruido5=0.02
si desea que el ruido adicional vaya a la señal a la entrada digite 1
 de lo contrario se considerara que el ruido adicional se agregara a la señal de referencia
opcion_ruido=1
ingrese el error del sistema=0.0000000000001
si desea que la señal de entrada se use como señal de referencia digite 1
de lo contrario se considerara otra señal diferente como señal de referencia
opcion_señal=1
```

Pantalla 4.45 Potencias obtenidas con el programa problema3_rls_b

```
potencia =

    0.5051

potencia1 =

    0.0179

potencia2 =

    0.0507

potencia3 =

    0.0597

potencia4 =

    0.0352

potencia5 =

    0.0179
```

Pantalla 4.46 Ejemplo problema3_rls_b. Comparación de los pesos estimados

Ejemplo problema3_rls_b
comparacion de los pesos estimados de una señal senosoidal con los pesos reales o actuales de una señal senosoidal utilizando un algoritmo rls, orden=5, numero pesos=5



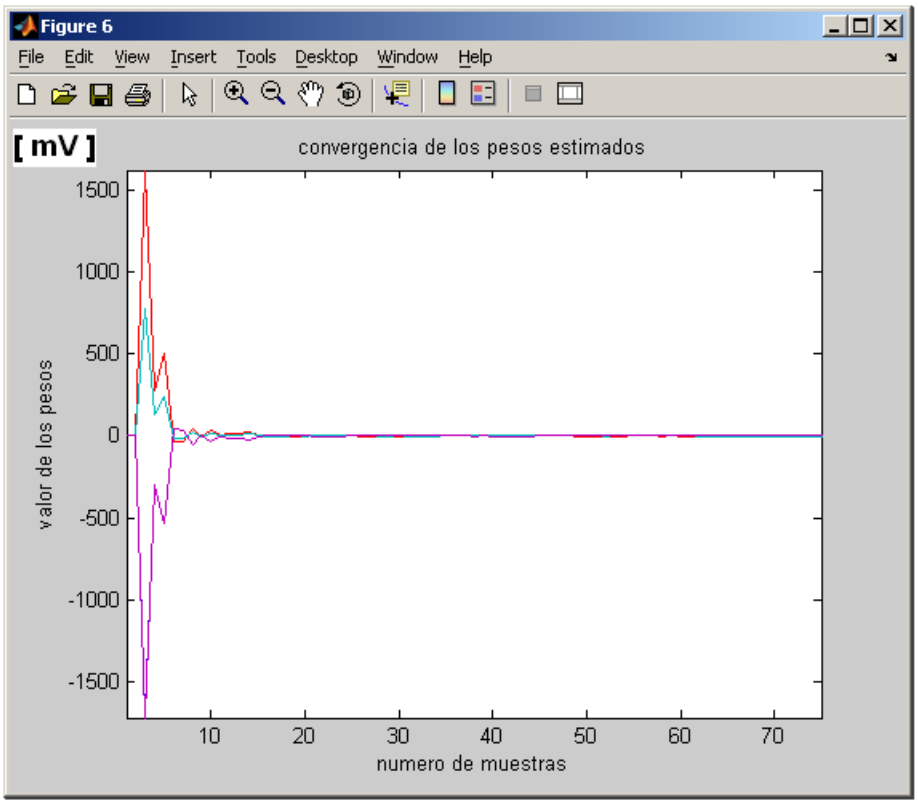
Eje de las x constante, eje de las y (amplitud de la señal en voltios). Amplitud señal vs Valor constante

Nos salio un resultado de valores cercanos a los valores reales, probablemente la curva de valores obtenidos es ligeramente más pequeña que la señal de referencia.

Pantalla 4.47 Ejemplo problema3_rls_b. Covergencia de los pesos estimados

Ejemplo problema3_rls_b

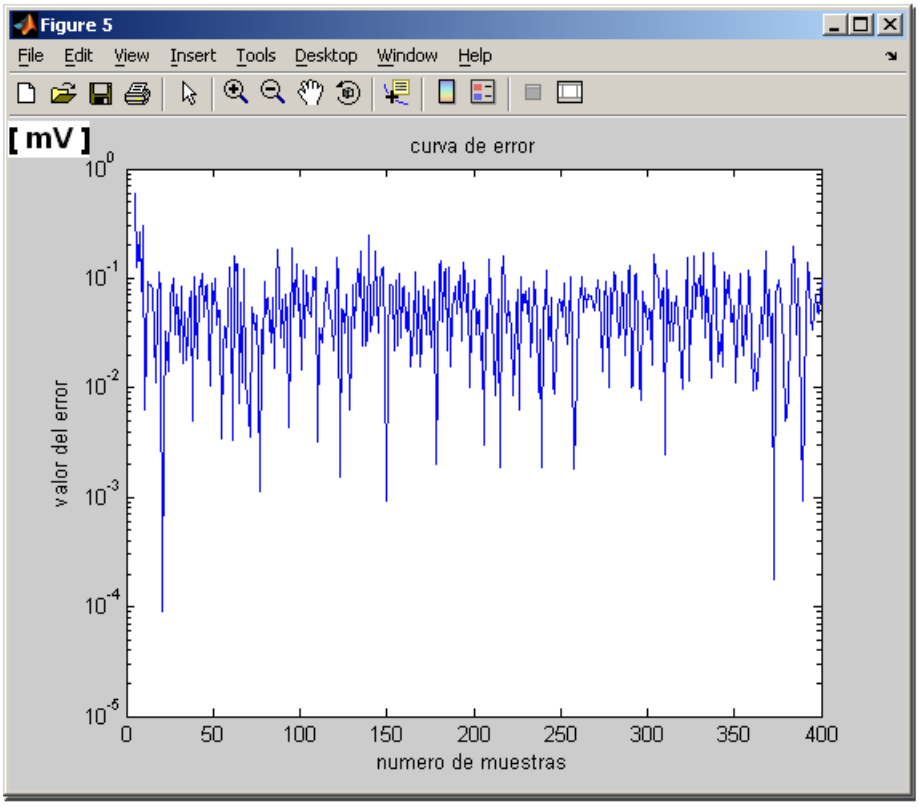
Grafico de la convergencia de los pesos estimados en una señal senosoidal utilizando un algoritmo rls, orden=5, numero de pesos=5



Valor de los pesos vs numero de muestras

Pantalla 4.48 Ejemplo problema3_rls_b. Curva de error de una señal senosoidal en un algoritmo rls

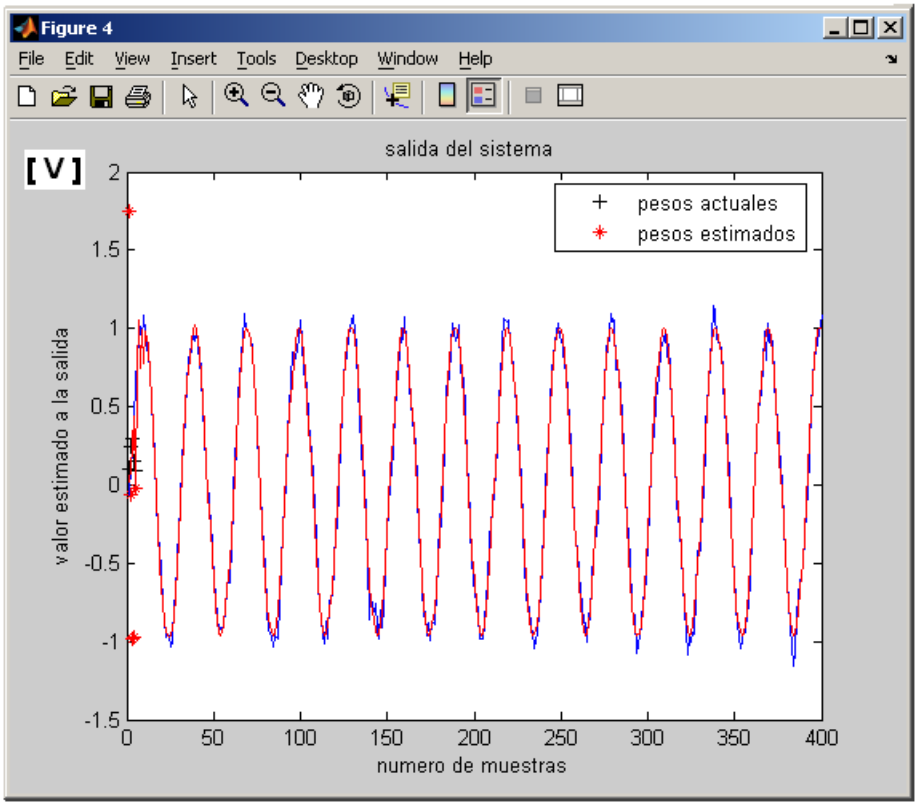
Ejemplo problema3_rls_b



Amplitud de la señal (voltios) vs numero de muestras

Pantalla 4.49 Ejemplo problema3_rls_b. Comparación entre la salida del sistema de una señal senosoidal (rls) con la señal de referencia del sistema

Ejemplo problema3_rls_b



Amplitud de la señal (voltios) vs numero de muestras
señal de referencia (color azul)
señal obtenida por el algoritmo rls (color rojo)

Problema3_cma_a

Pantalla 5.1 Problema3_cma_a

```
* PROBLEMA3_CMA-RLS.- Determinar los pesos, el numero de iteraciones y la     *.
* potencia de un grupo de antenas inteligentes utilizando el algoritmo CMA-RLS*.
* las antenas estan distribuidas en forma de celdas para lo cual requeriremos *.
* de los siguientes datos:                                                    *.
*                                                                             *.
*  • la señal aplicada a la entrada del filtro,   ej  Xn=sin((2*pi*k)/N)      *.
*  • el numero de antenas  x                                                  *.
*  • la señal de referencia, d(k)=cos((2*pi*k)/N);                          *.
*  • el ruido de las diferentes antenas, (hay que tener presente que el       *.
*    algoritmo LMS en su estructura internamente produce un ruido del 10      *.
*    aparte de este ruido puede haber otros.  n(f)                            *.
*  • el orden del filtro FIR cuyos coeficientes se van a ajustar, orden       *.
*  • el número de iteraciones en el aprendizaje, N (este número se hará       *.
*    hasta que cumpla el error)                                               *.
*  • la constante de ajuste   u                                               *.
*  • el vector de pesos inicial. Los voltajes de las antenas V(f)             *.
*  • el error deseado (puede ser ingresado por teclado) error                 *.
*                                                                             *.
*  A su salida devolverá:                                                     *.
*  • la salida en cada iteración durante el proceso de ajuste, y(t)           *.
*  • la curva de aprendizaje (grafico de la curva)                            *.
*  • los vectores de pesos obtenidos durante el proceso. W(t)                 *.
*  • la Potencia de la antena P(k)                                            *.
*  • el numero de iteraciones en que converge la curva y se cumple el valor   *.
*    del error.  j                                                            *.
* alumno: Juan Francisco Alvarez Alvarado                                     *.
* alumno: Maribel del Rosario Chuez Gonzales                                  *.
*                                                                             *.
* Profesor :Ing. Pedro Vargas                                                 *.
*                                                                             *.
*******************************************************************************.
```

Pantalla 5.2 Datos del programa problema3_cma_a

```
consideraremos un orden = 2  para la segunda etapa de un conformador de haz,    .
 esto es para que la ecuación de pesos que se utiliza en el programa sea         .
 la de un algoritmo CMA, este valor será constante en la ecuación, el valor      .
 del orden del filtro para determinar los valores de cota y de la potencia       .
 si variaran normalmente como si trabajaran en un algoritmo RLS                  .
                                                                                 .
 ingrese el orden del filtro, cuyo coeficiente se va ha ajustar....5

orden =

     5

 restriccion: ingrese un numero de antenas igual al del orden del filtro ingresado)
 ingrese el numero de antenas entre 1,2,3,4,5,6,7,8,9,10,11,12,13,14 o 15....5
ingrese el voltaje1 (en v)..... voltaje1=0.10
 ingrese la voltaje2 (en V).... voltaje2=0.12
 ingrese la voltaje3 (en V).....voltaje3=0.2
 ingrese la voltaje4 (en v)..... voltaje4=0.25
 ingrese la voltaje5 (en v)..... voltaje5=-0.2
ingrese el ruido en la antena1.....ruido1=0.01
ingrese el ruido en la antena2.....ruido2=0.01
ingrese el ruido en la antena3.....ruido3=0.01
ingrese el ruido en la antena4.....ruido4=0.02
ingrese el ruido en la antena5.....ruido5=0.02
si desea que el ruido adicional vaya a la señal a la entrada digite 1
 de lo contrario se considerara que el ruido adicional se agregara a la señal de referencia
opcion_ruido=1
ingrese el error del sistema=0.0000001
si desea que la señal de entrada se use como señal de referencia digite 1
de lo contrario se considerara otra señal diferente como señal de referencia
opcion_señal=1
```

Pantalla 5.3 Potencias obtenidas con el programa problema3_cma_a

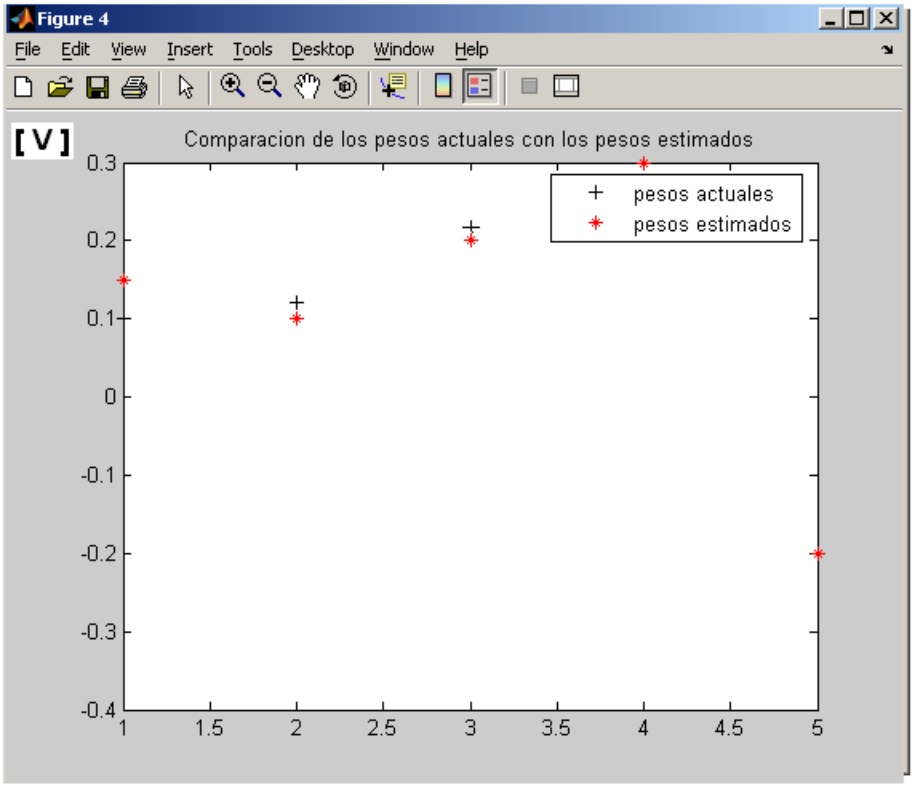
```
potencia =

    0.5039

potencia1 =

    0.0150

potencia2 =

    0.0195

potencia3 =

    0.0451

potencia4 =

    0.0728

potencia5 =

    0.0408
```

Nota utilizamos pesos iniciales de w= [0.15 0.10 0.20 0.3 -0.20] para evaluar dentro del algoritmo cma-rls

El grafico de la convergencia si se comporto más o menos dentro de lo esperado, pero las demás curvas se comportaron de una manera muy errática, distinta a los obtenidos en los algoritmo lms y rls. Pero este comportamiento de las curvas de error y de salida pueden ser características del algoritmo cma.

Pantalla 5.4 Ejemplo problema3_cma_a. Comparación de los pesos estimados

Ejemplo problema3_cma_a
comparacion de los pesos estimados de una señal senosoidal con los pesos reales o actuales de una señal senosoidal utilizando un algoritmo cma,orden=5, numero pesos=5

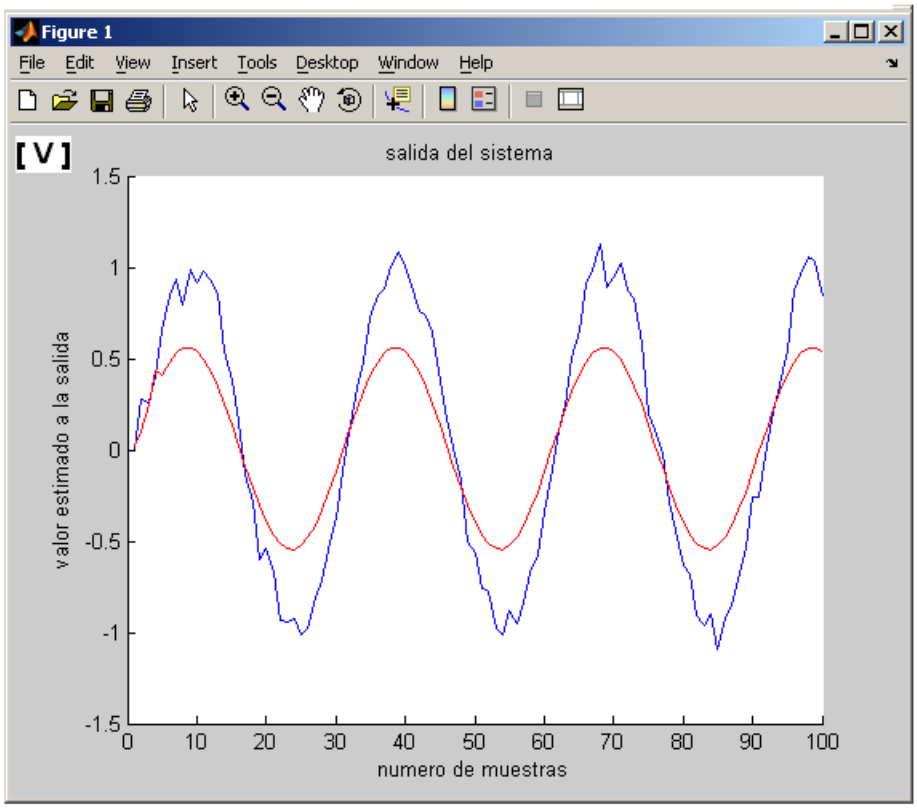


Eje de las x constante, eje de las y (amplitud de la señal en voltios). Amplitud_señal vs Valor constante

Nota.- se evaluó esta curva para un rango de seguridad de N=40, al querer usar una de N= 80 los valores se dispararon bastante.

Pantalla 5.5 Ejemplo problema3_cma_a. Comparación entre la salida del sistema de una señal senosoidal (cma) con la señal de referencia del sistema

Ejemplo problema3_cma_a

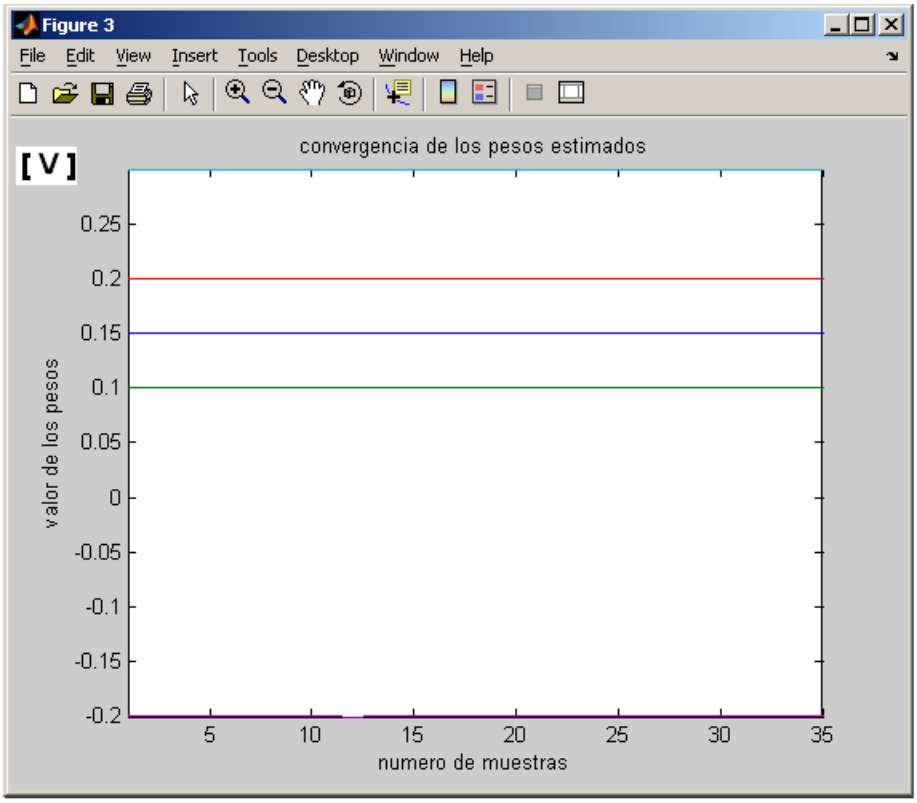


Amplitud de la señal (voltios) vs numero de muestras
señal de referencia (color azul)
señal obtenida por el algoritmo cma(color rojo)

seria conveniente usar solamente rangos de seguridad pequeños de entre N=20 y N=40. Para el algoritmo cma-rls.

Pantalla 5.6 Ejemplo problema3_cma_a. Covergencia de los pesos estimados

Ejemplo problema3_cma_a
Grafico de la convergencia de los pesos estimados en una señal senosoidal utilizando un algoritmo cma,orden=5, numero de pesos=5

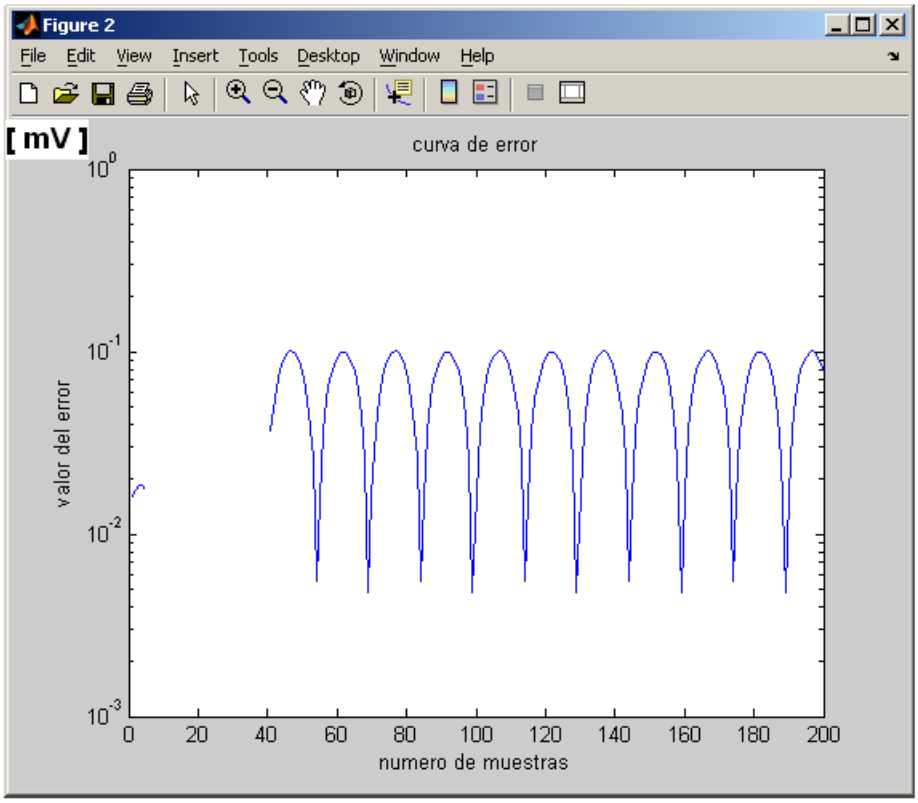


Valor de los pesos vs numero de muestras

El programa supone que los valores convergen inmediatamente, desde el primer momento, después veremos que esto no se cumple en la realidad, es decir que este grafico que se evalúa en el algoritmo rls correctamente, en el algoritmo cma da un resultado incorrecto.

Pantalla 5.7 Ejemplo problema3_cma_a. Curva de error de una señal senosoidal en un algoritmo cma

Ejemplo problema3_cma_a



Amplitud de la señal (voltios) vs numero de muestras

En la curva de error vemos que en la zona de seguridad de N=40 no se ven la señal de error , la zona de seguridad sirve para ayudar a la convergencia de la señal, para que salga mejor el valor de y (valor a la salida del filtro) a la salida del sistema, pero en el algoritmo cma vemos que causa que el valor

de la grafica de error de sistema no se vea. Esta es una característica muy interesante. Y un resultado inesperado. Esto es algo que no ocurre con los algoritmos lms y rls.

Problema3_cma_b

Pantalla 5.8 Problema3_cma_b

```
* PROBLEMA3_CMA-RLS.- Determinar los pesos, el numero de iteraciones y la     *.
* potencia de un grupo de antenas inteligentes utilizando el algoritmo CMA-RLS*.
* las antenas estan distribuidas en forma de celdas para lo cual requeriremos *.
* de los siguientes datos:                                                    *.
*                                                                             *.
*  • la señal aplicada a la entrada del filtro,   ej  Xn=sin((2*pi*k)/N)      *.
*  • el numero de antenas  x                                                  *.
*  • la señal de referencia, d(k)=cos((2*pi*k)/N);                          *.
*  • el ruido de las diferentes antenas, (hay que tener presente que el       *.
*    algoritmo LMS en su estructura internamente produce un ruido del 10      *.
*    aparte de este ruido puede haber otros.  n(f)                            *.
*  • el orden del filtro FIR cuyos coeficientes se van a ajustar, orden       *.
*  • el número de iteraciones en el aprendizaje, N (este número se hará       *.
*    hasta que cumpla el error)                                               *.
*  • la constante de ajuste   u                                               *.
*  • el vector de pesos inicial. Los voltajes de las antenas V(f)             *.
*  • el error deseado (puede ser ingresado por teclado) error                 *.
*                                                                             *.
*  A su salida devolverá:                                                     *.
*  • la salida en cada iteración durante el proceso de ajuste, y(t)           *.
*  • la curva de aprendizaje (grafico de la curva)                            *.
*  • los vectores de pesos obtenidos durante el proceso. W(t)                 *.
*  • la Potencia de la antena P(k)                                            *.
*  • el numero de iteraciones en que converge la curva y se cumple el valor   *.
*    del error.  j                                                            *.
* alumno: Juan Francisco Alvarez Alvarado                                     *.
* alumno: Maribel del Rosario Chuez Gonzales                                  *.
*                                                                             *.
* Profesor :Ing. Pedro Vargas                                                 *.
*                                                                             *.
*******************************************************************************.
```

Pantalla 5.9 Datos del programa problema3_cma_b

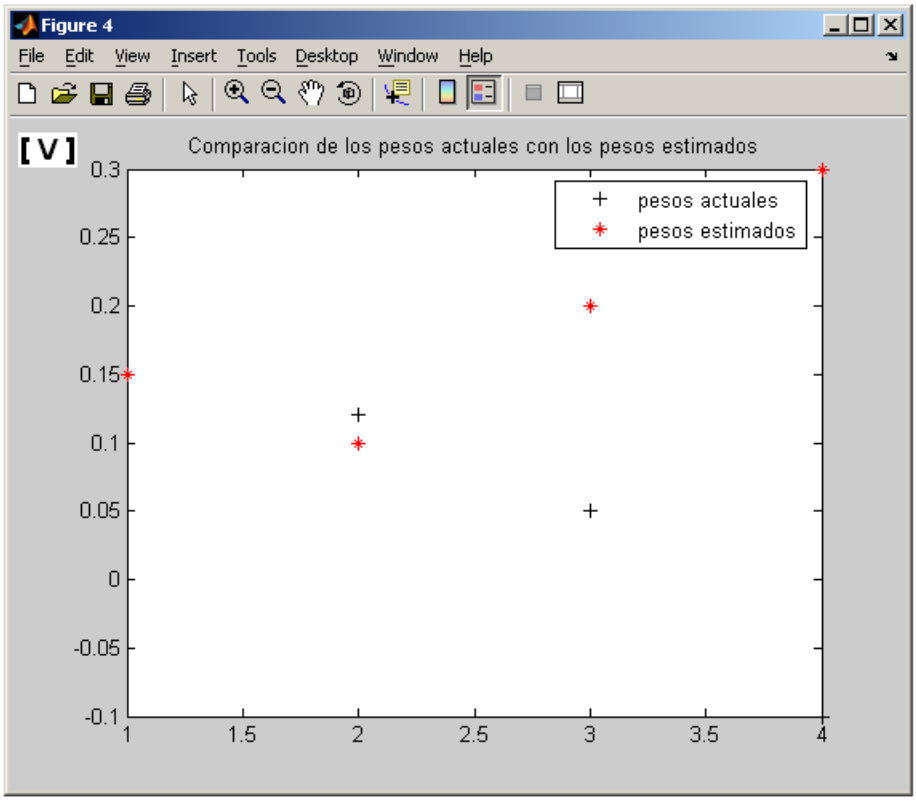
```
consideraremos un orden = 2  para la segunda etapa de un conformador de haz,    .
 esto es para que la ecuación de pesos que se utiliza en el programa sea        .
 la de un algoritmo CMA, este valor será constante en la ecuación, el valor     .
 del orden del filtro para determinar los valores de cota y de la potencia      .
 si variaran normalmente como si trabajaran en un algoritmo RLS                 .
                                                                                .
 ingrese el orden del filtro, cuyo coeficiente se va ha ajustar....4

orden =

     4

 restriccion: ingrese un numero de antenas igual al del orden del filtro ingresado)
 ingrese el numero de antenas entre 1,2,3,4,5,6,7,8,9,10,11,12,13,14 o 15....4
ingrese el voltaje1 (en v)..... voltaje1=0.15
 ingrese la voltaje2 (en V).... voltaje2=0.12
 ingrese la voltaje3 (en V).....voltaje3=0.05
 ingrese la voltaje4 (en v)..... voltaje4=-0.10
ingrese el ruido en la antena1.....ruido1=0.01
ingrese el ruido en la antena2.....ruido2=0.01
ingrese el ruido en la antena3.....ruido3=0.02
ingrese el ruido en la antena4.....ruido4=0.02
si desea que el ruido adicional vaya a la señal a la entrada digite 1
 de lo contrario se considerara que el ruido adicional se agregara a la señal de referencia
opcion_ruido=1
ingrese el error del sistema=0.0000001
si desea que la señal de entrada se use como señal de referencia digite 1
de lo contrario se considerara otra señal diferente como señal de referencia
opcion_señal=2
```

Pantalla 5.10 Potencias obtenidas con el programa problema3_cma_b

```
potencia =

    0.4969

potencia1 =

    0.0303

potencia2 =

    0.0217

potencia3 =

    0.0097

potencia4 =

    0.0114
```

Pantalla 5.11 Ejemplo problema3_cma_b. Comparación de los pesos estimados

Ejemplo problema3_cma_b
comparacion de los pesos estimados de una señal senosoidal con los pesos reales o actuales de una señal senosoidal utilizando un algoritmo cma,orden=4, numero pesos=4



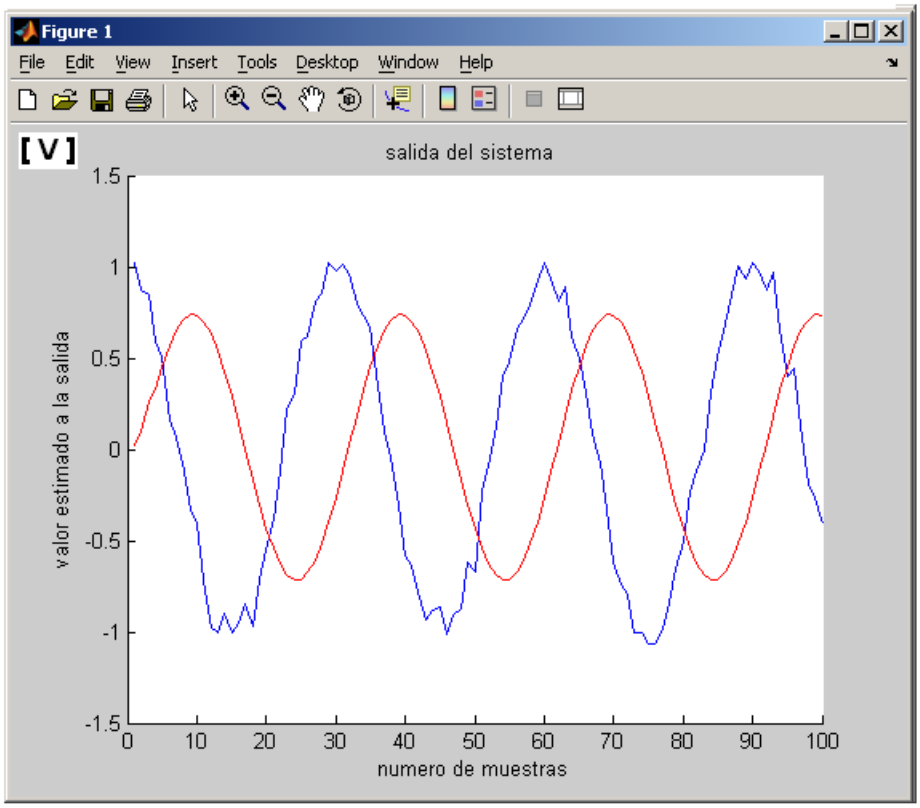
Eje de las x constante, eje de las y (amplitud de la señal en voltios). Amplitud_señal vs Valor constante

La convergencia en tres de los cuatro puntos es muy buena y hay una que tiene un error muy grande.

Las demás curvas nos salieron de la siguiente manera:

Pantalla 5.12 Ejemplo problema3_cma_b. Comparación entre la salida del sistema de una señal senosoidal (cma) con la señal de referencia del sistema. .opcion_señal=2

Ejemplo problema3_cma_b opcion señal = 2



Amplitud de la señal (voltios) vs numero de muestras
señal de referencia (color azul)
señal obtenida por el algoritmo cma (color rojo)

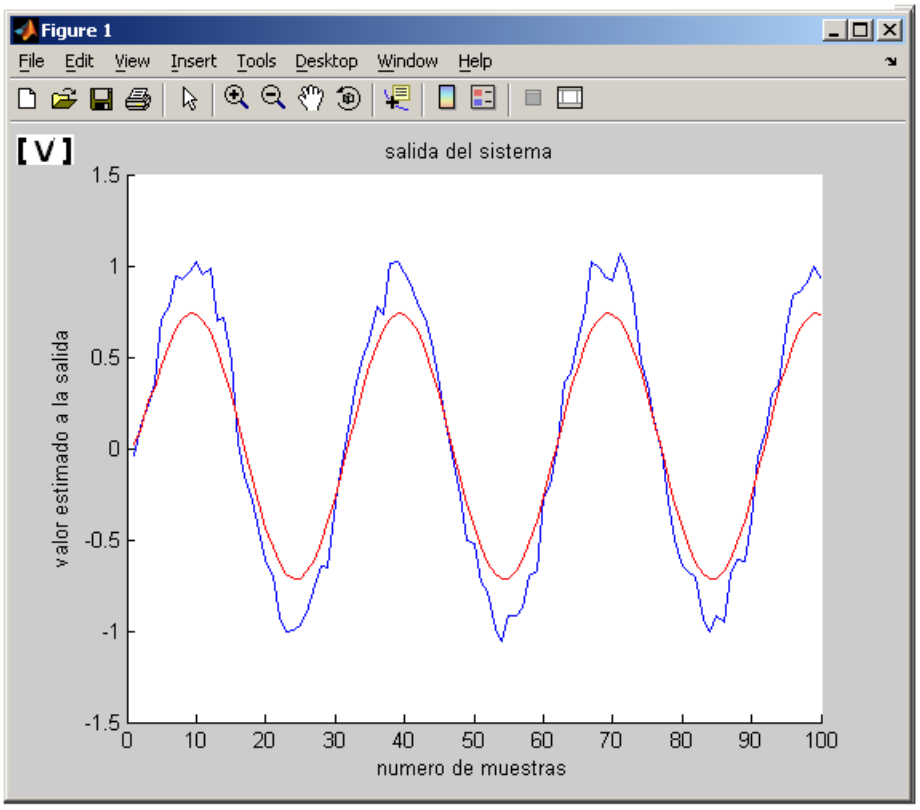
Para la opción señal = 2 la curva de referencia cma no está del todo igual a la

curva de los valores obtenidos, esto explicaría porque una de los pesos no convergió, por lo tanto es mejor trabajar con la opción 1 para el algoritmo cma-rls.

Al utilizar la opción_señal = 1 nos quedo este grafico:

Pantalla 5.13 Ejemplo problema3_cma_b. Comparación entre la salida del sistema de una señal senosoidal (cma) con la señal de referencia del sistema. .opcion_señal=1

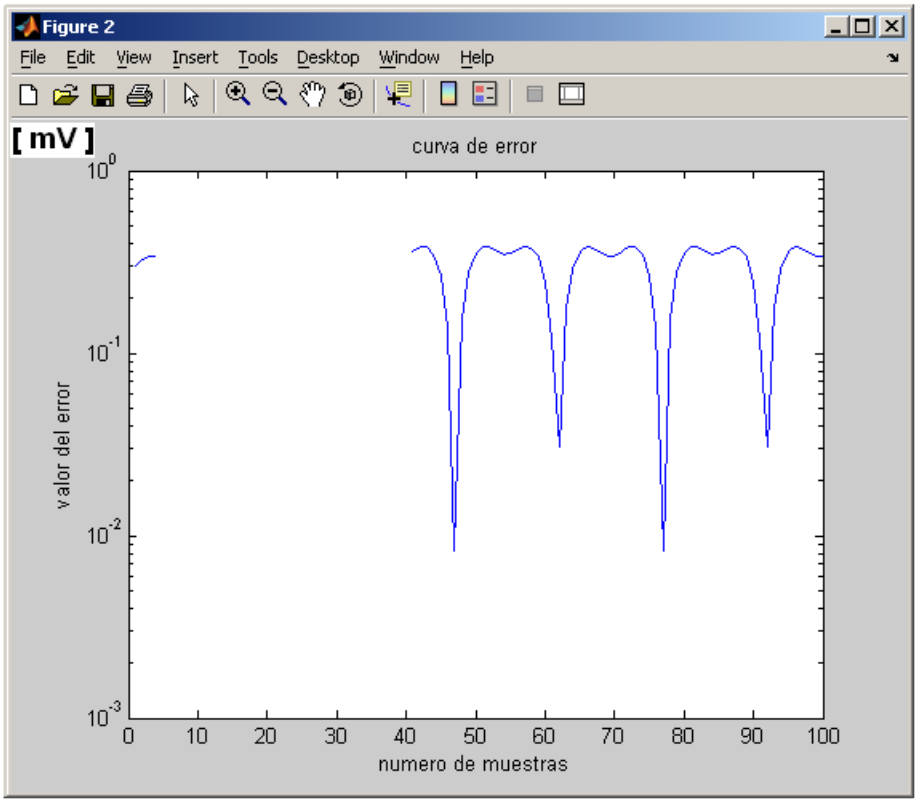
Ejemplo problema3_cma_b opcion_señal = 1



Amplitud de la señal (voltios) vs numero de muestras
señal de referencia (color azul)
señal obtenida por el algoritmo cma(color rojo)

Pantalla 5.14 Ejemplo problema3_cma_b. Curva de error de una señal senosoidal en un algoritmo cma

Ejemplo problema3_cma_b

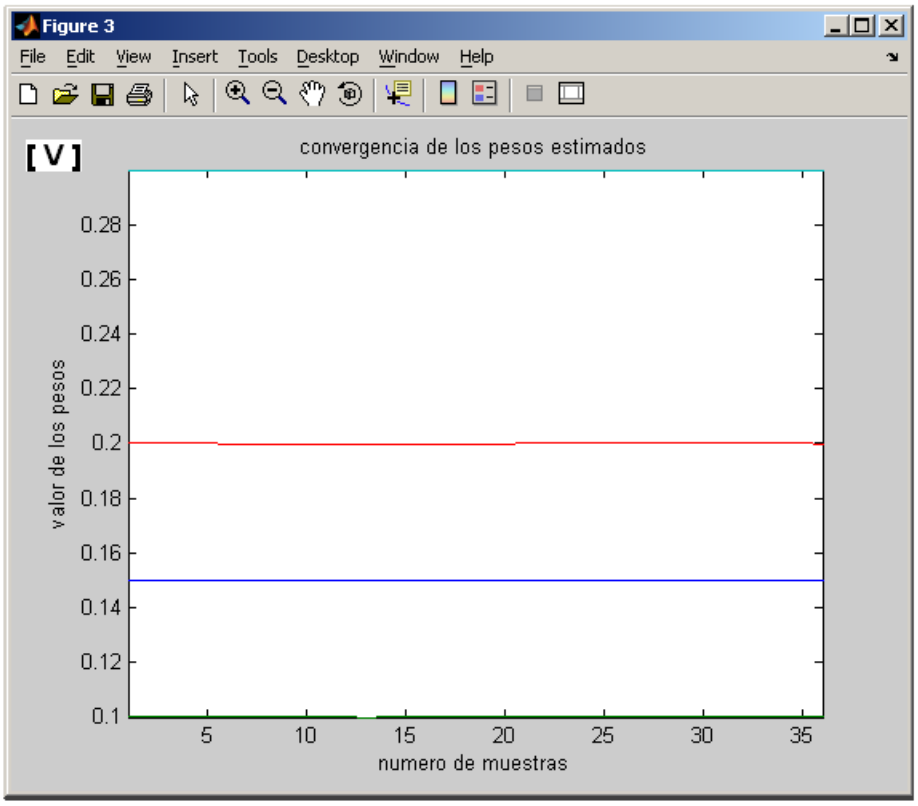


Amplitud de la señal (voltios) vs numero de muestras

Pantalla 5.15 Ejemplo problema3_cma_b. Covergencia de los pesos estimados

Ejemplo problema3_cma_b

Grafico de la convergencia de los pesos estimados en una señal senosoidal utilizando un algoritmo cma, orden=4. numero de pesos=4



Valor de los pesos vs numero de muestras

Problema2_cma_a

Pantalla 5.16 Problema2_cma_a

```
   * PROBLEMA2_CMA.- Determinar los pesos, el numero de iteraciones y la         *.
   * Potencia de un grupo de antenas inteligentes utilizando el algoritmo CMA-RLS*.
   * las antenas estan distribuidas en forma de celdas para lo cual requeriremos *.
   *  de los siguientes datos:                                                    *.
   *                                                                              *.
   *  • la señal aplicada a la entrada del filtro,   ej  Xn=sin((2*pi*k)/N)      *.
   *  • el numero de antenas  x                                                   *.
   *  • la señal de referencia, d(k)=cos((2*pi*k)/N);                             *.
   *  • el ruido de las diferentes antenas, (hay que tener presente que el        *.
   *    algoritmo CMA en su estructura internamente produce un ruido del 2,3 y    *.
   *    hasta 10 por ciento de error, aparte de este ruido puede haber otros.n(f)*.
   *  • el orden del filtro FIR cuyos coeficientes se van a ajustar, orden        *.
   *  • el número de iteraciones en el aprendizaje, N (este número se hará        *.
   *    hasta que cumpla el error)                                                *.
   *  • la constante de ajuste   u                                                *.
   *  • el vector de pesos inicial. Los voltajes de las antenas V(f)              *.
   *  • el error deseado (puede ser ingresado por teclado) error                  *.
   *                                                                              *.
   *  A su salida devolverá:                                                      *.
   *  • la salida en cada iteración durante el proceso de ajuste, y(t)            *.
   *  • la curva de aprendizaje (grafico de la curva)                             *.
   *  • los vectores de pesos obtenidos durante el proceso. W(t)                  *.
   *  • la Potencia de la antena P(k)                                             *.
   *  • el numero de iteraciones en que converge la curva y se cumple el valor    *.
   *    del error.  j                                                             *.
   * alumno: Juan Francisco Alvarez Alvarado                                      *.
   * alumno: Maribel del Rosario Chuez Gonzales                                   *.
   *                                                                              *.
   * Profesor :Ing. Pedro Vargas                                                  *.
   *                                                                              *.
   ********************************************************************************.
teclee enter para proseguir
```

Pantalla 5.17 Datos del programa problema2_cma_a

```
consideraremos un orden = 2  para la segunda etapa de un
conformador de haz, esto es para que la ecuación de
pesos que se utiliza en el programa sea la de un
algoritmo CMA, este valor será constante en la ecuación
el valor del orden del filtro para determinar los valo
res de cota y de la potencia si variaran normalmente como
si trabajaran en un algoritmo RLS

 ingrese el orden del filtro, cuyo coeficiente se va ha ajustar....5
 restriccion: ingrese un numero de antenas igual al del orden del filtro ingresado)
 ingrese el numero de antenas entre 1,2,3,4,5,6,7,8,9,10,11,12,13,14 o 15....5
ingrese el voltaje1 (en v)..... voltaje1=0.15
 ingrese la voltaje2 (en V).... voltaje2=0.18
 ingrese la voltaje3 (en V).....voltaje3=0.20
 ingrese la voltaje4 (en v)..... voltaje4=0.12
 ingrese la voltaje5 (en v)..... voltaje5=0.14
ingrese el ruido en la antena1.....ruido1=0.01
ingrese el ruido en la antena2.....ruido2=0.01
ingrese el ruido en la antena3.....ruido3=0.01
ingrese el ruido en la antena4.....ruido4=0.02
ingrese el ruido en la antena5.....ruido5=0.02
si desea que el ruido adicional vaya a la señal a la entrada digite 1
 de lo contrario se considerara que el ruido adicional se agregara a la señal de referencia
opcion_ruido=1
ingrese el error del sistema=0.0000001
```

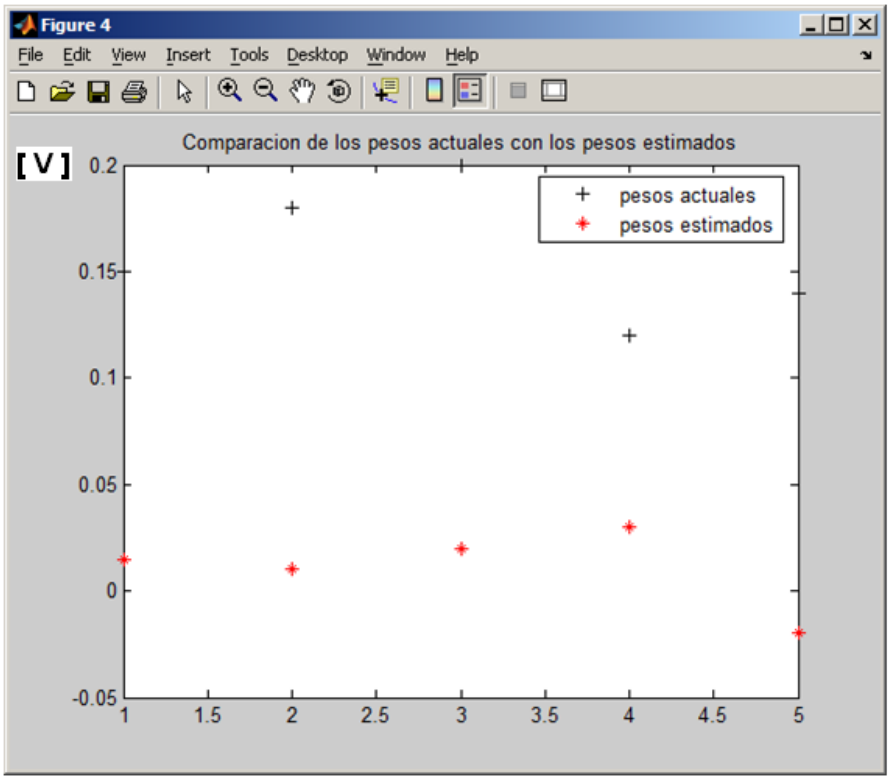
Pantalla 5.18 Potencias obtenidas con el programa problema2_cma_a

```
potencia =

    0.3156

potencia1 =

    0.0291

potencia2 =

    0.0396

potencia3 =

    0.0477

potencia4 =

    0.0230

Potencia5 =

    0.0291
```

El segundo y cuarto pesos tienen un error mayor al esperado.

Pantalla 5.19 Ejemplo problema2_cma_a. Comparación de los pesos estimados

Ejemplo problema2_cma_a
comparacion de los pesos estimados de una señal gaussiana
con los pesos reales o actuales de una señal gaussiana
utilizando un algoritmo cma,orden=5, numero pesos=5

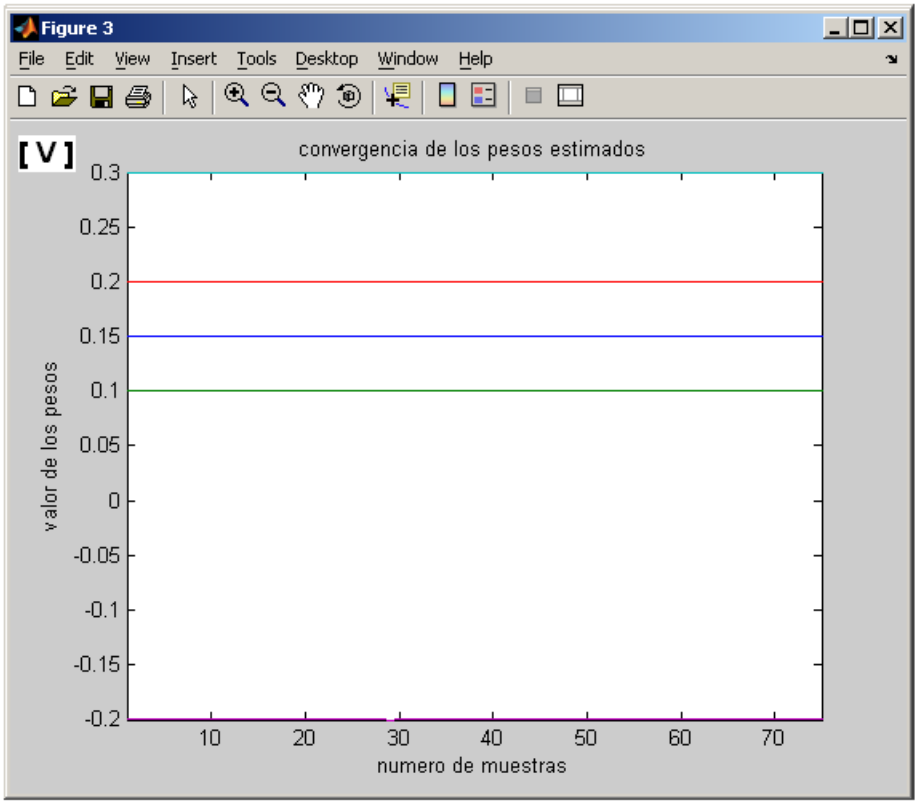


Eje de las x constante, eje de las y (amplitud de la señal en voltios). Amplitud_señal vs Valor constante

Pantalla 5.20 Ejemplo problema2_cma_a. Convergencia de los pesos estimados

Ejemplo problema2_cma_a

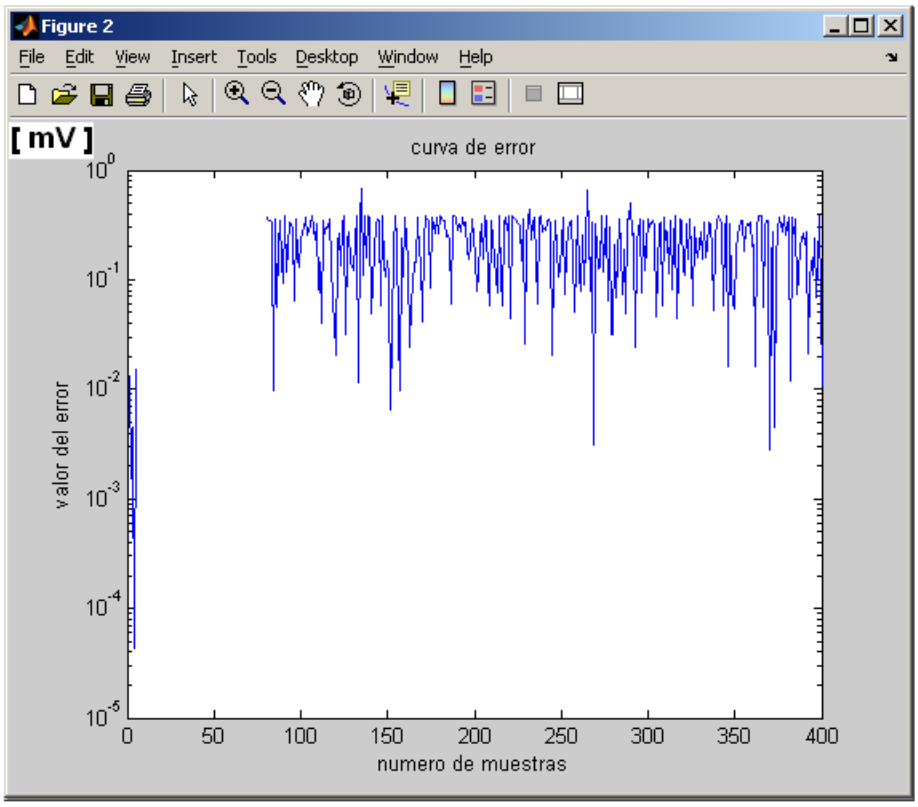
Grafico de la convergencia de los pesos estimados en una señal gaussiana utilizando un algoritmo cma, orden =5, numero de pesos =5.



Valor de los pesos vs numero de muestras

Pantalla 5.21 Ejemplo problema2_cma_a. Curva de error de una señal gaussiana en un algoritmo cma

Ejemplo problema2_cma_a

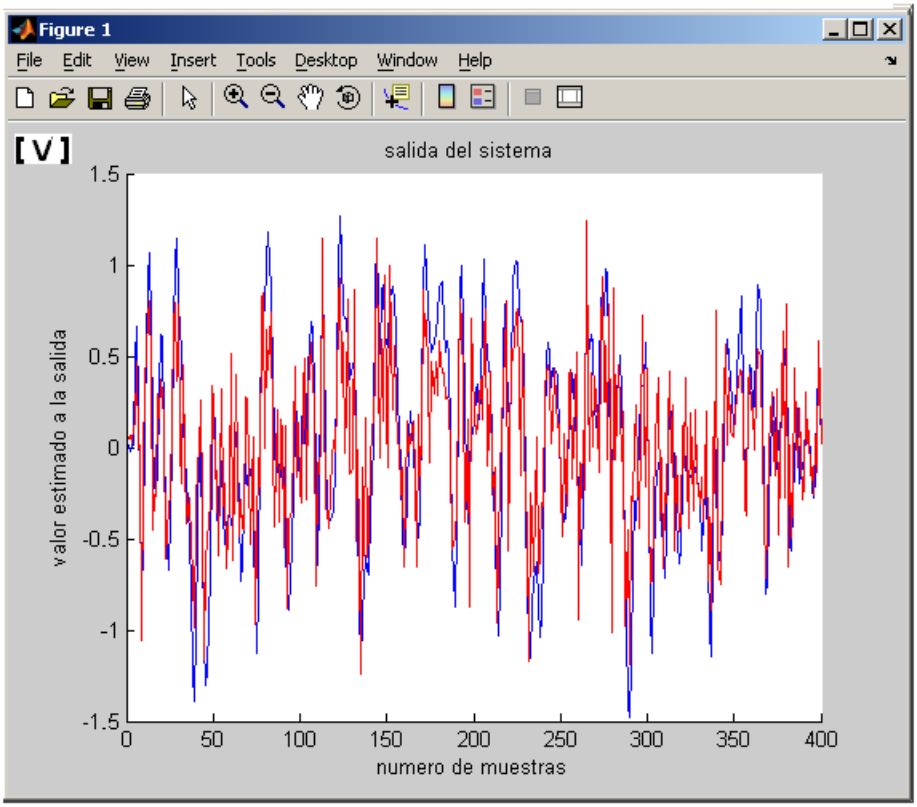


Amplitud de la señal (voltios) vs numero de muestras

La grafica del error sigue sin aparecer en la zona de seguridad de N=80

Pantalla 5.22 Ejemplo problema2_cma_a. Comparación entre la salida del sistema de una señal gaussiana (cma) con la señal de referencia del sistema. .opcion_ruido=1

Ejemplo problema2_cma_a, opcion_ruido = 1



Amplitud de la señal (voltios) vs numero de muestras
señal de referencia (color azul)
señal obtenida por el algoritmo cma(color rojo)

Problema1_cma_a

Pantalla 5.23 Problema1_cma_a

```
   * PROBLEMA1_cma.- Determinar los pesos, el numero de iteraciones y la potencia*
   * de un grupo de antenas inteligentes utilizando el algoritmo CMA-RLS         *
   * las antenas estan distribuidas en forma de celdas para lo cual requeriremos *
   * de los siguientes datos:                                                     *
   *                                                                              *
   *  • la señal aplicada a la entrada del filtro,   ej  Xn=sin((2*pi*k)/N)    *.
   *  • el numero de antenas  x                                                   *
   *  • la señal de referencia, d(k)=cos((2*pi*k)/N);                           *.
   *  • el ruido de las diferentes antenas, (hay que tener presente que el       *
   *    algoritmo CMA en su estructura internamente produce un ruido del 2, 3    *
   *    y hasta un 10 por ciento, aparte de este ruido puede haber otros.  n(f)  *
   *  • el orden del filtro FIR cuyos coeficientes se van a ajustar, orden       *
   *  • el número de iteraciones en el aprendizaje, N (este número se hará       *
   *    hasta que cumpla el error)                                               *
   *  • la constante de ajuste   u                                               *
   *  • el vector de pesos inicial. Los voltajes de las antenas V(f)             *
   *  • el error deseado (puede ser ingresado por teclado) error                 *
   *                                                                             *
   *  A su salida devolverá:                                                     *
   *  • la salida en cada iteración durante el proceso de ajuste, y(t)           *
   *  • la curva de aprendizaje (grafico de la curva)                            *
   *  • los vectores de pesos obtenidos durante el proceso. W(t)                 *
   *  • la Potencia de la antena P(k)                                            *
   *  • el numero de iteraciones en que converge la curva y se cumple el valor   *
   *    del error.  j                                                            *
   * alumno: Juan Francisco Alvarez Alvarado                                     *
   * alumno: Maribel del Rosario Chuez Gonzales                                  *
   *                                                                             *
   * Profesor :Ing. Pedro Vargas                                                 *
   *                                                                             *
   *******************************************************************************
teclee enter para proseguir
```

Pantalla 5.24 Datos del programa problema1_cma_a

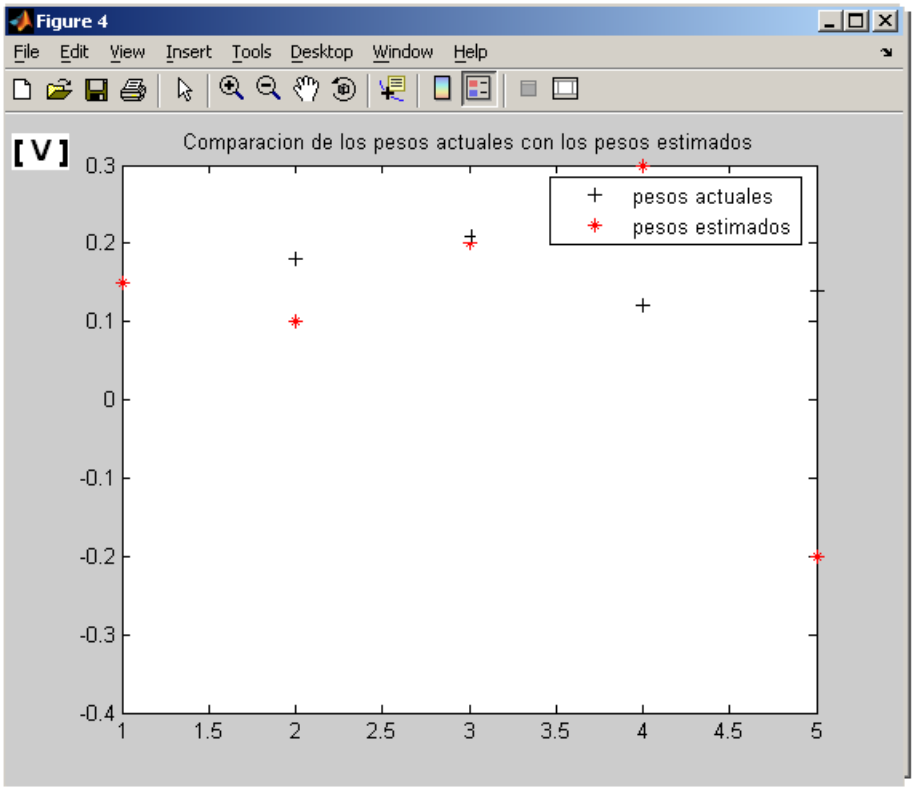
```
 ingrese el orden del filtro, cuyo coeficiente se va ha ajustar....5

orden =

     5

 restricción: ingrese un número de antenas igual al del orden del filtro ingresado
 ingrese el numero de antenas entre 1,2,3,4,5,6,7,8,9,10,11,12,13,14 o 15....5
ingrese el voltaje1 (en v)..... voltaje1=0.15
 ingrese la voltaje2 (en V).... voltaje2=0.18
 ingrese la voltaje3 (en V).....voltaje3=0.20
 ingrese la voltaje4 (en v)..... voltaje4=0.12
 ingrese la voltaje5 (en v)..... voltaje5=0.14
ingrese el ruido en la antena1.....ruido1=0.01
ingrese el ruido en la antena2.....ruido2=0.01
ingrese el ruido en la antena3.....ruido3=0.01
ingrese el ruido en la antena4.....ruido4=0.02
ingrese el ruido en la antena5.....ruido5=0.02
si desea que el ruido adicional vaya a la señal a la entrada digite 1
 de lo contrario se considerara que el ruido adicional se agregara a la señal de referencia
opcion_ruido=1
ingrese el error del sistema=0.0000001

h =

    0.1500
    0.1800
    0.2000
    0.1200
    0.1400
```

Pantalla 5.25 Potencias obtenidas con el programa problema1_cma_a

```
potencia =

    0.4879

potencia1 =

    0.0315

potencia2 =

    0.0422

potencia3 =

    0.0503

potencia4 =

    0.0254

Potencia5 =

    0.0315
```

Pantalla 5.26 Ejemplo problema1_cma_a. Comparación de los pesos estimados

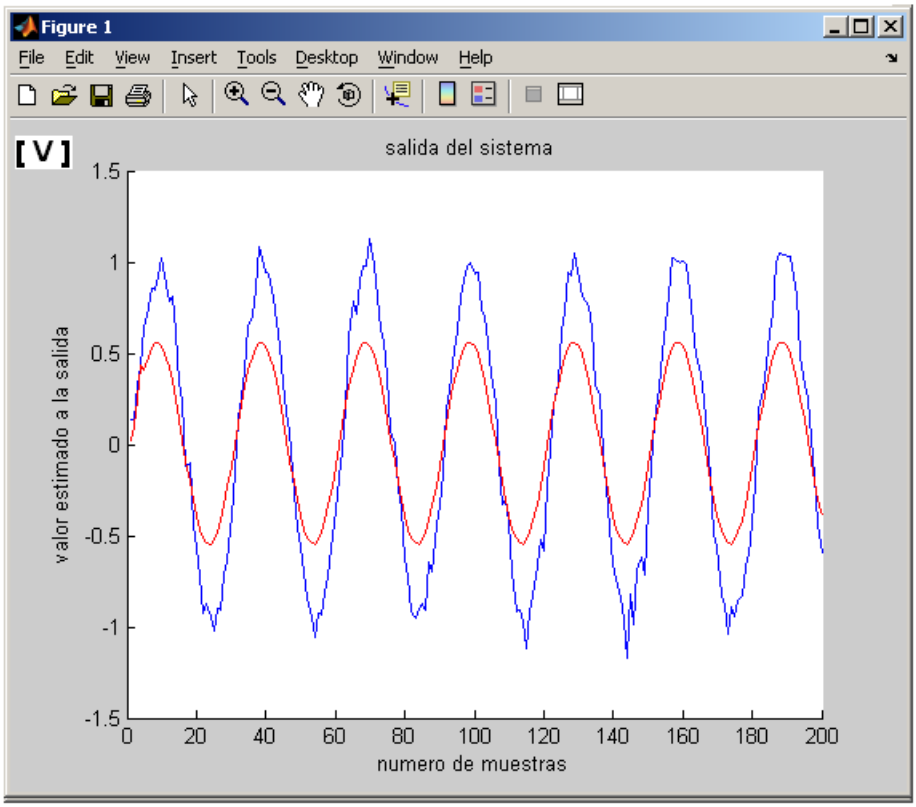
Ejemplo problema1_cma_a
comparacion de los pesos estimados de una señal senosoidal con los pesos reales o actuales de una señal senosoidal utilizando un algoritmo cma,orden=5, numero pesos=5



Eje de las x constante, eje de las y (amplitud de la señal en voltios). Amplitud_señal vs Valor constante

Pantalla 5.27 Ejemplo problema1_cma_a. Comparación entre la salida del sistema de una señal senosoidal (cma) con la señal de referencia del sistema.

Ejemplo problema1_cma_a

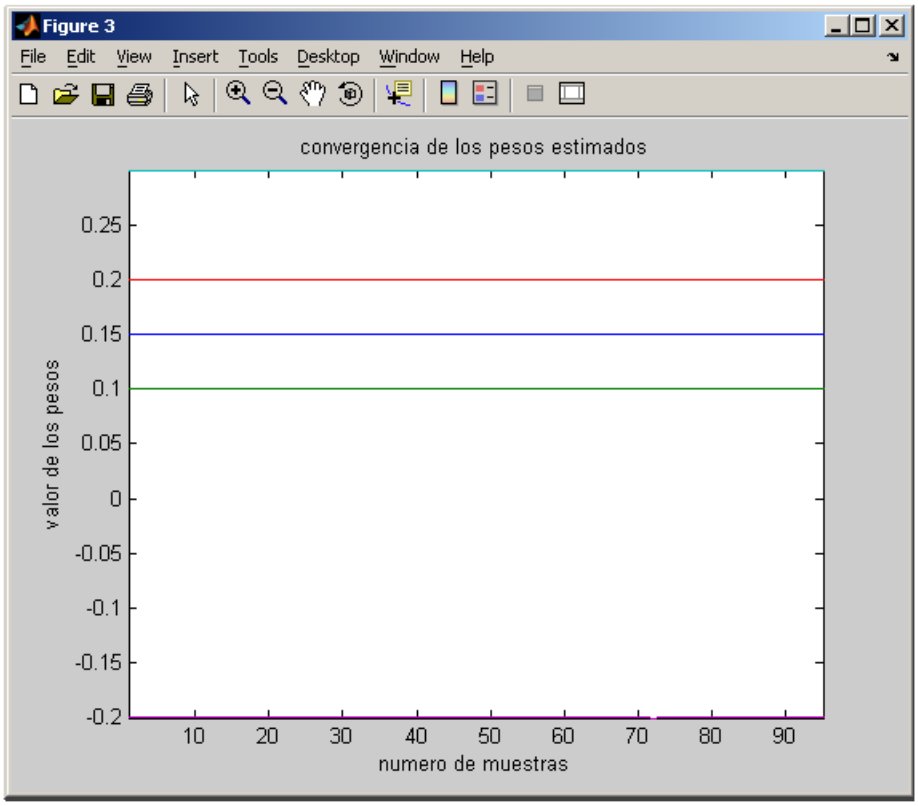


Amplitud de la señal (voltios) vs numero de muestras
señal de referencia (color azul)
señal obtenida por el algoritmo cma (color rojo)

Pantalla 5.28 Ejemplo problema1_cma_a. Convergencia de los pesos estimados

Ejemplo problema1_cma_a

Grafico de la convergencia de los pesos estimados en una señal senosoidal utilizando un algoritmo cma,orden=5, numero de pesos=5



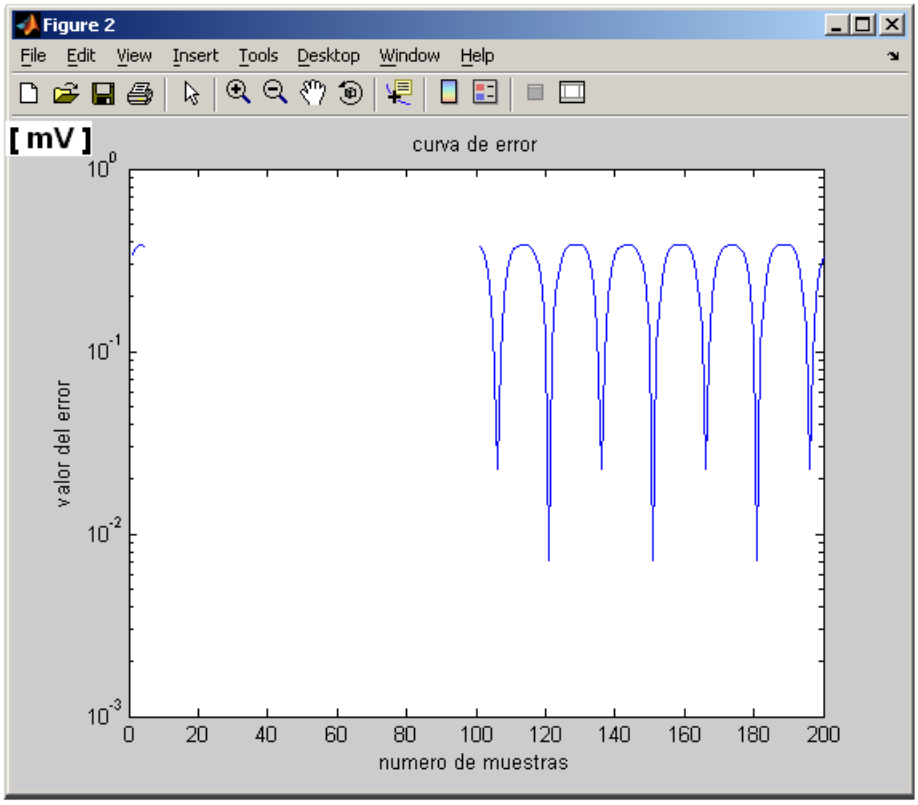
Valor de los pesos vs numero de muestras

Según este grafico la convergencia es casi instantánea, pero sabemos por teoría que el algoritmo cma no siempre converge, por lo que este grafico puede guiarnos a un error, este grafico de convergencias es muy útil para los algoritmos lms y rls, pero para el algoritmo cma no lo es, porque este

algoritmo no siempre converge en la realidad.

Pantalla 5.29 Ejemplo problema1_cma_a. Curva de error de una señal senosoidal en un algoritmo cma

Ejemplo problema1_cma_a



Amplitud de la señal (voltios) vs numero de muestras

El programa no puede graficar la curva del error dentro del rango de seguridad de N=80, debe ser una propiedad del algoritmo cma. Los valores obtenidos tienden a girar alrededor de los valores ingresados como pesos, en

la teoría del algoritmo cma se recomienda en ese caso buscar otros valores como pesos iniciales o considerar otro tipo de algoritmos para evaluar estos valores.

Problema3_dmi_a

Pantalla 6.1 Problema3_dmi_a

```
* PROBLEMA3_dmi.- Determinar los pesos, el numero de iteraciones y la potencia*.
* de un grupo de antenas inteligentes utilizando el algoritmo DMI (algoritmo  *.
* adaptativo de matriz de inversion directa, las antenas estan distribuidas en*.
* forma de celdas para lo cual requeriremos los siguientes datos:             *.
*                                                                             *.
*  • la señal aplicada a la entrada del filtro,   ej  Xn=sin((2*pi*k)/N)      *.
*  • el numero de antenas  x                                                  *.
*  • la señal de referencia, d(k)=cos((2*pi*k)/N);                            *.
*  • el ruido de las diferentes antenas, (hay que tener presente que el       *.
*    algoritmo DMI en su estructura internamente produce un ruido del 2,3 y   *.
*    hasta 10 por ciento de error aparte de este ruido puede haber otros.n(f) *.
*  • el orden del filtro FIR cuyos coeficientes se van a ajustar, orden       *.
*  • el número de iteraciones en el aprendizaje, N (este número se hará       *.
*    hasta que cumpla el error)                                               *.
*  • la constante de ajuste   u                                               *.
*  • el vector de pesos inicial. Los voltajes de las antenas V(f)             *.
*  • el error deseado (puede ser ingresado por teclado) error                 *.
*                                                                             *.
*  A su salida devolverá:                                                     *.
*  • la salida en cada iteración durante el proceso de ajuste, y(t)           *.
*  • la curva de aprendizaje (grafico de la curva)                            *.
*  • los vectores de pesos obtenidos durante el proceso. W(t)                 *.
*  • la Potencia de la antena P(k)                                            *.
*  • el numero de iteraciones en que converge la curva y se cumple el valor   *.
*    del error.  j                                                            *.
* alumno: Juan Francisco Alvarez Alvarado                                     *.
* alumno: Maribel del Rosario Chuez Gonzales                                  *.
*                                                                             *.
* Profesor :Ing. Pedro Vargas                                                 *.
*                                                                             *.
*******************************************************************************.
teclee enter para proseguir
```

. Para 400 muestras, señal senosoidal opcion1, N =80 hallar pesos y N=200 zona de seguridad de la curva

Pantalla 6.2 Datos del programa problema3_dmi_a

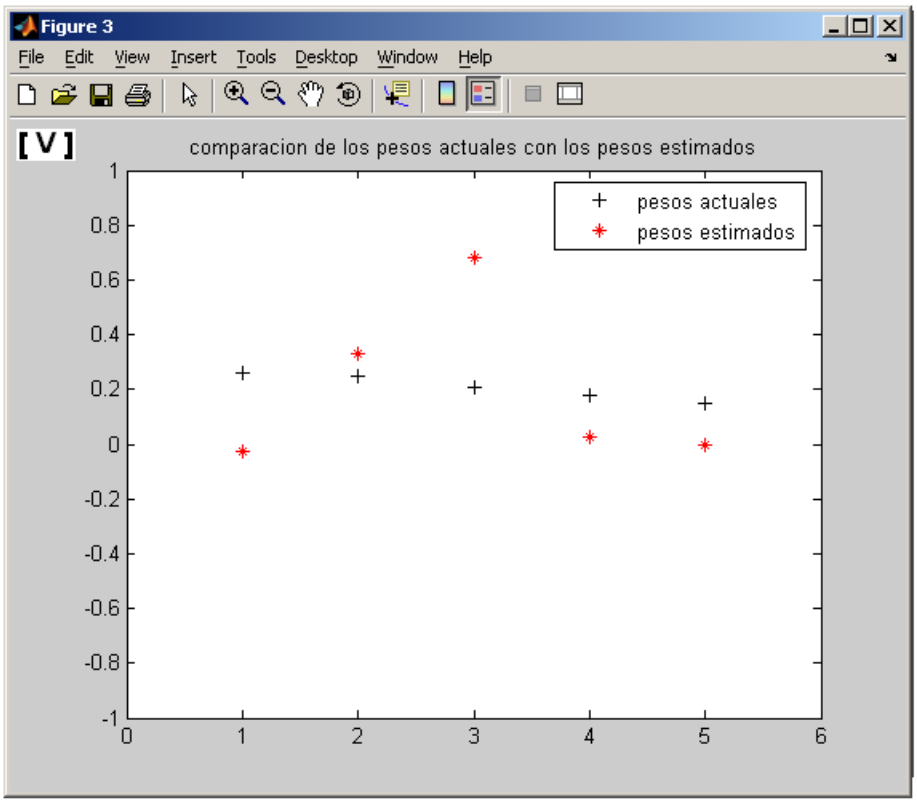
```
 ingrese el orden del filtro, cuyo coeficiente se va ha ajustar....5

orden =

     5

 ingrese el numero de antenas entre 1 y 5
 ingrese el numero de antenas entre 1,2,3,4 o 5....5
ingrese el voltaje1 (en v)..... voltaje1=0.26
 ingrese la voltaje2 (en V).... voltaje2=0.25
 ingrese la voltaje3 (en V).....voltaje3=0.21
 ingrese la voltaje4 (en v)..... voltaje4=0.18
 ingrese la voltaje5 (en v)..... voltaje5=0.15
ingrese el ruido en la antena1.....ruido1=0.01
ingrese el ruido en la antena2.....ruido2=0.01
ingrese el ruido en la antena3.....ruido3=0.01
ingrese el ruido en la antena4.....ruido4=0.02
ingrese el ruido en la antena5.....ruido5=0.02
si desea que el ruido adicional vaya a la señal a la entrada digite 1
 de lo contrario se considerara que el ruido adicional se agregara a la señal de referencia
opcion_ruido=1
ingrese el error del sistema=0.0000001
si desea que la señal de entrada se use como señal de referencia digite 1
 de lo contrario se considerara otra señal diferente como señal de referencia
opcion_señal=1
```

Pantalla 6.3 Potencias obtenidas con el programa problema3_dmi_a

```
potencia =

    0.5070

potencia1 =

    0.0792

potencia2 =

    0.0738

potencia3 =

    0.0544

potencia4 =

    0.0459

potencia5 =

    0.0347
```

Pantalla 6.4 Ejemplo problema3_dmi_a. Comparación de los pesos estimados

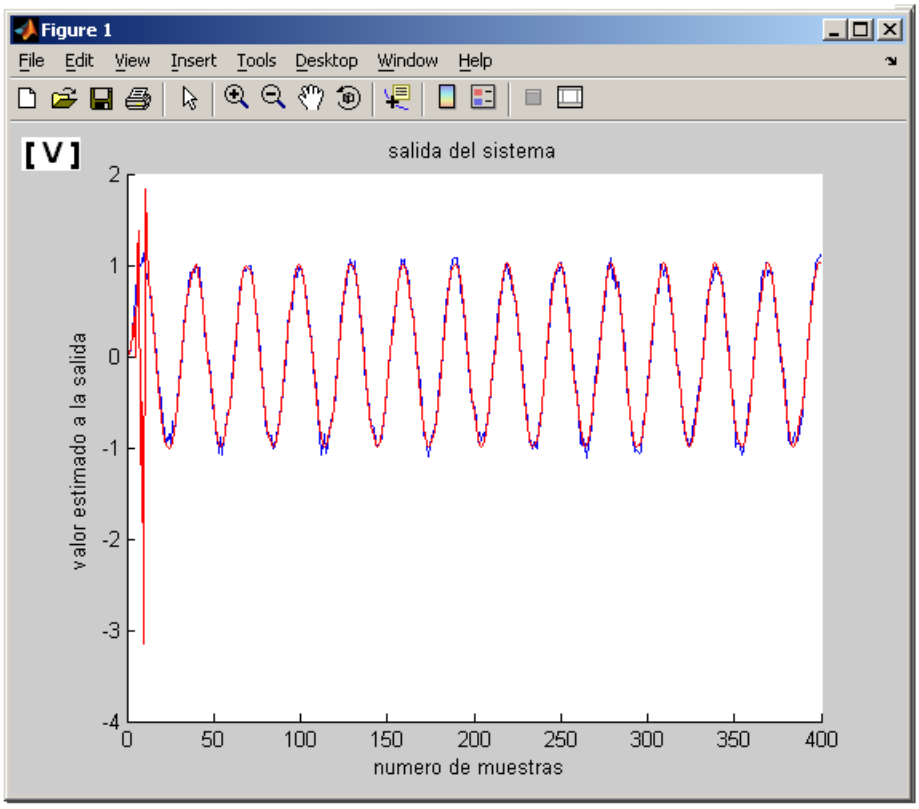
Ejemplo problema3_dmi_a
comparacion de los pesos estimados de una señal senosoidal con los pesos reales o actuales de una señal senosoidal utilizando un algoritmo dmi,orden=5, numero pesos=5



Eje de las x constante, eje de las y (amplitud de la señal en voltios). Amplitud_señal vs Valor constante

Pantalla 6.5 Ejemplo problema3_dmi_a. Comparación entre la salida del sistema de una señal senosoidal (dmi) con la señal de referencia del sistema

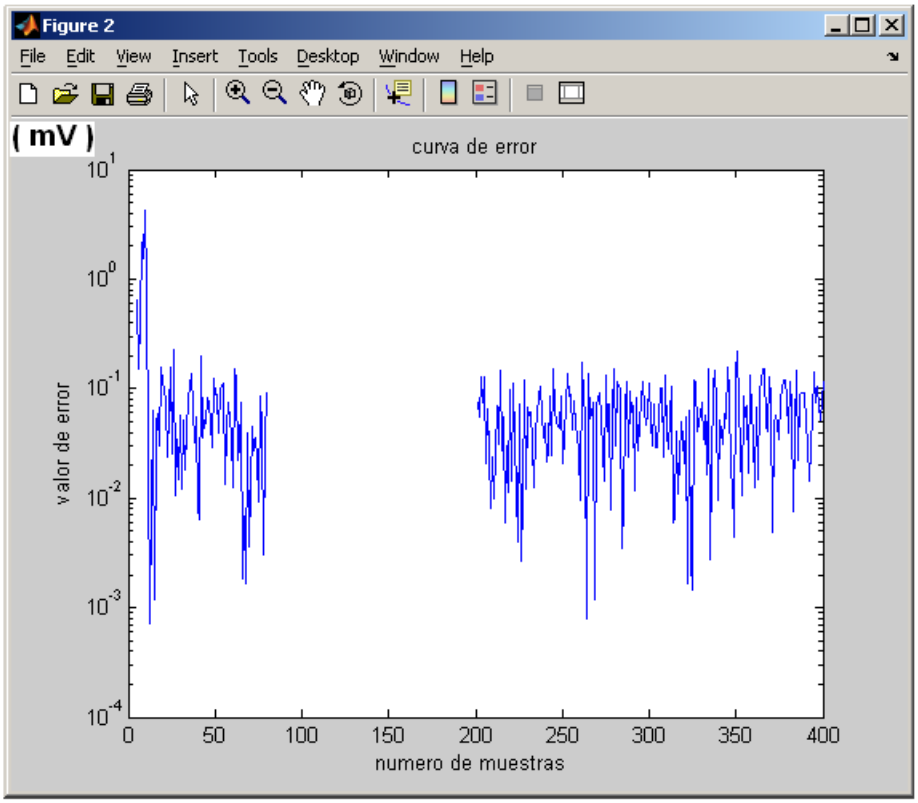
Ejemplo problema3_ dmi_a



Amplitud de la señal (voltios) vs numero de muestras
señal de referencia (color azul)
señal obtenida por el algoritmo dmi (color rojo)

Pantalla 6.6 Ejemplo problema3_dmi_a. Curva de error de una señal senosoidal en un algoritmo dmi

Ejemplo problema3_ dmi_a



Amplitud de la señal (voltios) vs numero de muestras

Problema3_dmi_b

Pantalla 6.7 Problema3_dmi_b

```
   * PROBLEMA3_dmi.- Determinar los pesos, el numero de iteraciones y la potencia*.
   * de un grupo de antenas inteligentes utilizando el algoritmo DMI (algoritmo  *.
   * adaptativo de matriz de inversion directa, las antenas estan distribuidas en*.
   * forma de celdas para lo cual requeriremos los siguientes datos:              *.
   *                                                                              *.
   *  • la señal aplicada a la entrada del filtro,   ej  Xn=sin((2*pi*k)/N)       *.
   *  • el numero de antenas  x                                                   *.
   *  • la señal de referencia, d(k)=cos((2*pi*k)/N);                             *.
   *  • el ruido de las diferentes antenas, (hay que tener presente que el        *.
   *    algoritmo DMI en su estructura internamente produce un ruido del 2,3 y    *.
   *    hasta 10 por ciento de error aparte de este ruido puede haber otros.n(f)  *.
   *  • el orden del filtro FIR cuyos coeficientes se van a ajustar, orden        *.
   *  • el número de iteraciones en el aprendizaje, N (este número se hará        *.
   *    hasta que cumpla el error)                                                *.
   *  • la constante de ajuste   u                                                *.
   *  • el vector de pesos inicial. Los voltajes de las antenas V(f)              *.
   *  • el error deseado (puede ser ingresado por teclado) error                  *.
   *                                                                              *.
   *  A su salida devolverá:                                                      *.
   *  • la salida en cada iteración durante el proceso de ajuste, y(t)            *.
   *  • la curva de aprendizaje (grafico de la curva)                             *.
   *  • los vectores de pesos obtenidos durante el proceso. W(t)                  *.
   *  • la Potencia de la antena P(k)                                             *.
   *  • el numero de iteraciones en que converge la curva y se cumple el valor    *.
   *    del error.  j                                                             *.
   * alumno: Juan Francisco Alvarez Alvarado                                      *.
   * alumno: Maribel del Rosario Chuez Gonzales                                   *.
   *                                                                              *.
   * Profesor :Ing. Pedro Vargas                                                  *.
   *                                                                              *.
   ********************************************************************************.
teclee enter para proseguir
```

Función senosoidal, para 400 muestras, N =80 para la evaluación de pesos, N= 200 para la zona de control que evalua la curva, señal de referencia opción señal=1, la misma señal de entrada.

Pantalla 6.8 Datos del programa problema3_dmi_b

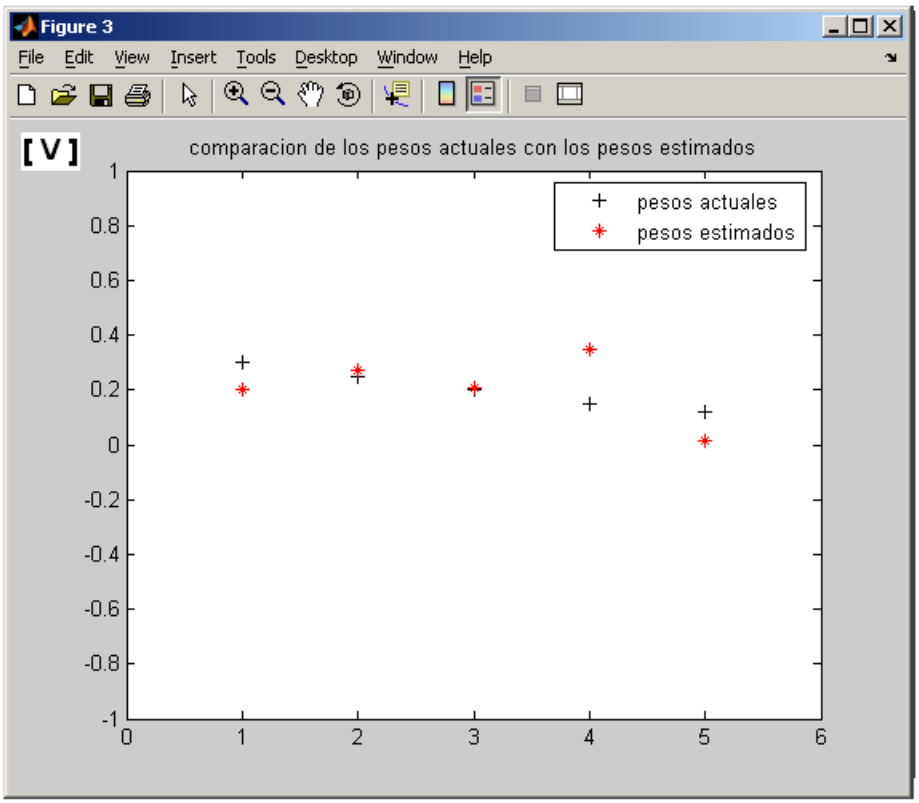
```
 ingrese el orden del filtro, cuyo coeficiente se va ha ajustar....5

orden =

     5

 ingrese el numero de antenas entre 1 y 5
 ingrese el numero de antenas entre 1,2,3,4 o 5....5
ingrese el voltaje1 (en v)..... voltaje1=0.30
 ingrese la voltaje2 (en V).... voltaje2=0.25
 ingrese la voltaje3 (en V).....voltaje3=0.20
 ingrese la voltaje4 (en v)..... voltaje4=0.15
 ingrese la voltaje5 (en v)..... voltaje5=0.12
ingrese el ruido en la antena1.....ruido1=0.01
ingrese el ruido en la antena2.....ruido2=0.01
ingrese el ruido en la antena3.....ruido3=0.01
ingrese el ruido en la antena4.....ruido4=0.02
ingrese el ruido en la antena5.....ruido5=0.02
si desea que el ruido adicional vaya a la señal a la entrada digite 1
 de lo contrario se considerara que el ruido adicional se agregara a la señal de referencia
opcion_ruido=1
ingrese el error del sistema=0.0000001
si desea que la señal de entrada se use como señal de referencia digite 1
 de lo contrario se considerara otra señal diferente como señal de referencia
opcion_señal=1
```

Pantalla 6.9 Potencias obtenidas con el programa problema3_dmi_b

```
potencia =

    0.5040

potencia1 =

    0.1019

potencia2 =

    0.0733

potencia3 =

    0.0496

potencia4 =

    0.0343

potencia5 =

    0.0250
```

Pantalla 6.10 Ejemplo problema3_dmi_b. Comparación de los pesos estimados

Ejemplo problema3_dmi_b
comparacion de los pesos estimados de una señal senosoidal con los pesos reales o actuales de una señal senosoidal utilizando un algoritmo dmi,orden=5, numero pesos=5

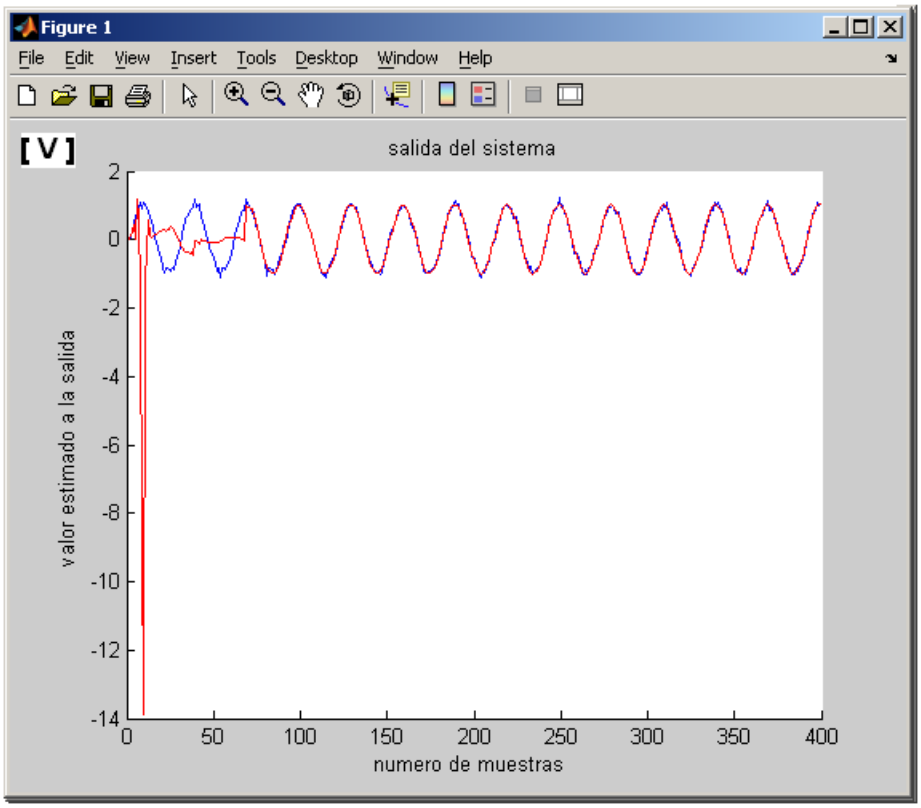


Eje de las x constante, eje de las y (amplitud de la señal en voltios). Amplitud_señal vs Valor constante

Notece como hay un pico causado por el programa dmi en la zona del lazo de evaluación de pesos.

Pantalla 6.11 Ejemplo problema3_dmi_b. Comparación entre la salida del sistema de una señal senosoidal (dmi) con la señal de referencia del sistema

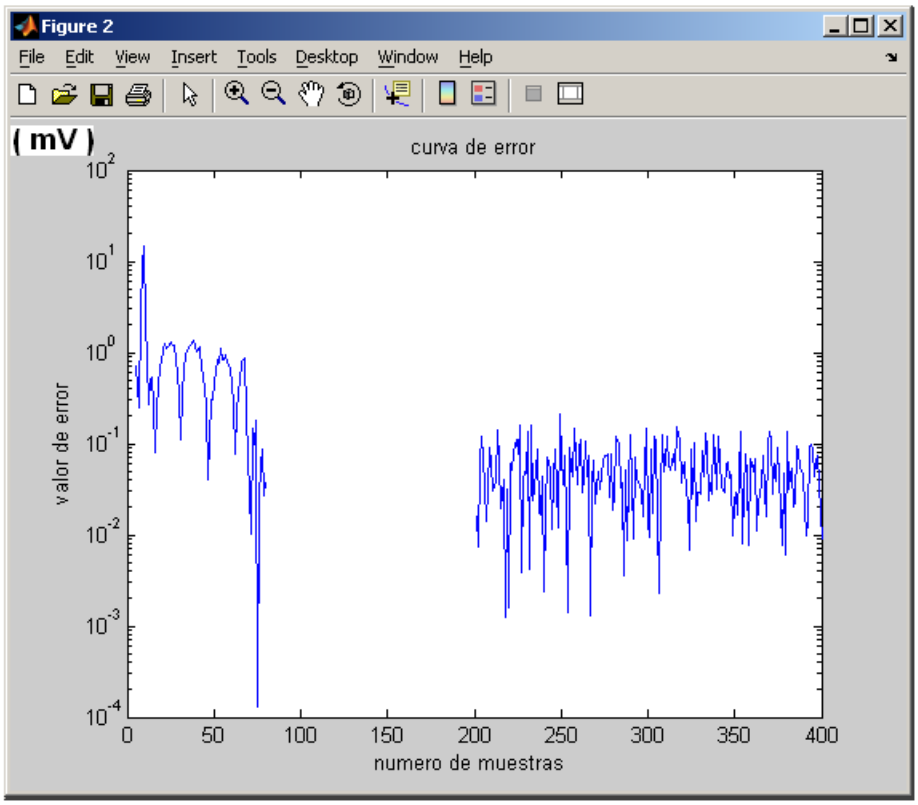
Ejemplo problema3_ dmi_b



Amplitud de la señal (voltios) vs numero de muestras
señal de referencia (color azul)
señal obtenida por el algoritmo dmi (color rojo)

Pantalla 6.12 Ejemplo problema3_dmi_b. Curva de error de una señal senosoidal en un algoritmo dmi

Ejemplo problema3_dmi_b



Amplitud de la señal (voltios) vs numero de muestras

Problema3_dmi_c

Pantalla 6.13 Problema3_dmi_c

```
   * PROBLEMA3_dmi.- Determinar los pesos, el numero de iteraciones y la potencia*.
   * de un grupo de antenas inteligentes utilizando el algoritmo DMI (algoritmo  *.
   * adaptativo de matriz de inversion directa, las antenas estan distribuidas en*.
   * forma de celdas para lo cual requeriremos los siguientes datos:              *.
   *                                                                              *.
   *  • la señal aplicada a la entrada del filtro,   ej  Xn=sin((2*pi*k)/N)       *.
   *  • el numero de antenas  x                                                   *.
   *  • la señal de referencia, d(k)=cos((2*pi*k)/N);                             *.
   *  • el ruido de las diferentes antenas, (hay que tener presente que el        *.
   *    algoritmo DMI en su estructura internamente produce un ruido del 2,3 y    *.
   *    hasta 10 por ciento de error aparte de este ruido puede haber otros.n(f)  *.
   *  • el orden del filtro FIR cuyos coeficientes se van a ajustar, orden        *.
   *  • el número de iteraciones en el aprendizaje, N (este número se hará        *.
   *    hasta que cumpla el error)                                                *.
   *  • la constante de ajuste   u                                                *.
   *  • el vector de pesos inicial. Los voltajes de las antenas V(f)              *.
   *  • el error deseado (puede ser ingresado por teclado) error                  *.
   *                                                                              *.
   *  A su salida devolverá:                                                      *.
   *  • la salida en cada iteración durante el proceso de ajuste, y(t)            *.
   *  • la curva de aprendizaje (grafico de la curva)                             *.
   *  • los vectores de pesos obtenidos durante el proceso. W(t)                  *.
   *  • la Potencia de la antena P(k)                                             *.
   *  • el numero de iteraciones en que converge la curva y se cumple el valor    *.
   *    del error.  j                                                             *.
   * alumno: Juan Francisco Alvarez Alvarado                                      *.
   * alumno: Maribel del Rosario Chuez Gonzales                                   *.
   *                                                                              *.
   * Profesor :Ing. Pedro Vargas                                                  *.
   *                                                                              *.
   ********************************************************************************.
teclee enter para proseguir
```

Pantalla 6.14 Datos del programa problema3_dmi_c

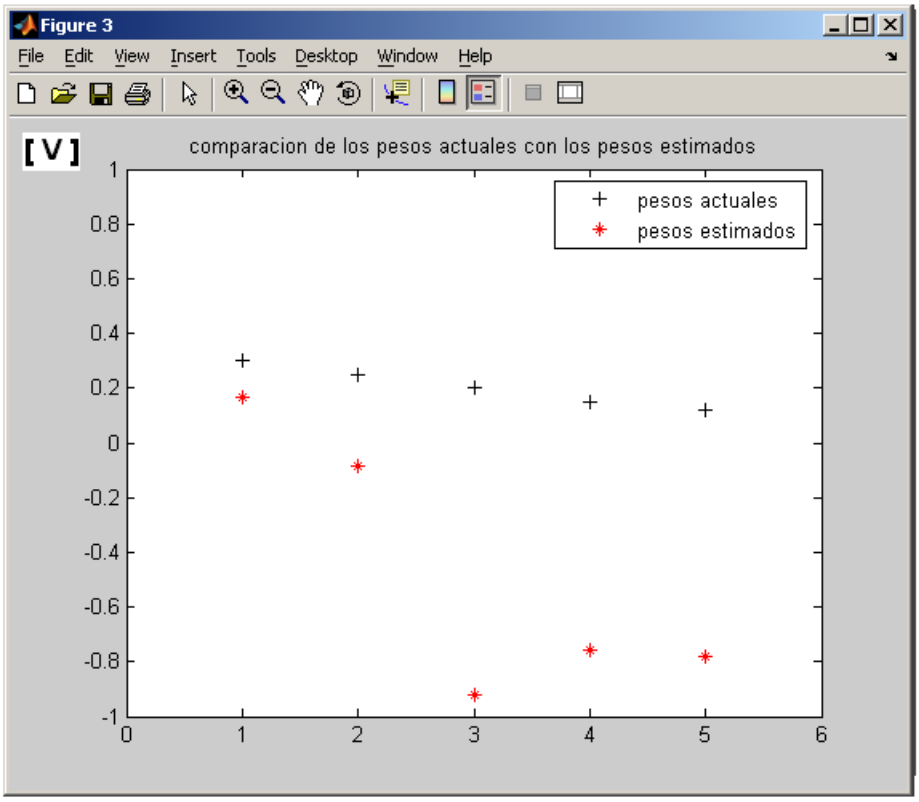
```
 ingrese el orden del filtro, cuyo coeficiente se va ha ajustar....5

orden =

     5

 ingrese el numero de antenas entre 1 y 5
 ingrese el numero de antenas entre 1,2,3,4 o 5....5
ingrese el voltaje1 (en v)..... voltaje1=0.30
 ingrese la voltaje2 (en V).... voltaje2=0.25
 ingrese la voltaje3 (en V).....voltaje3=0.20
 ingrese la voltaje4 (en v)..... voltaje4=0.15
 ingrese la voltaje5 (en v)..... voltaje5=0.12
ingrese el ruido en la antena1.....ruido1=0.01
ingrese el ruido en la antena2.....ruido2=0.01
ingrese el ruido en la antena3.....ruido3=0.01
ingrese el ruido en la antena4.....ruido4=0.02
ingrese el ruido en la antena5.....ruido5=0.02
si desea que el ruido adicional vaya a la señal a la entrada digite 1
 de lo contrario se considerara que el ruido adicional se agregara a la señal de referencia
opcion_ruido=1
ingrese el error del sistema=0.0000001
si desea que la señal de entrada se use como señal de referencia digite 1
 de lo contrario se considerara otra señal diferente como señal de referencia
opcion_señal=2
```

Pantalla 6.15 Potencias obtenidas con el programa problema3_dmi_c

```
potencia =

    0.4983

potencia1 =

    0.0991

potencia2 =

    0.0710

potencia3 =

    0.0478

potencia4 =

    0.0328

potencia5 =

    0.0237
```

Pantalla 6.16 Ejemplo problema3_dmi_c. Comparación de los pesos estimados

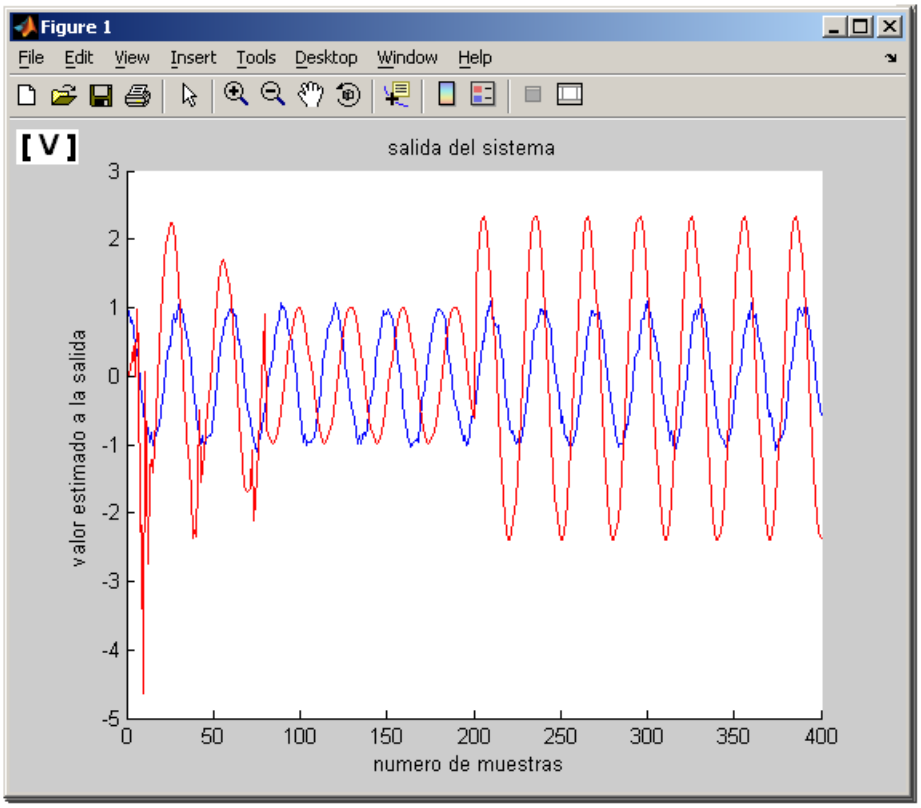
Ejemplo problema3_dmi_c
comparacion de los pesos estimados de una señal senosoidal con los pesos reales o actuales de una señal senosoidal utilizando un algoritmo dmi,orden=5, numero pesos=5



Eje de las x constante, eje de las y (amplitud de la señal en voltios). Amplitud_señal vs Valor constante

Pantalla 6.17 Ejemplo problema3_dmi_c. Comparación entre la salida del sistema de una señal senosoidal (dmi) con la señal de referencia del sistema

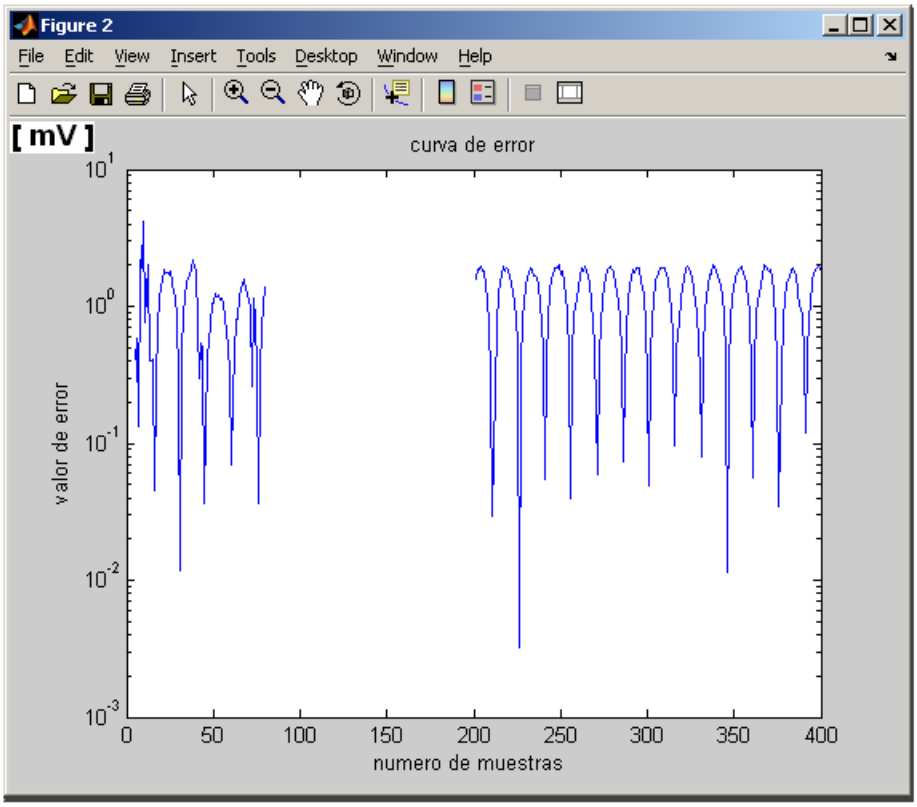
Ejemplo problema3_dmi_c utilizando (opcion_señal=2)



Amplitud de la señal (voltios) vs numero de muestras
señal de referencia (color azul)
señal obtenida por el algoritmo dmi (color rojo)

Pantalla 6.18 Ejemplo problema3_dmi_c. Curva de error de una señal senosoidal en un algoritmo dmi

Ejemplo problema3_ dmi_ c



Amplitud de la señal (voltios) vs numero de muestras

Problema2_dmi_a

Pantalla 6.19 Problema2_dmi_a

```
   * PROBLEMA2_DMI.- Determinar los pesos, el numero de iteraciones y la potencia*.
   * de un grupo de antenas inteligentes utilizando el algoritmo DMI (algoritmo  *.
   * de matriz inversa) , las antenas estan distribuidas en forma de celdas para *.
   * lo cual requeriremos los siguientes datos:                                   *.
   *                                                                             *.
   *  • la señal aplicada a la entrada del filtro,   ej  Xn=sin((2*pi*k)/N)      *.
   *  • el numero de antenas  x                                                  *.
   *  • la señal de referencia, ejemplo d(k)=cos((2*pi*k)/N);                    *.
   *  • el ruido de las diferentes antenas, (hay que tener presente que el       *.
   *    algoritmo LMS en su estructura internamente produce un ruido del 10      *.
   *    por ciento aparte de este ruido puede haber otros ruidos.                *.
   *  • el orden del filtro FIR cuyos coeficientes se van a ajustar, orden       *.
   *  • el número de iteraciones en el aprendizaje, N (este número se hará       *.
   *    hasta que cumpla el error)                                               *.
   *  • la constante de ajuste   u                                               *.
   *  • el vector de pesos inicial. Los voltajes de las antenas V(f)             *.
   *  • el error deseado (puede ser ingresado por teclado) error                 *.
   *                                                                             *.
   *  A su salida devolverá:                                                     *.
   *  • la salida en cada iteración durante el proceso de ajuste, y(t)           *.
   *  • la curva de aprendizaje (grafico de la curva)                            *.
   *  • los vectores de pesos obtenidos durante el proceso. W(t)                 *.
   *  • la Potencia de la antena P(k)                                            *.
   *  • el numero de iteraciones en que converge la curva y se cumple el valor   *.
   *    del error. j                                                             *.
   * alumno: Juan Francisco Alvarez Alvarado                                     *.
   * alumno: Maribel del Rosario Chuez Gonzales                                  *.
   *                                                                             *.
   * Profesor :Ing. Pedro Vargas                                                 *.
   *                                                                             *.
   *******************************************************************************.
teclee enter para proseguir
```

Pantalla 6.20 Datos del programa problema2_dmi_a

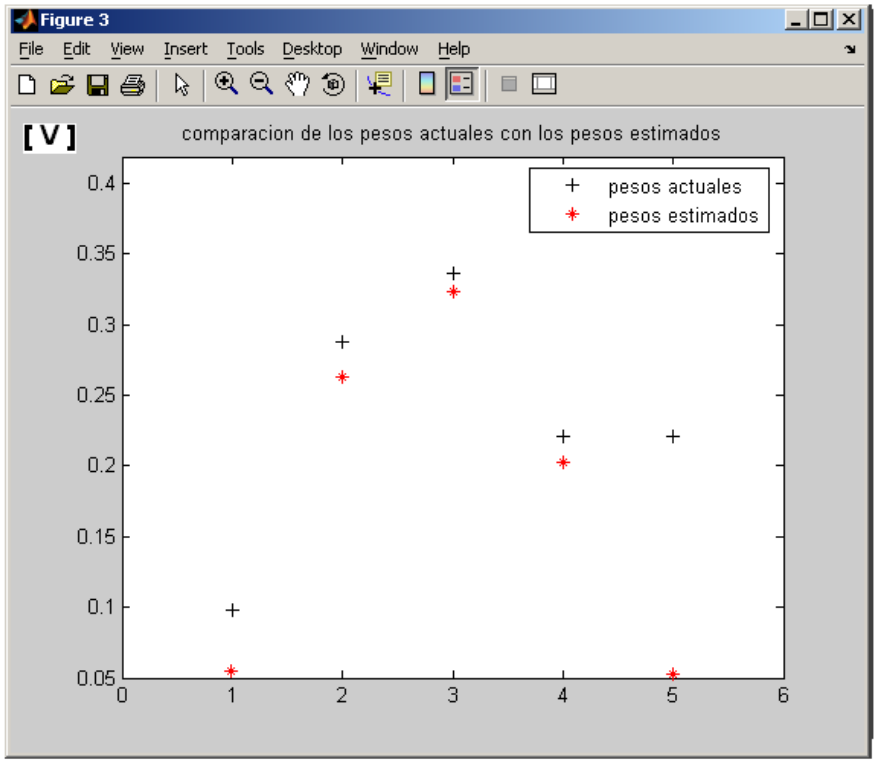
```
 ingrese el orden del filtro, cuyo coeficiente se va ha ajustar....5

orden =

     5

 ingrese el numero de antenas entre 1 y 5
 ingrese el numero de antenas entre 1,2,3,4 o 5....5
ingrese el voltaje1 (en v)..... voltaje1=0.0976
 ingrese la voltaje2 (en V).... voltaje2=0.2873
 ingrese la voltaje3 (en V).....voltaje3=0.3360
 ingrese la voltaje4 (en v)..... voltaje4=0.2210
 ingrese la voltaje5 (en v)..... voltaje5=0.0964
ingrese el ruido en la antena1.....ruido1=0.01
ingrese el ruido en la antena2.....ruido2=0.01
ingrese el ruido en la antena3.....ruido3=0.01
ingrese el ruido en la antena4.....ruido4=0.01
ingrese el ruido en la antena5.....ruido5=0.01
si desea que el ruido adicional vaya a la señal a la entrada digite 1
de lo contrario se considerara que el ruido adicional se agregara a la señal de referencia
opcion_ruido=1
ingrese el error del sistema=0.0000001
```

Pantalla 6.21 Potencias obtenidas con el programa problema2_dmi_a

```
potencia =

    0.3000

potencia1 =

    0.0146

potencia2 =

    0.0914

potencia3 =

    0.1227

potencia4 =

    0.0564

potencia5 =

    0.0143
```

Pantalla 6.22 Ejemplo problema2_dmi_a. Comparación de los pesos estimados

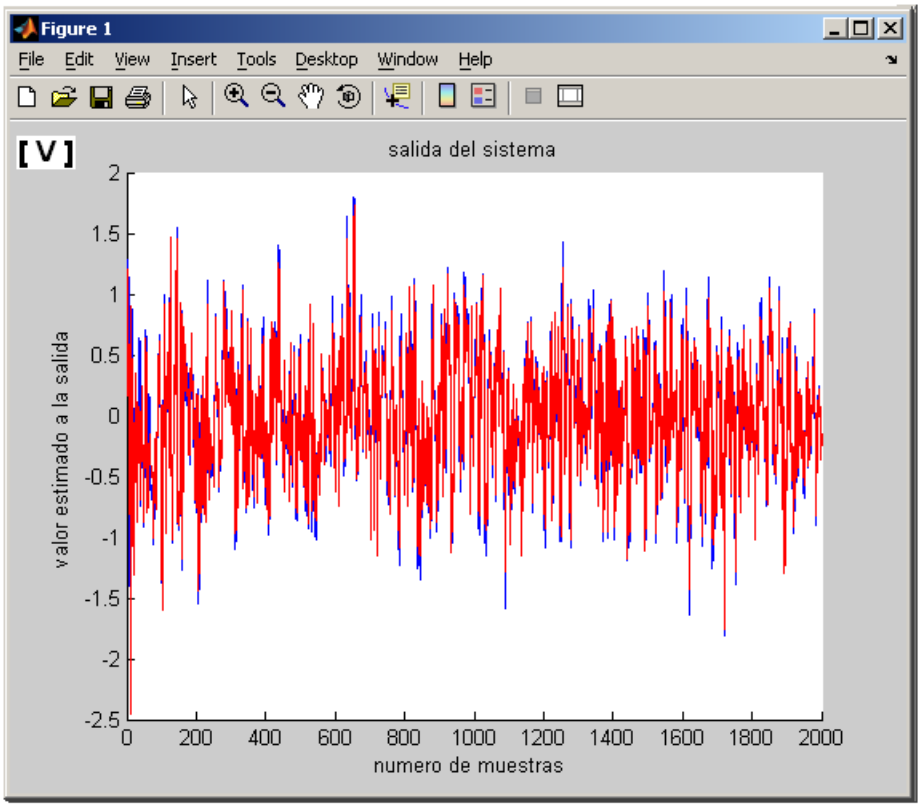
Ejemplo problema2_dmi_a
comparacion de los pesos estimados de una señal gaussiana con los pesos reales o actuales de una señal gaussiana utilizando un algoritmo dmi,orden=5, numero pesos=5



Eje de las x constante, eje de las y (amplitud de la señal en voltios). Amplitud_señal vs Valor constante

Pantalla 6.23 Ejemplo problema2_dmi_a. Comparación entre la salida del sistema de una señal gaussiana (dmi) con la señal de referencia del sistema

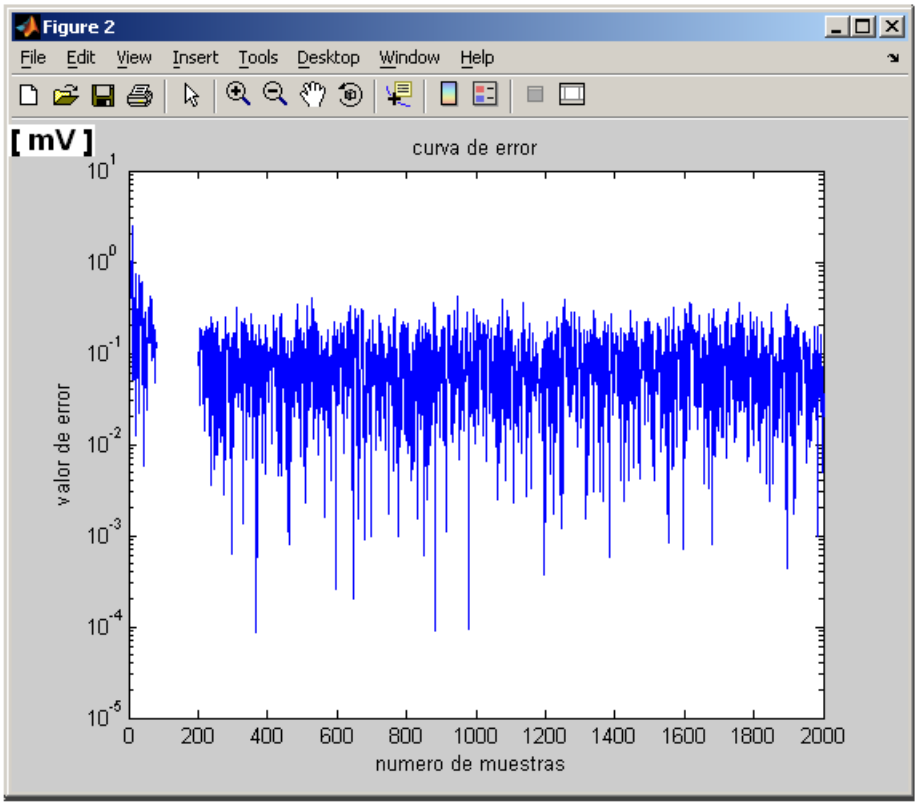
Ejemplo problema2_dmi_a



Amplitud de la señal (voltios) vs numero de muestras
señal de referencia (color azul)
señal obtenida por el algoritmo dmi (color rojo)

Pantalla 6.24 Ejemplo problema2_dmi_a. Curva de error de una señal gaussiana en un algoritmo dmi

Ejemplo problema2_dmi_a



Amplitud de la señal (voltios) vs numero de muestras

Problema1_dmi_a

Pantalla 6.25 Problema1_dmi_a

```
   * PROBLEMA1_dmi.- Determinar los pesos, el numero de iteraciones y la potencia*.
   * de un grupo de antenas inteligentes utilizando el algoritmo DMI (algoritmo  *.
   * de matriz inversa, las antenas estan distribuidas en forma de celdas para lo*.
   * cual requeriremos los siguientes datos:                                      *.
   *                                                                              *.
   *  • la señal aplicada a la entrada del filtro,   ej  Xn=2*sin((2*pi*k)/N)     *.
   *  • el numero de antenas  x                                                   *.
   *  • la señal de referencia, d(k)=2*cos((2*pi*k)/N);                           *.
   *  • el ruido de las diferentes antenas, (hay que tener presente que el        *.
   *    algoritmo LMS en su estructura internamente produce un ruido del 10       *.
   *    aparte de este ruido puede haber otros.  n(f)                             *.
   *  • el orden del filtro FIR cuyos coeficientes se van a ajustar, orden        *.
   *  • el número de iteraciones en el aprendizaje, N (este número se hará        *.
   *    hasta que cumpla el error)                                                *.
   *  • la constante de ajuste   u                                                *.
   *  • el vector de pesos inicial. Los voltajes de las antenas V(f)              *.
   *  • el error deseado (puede ser ingresado por teclado) error                  *.
   *                                                                              *.
   *  A su salida devolverá:                                                      *.
   *  • la salida en cada iteración durante el proceso de ajuste, y(t)            *.
   *  • la curva de aprendizaje (grafico de la curva)                             *.
   *  • los vectores de pesos obtenidos durante el proceso. W(t)                  *.
   *  • la Potencia de la antena P(k)                                             *.
   *  • el numero de iteraciones en que converge la curva y se cumple el valor    *.
   *    del error.  j                                                             *.
   * alumno: Juan Francisco Alvarez Alvarado                                      *.
   * alumno: Maribel del Rosario Chuez Gonzales                                   *.
   *                                                                              *.
   * Profesor :Ing. Pedro Vargas                                                  *.
   *                                                                              *.
   ********************************************************************************.
teclee enter para proseguir
```

Pantalla 6.26 Datos del programa problema1_dmi_a

```
 ingrese el orden del filtro, cuyo coeficiente se va ha ajustar....5

orden =

     5

 ingrese el numero de antenas entre 1 y 5
 ingrese el numero de antenas entre 1,2,3,4 o 5....5
ingrese el voltaje1 (en v)..... voltaje1=0.26
 ingrese la voltaje2 (en V).... voltaje2=0.24
 ingrese la voltaje3 (en V).....voltaje3=0.21
 ingrese la voltaje4 (en v)..... voltaje4=0.18
 ingrese la voltaje5 (en v)..... voltaje5=0.15
ingrese el ruido en la antena1.....ruido1=0.01
ingrese el ruido en la antena2.....ruido2=0.01
ingrese el ruido en la antena3.....ruido3=0.01
ingrese el ruido en la antena4.....ruido4=0.02
ingrese el ruido en la antena5.....ruido5=0.02
si desea que el ruido adicional vaya a la señal a la entrada digite 1
 de lo contrario se considerara que el ruido adicional se agregara a la señal de referencia
opcion_ruido=1
ingrese el error del sistema=0.0000001
```

Pantalla 6.27 Potencias obtenidas con el programa problema1_dmi_a

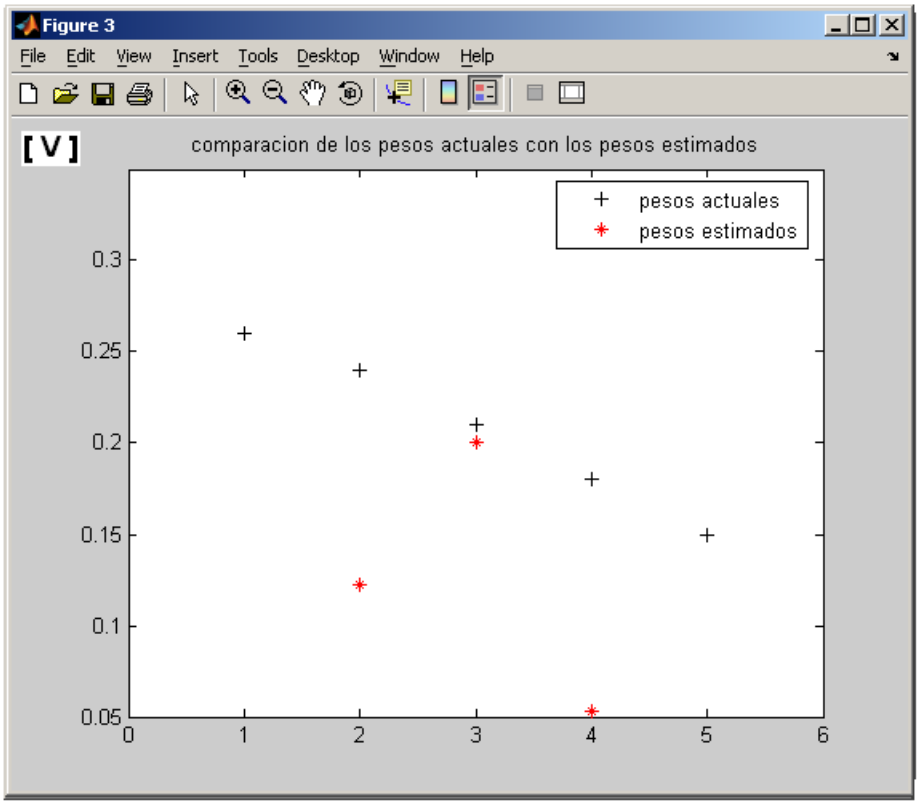
```
potencia =

   0.5090

potencia1 =

   0.0817

potencia2 =

   0.0710

potencia3 =

   0.0565

potencia4 =

   0.0478
```

potencia5 =

0.0363

Pantalla 6.28 Ejemplo problema1_dmi_a. Comparación de los pesos estimados

Ejemplo problema1_dmi_a
comparacion de los pesos estimados de una señal senosoidal con los pesos reales o actuales de una señal senosoidal utilizando un algoritmo dmi,orden=5, numero pesos=5

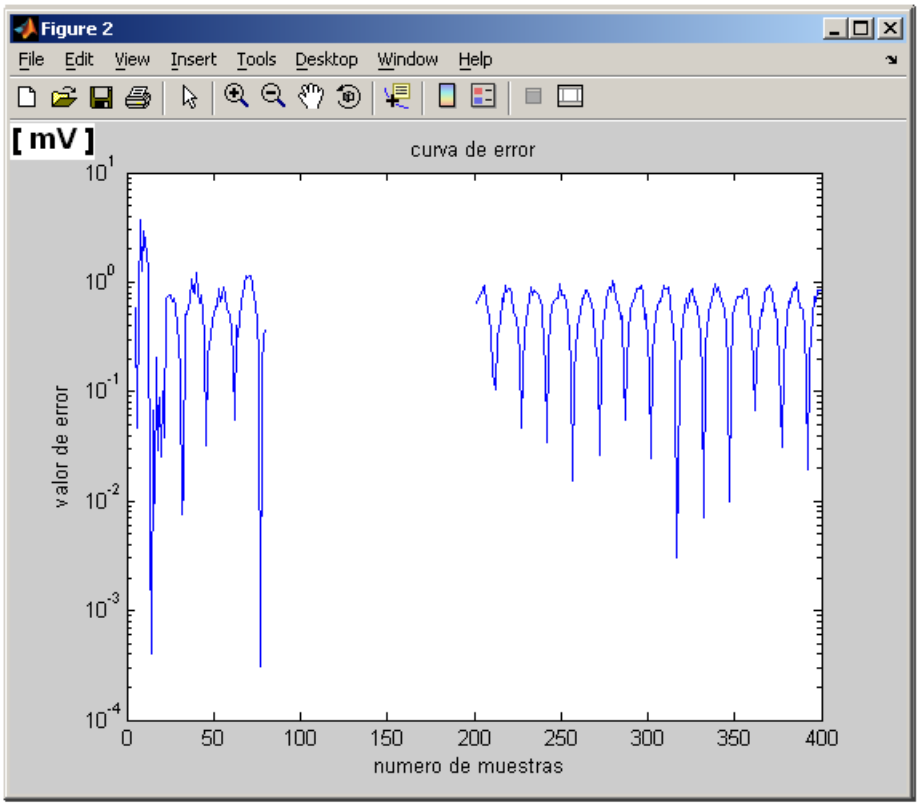


Eje de las x constante, eje de las y (amplitud de la señal en voltios). Amplitud_señal vs Valor constante

Solo 3 de los cinco pesos se evaluaron algo cerca. Y dos más lejanos fuera del cuadro de la grafica

Pantalla 6.29 Ejemplo problema1_dmi_a. Curva de error de una señal senosoidal en un algoritmo dmi

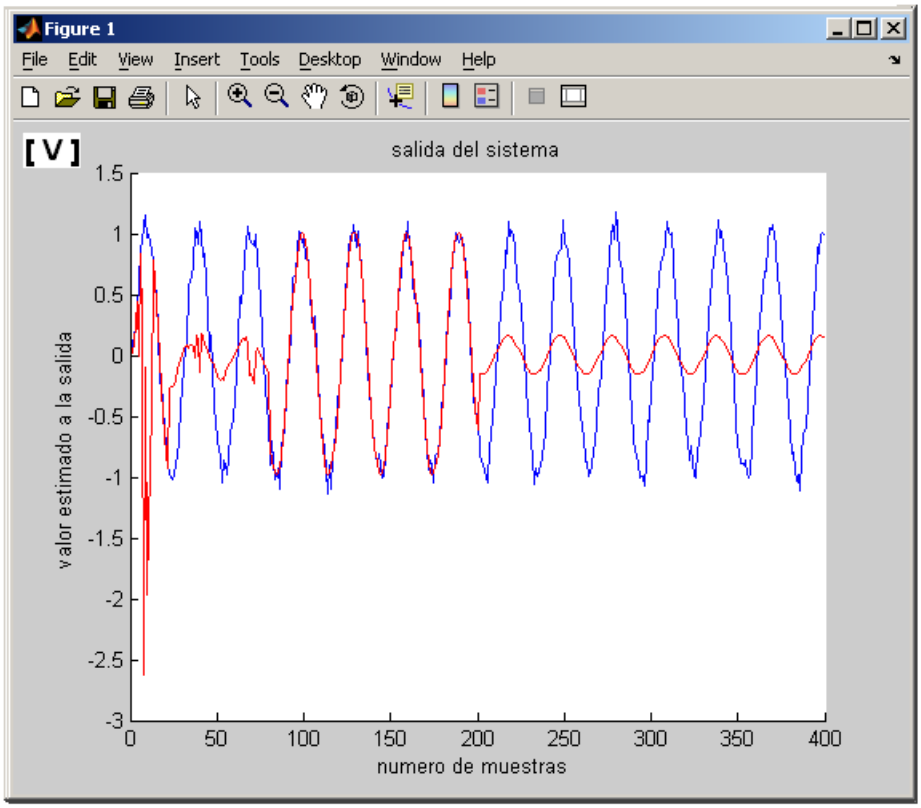
Ejemplo problema1_dmi_a



Amplitud de la señal (voltios) vs numero de muestras

Pantalla 6.30 Ejemplo problema1_dmi_a. Comparación entre la salida del sistema de una señal senosoidal (dmi) con la señal de referencia del sistema

Ejemplo problema1_dmi_a



Amplitud de la señal (voltios) vs numero de muestras
señal de referencia (color azul)
señal obtenida por el algoritmo dmi (color rojo)

Comparación de resultados entre los ejemplos problema3_lms_x, Problema3_rls_x, problema3_cma_x y problema3_dmi_x.

Pantalla 6.31 Datos del programa problema3_lms_x, problema3_rls_x, problema3_cma_x y problema3_dmi_x

```
 ingrese el orden del filtro, cuyo coeficiente se va ha ajustar....5

orden =

     5

 ingrese el numero de antenas entre 1 y 5
 ingrese el numero de antenas entre 1,2,3,4 o 5....5
ingrese el voltaje1 (en v)..... voltaje1=0.10
 ingrese la voltaje2 (en V).... voltaje2=0.08
 ingrese la voltaje3 (en V).....voltaje3=0.02
 ingrese la voltaje4 (en v)..... voltaje4=0.05
 ingrese la voltaje5 (en v)..... voltaje5=0.10
ingrese el ruido en la antena1.....ruido1=0.01
ingrese el ruido en la antena2.....ruido2=0.01
ingrese el ruido en la antena3.....ruido3=0.01
ingrese el ruido en la antena4.....ruido4=0.02
ingrese el ruido en la antena5.....ruido5=0.02
si desea que el ruido adicional vaya a la señal a la entrada digite 1
 de lo contrario se considerara que el ruido adicional se agregara a la señal de referencia
opcion_ruido=1
ingrese el error del sistema=0.0000001
si desea que la señal de entrada se use como señal de referencia digite 1
 de lo contrario se considerara otra señal diferente como señal de referencia
opcion_señal=2

h =

    0.1000
    0.0800
    0.0200
    0.0500
    0.1000
```

Pantalla 6.32 Potencias obtenidas con el programa problema3_lms_x

```
potencia =

    0.4983

potencia1 =

    0.0164

potencia2 =

    0.0125

potencia3 =

    0.0057

potencia4 =

    0.0094

potencia5 =

    0.0186
```

Pantalla 6.33 Potencias obtenidas con el programa problema3_rls_x

```
potencia =

    0.4959

potencia1 =

    0.0162

potencia2 =

    0.0124

potencia3 =

    0.0057

potencia4 =
```

```
    0.0093

potencia5 =

    0.0184
```

Pantalla 6.34 Potencias obtenidas con el programa problema3_cma_x

```
potencia =

    0.4866

potencia1 =

    0.0158

potencia2 =

    0.0120

potencia3 =

    0.0055

potencia4 =

    0.0091

potencia5 =

    0.0180
```

Pantalla 6.35 Potencias obtenidas con el programa problema3_dmi_x

```
potencia =

    0.5111

potencia1 =

    0.0198

potencia2 =
```

0.0153

potencia3 =

0.0067

potencia4 =

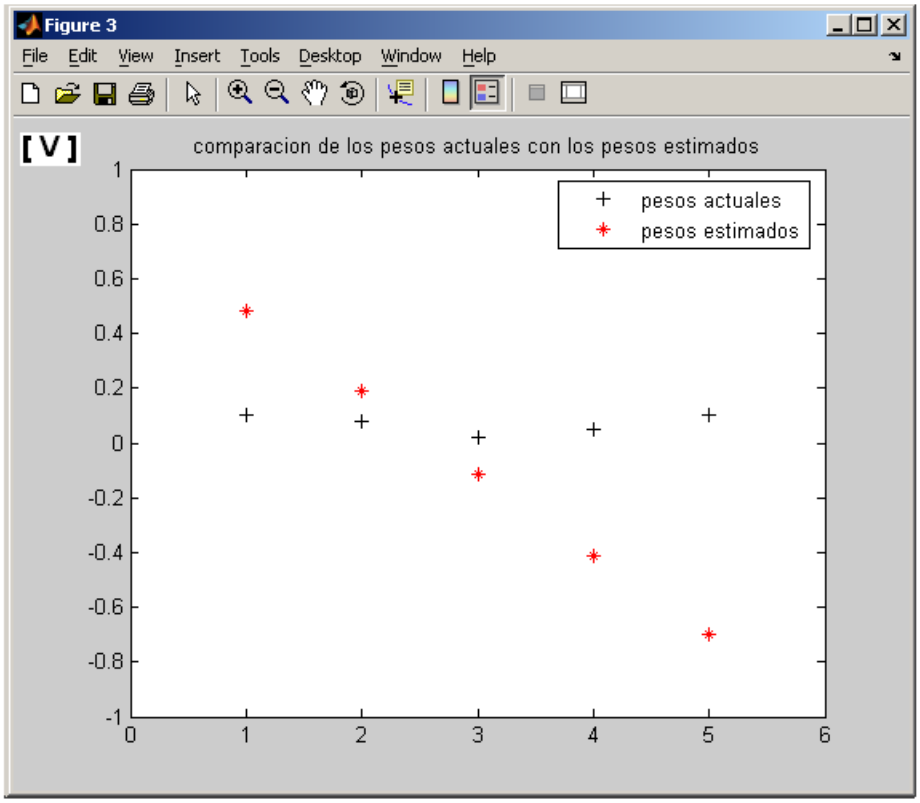
0.0116

potencia5 =

0.0223

Pantalla 6.36 Ejemplo problema3_lms_x. Comparación de los pesos estimados

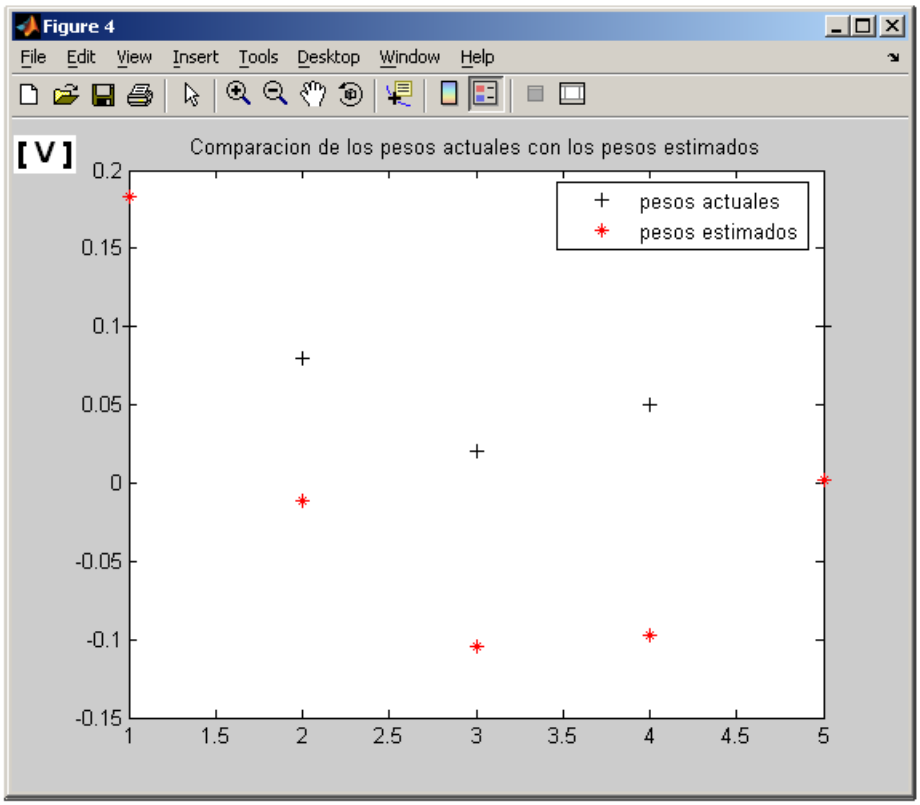
Ejemplo problema3_lms_x
comparacion de los pesos estimados de una señal senosoidal con los pesos reales o actuales de una señal senosoidal utilizando un algoritmo lms ,orden=5, numero pesos=5



Eje de las x constante, eje de las y (amplitud de la señal en voltios). Amplitud_señal vs Valor constante

Pantalla 6.37 Ejemplo problema3_rls_x. Comparación de los pesos estimados

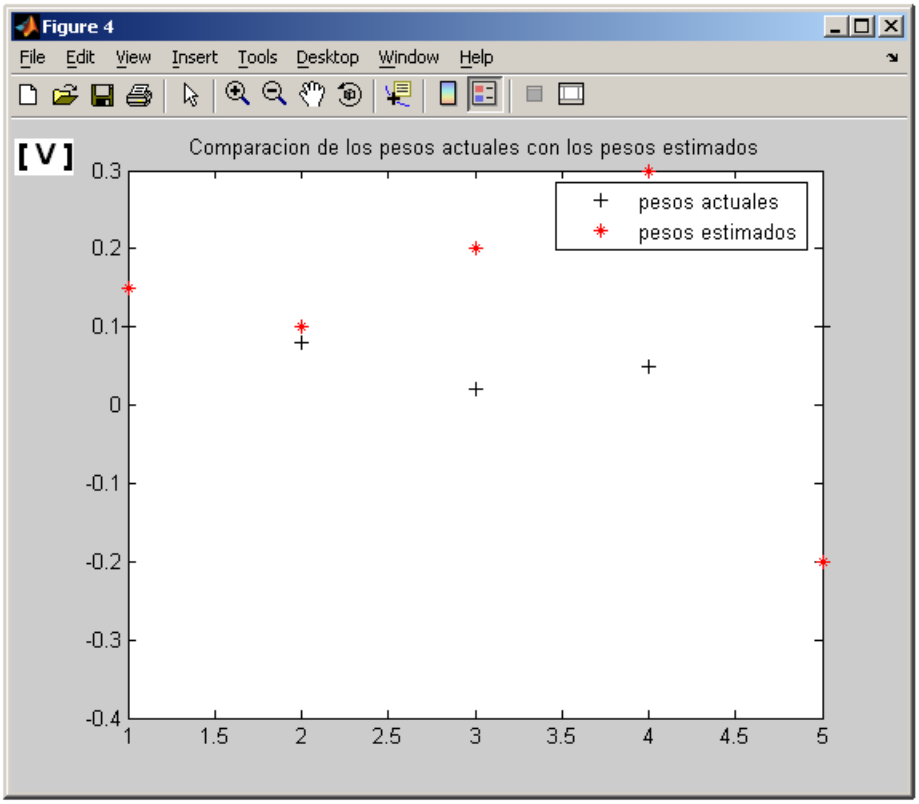
Ejemplo problema3_rls_x
comparacion de los pesos estimados de una señal senosoidal con los pesos reales o actuales de una señal senosoidal utilizando un algoritmo rls ,orden=5, numero pesos=5



Eje de las x constante, eje de las y (amplitud de la señal en voltios). Amplitud_señal vs Valor constante

Pantalla 6.38 Ejemplo problema3_cma_x. Comparación de los pesos estimados

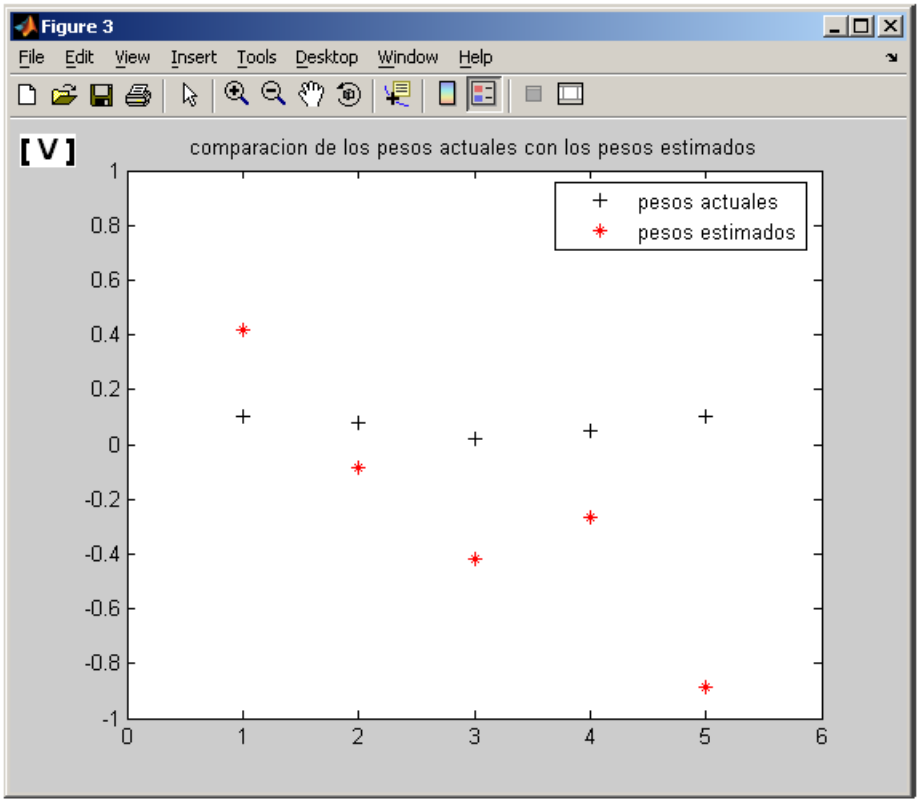
Ejemplo problema3_cma_x
comparacion de los pesos estimados de una señal senosoidal con los pesos reales o actuales de una señal senosoidal utilizando un algoritmo cma,orden=5, numero pesos=5



Eje de las x constante, eje de las y (amplitud de la señal en voltios). Amplitud_señal vs Valor constante

Pantalla 6.39 Ejemplo problema3_dmi_x. Comparación de los pesos estimados

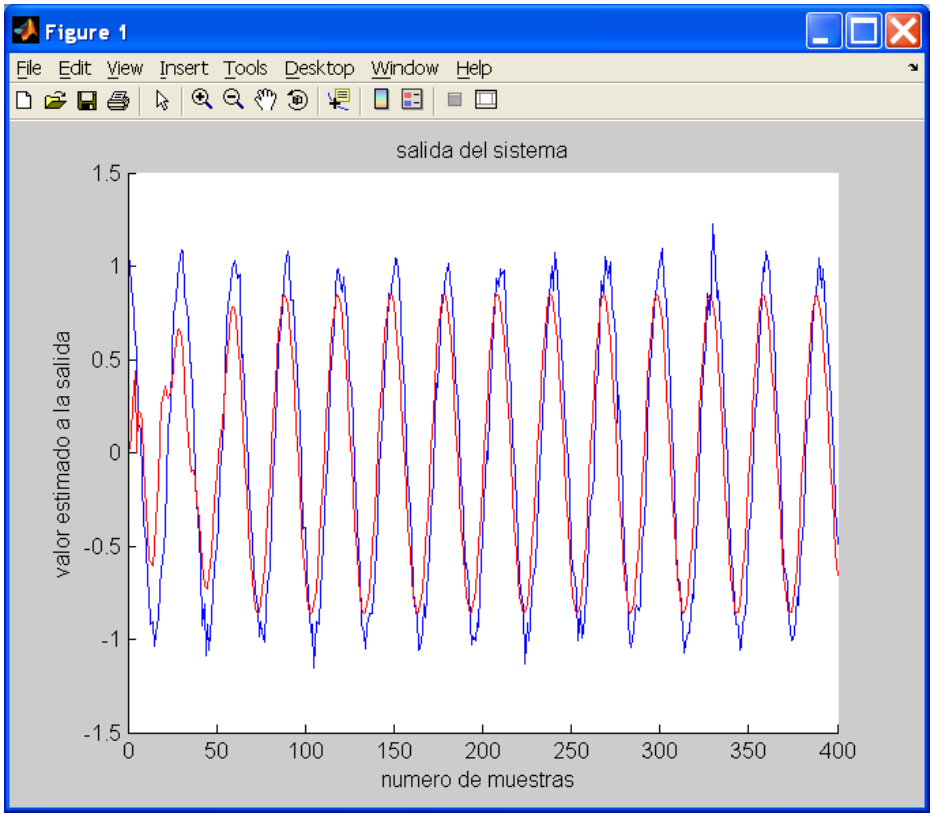
Ejemplo problema3_dmi_x
comparacion de los pesos estimados de una señal senosoidal
con los pesos reales o actuales de una señal senosoidal
utilizando un algoritmo dmi,orden=5, numero pesos=5



Eje de las x constante, eje de las y (amplitud de la señal en voltios). Amplitud_señal vs Valor constante

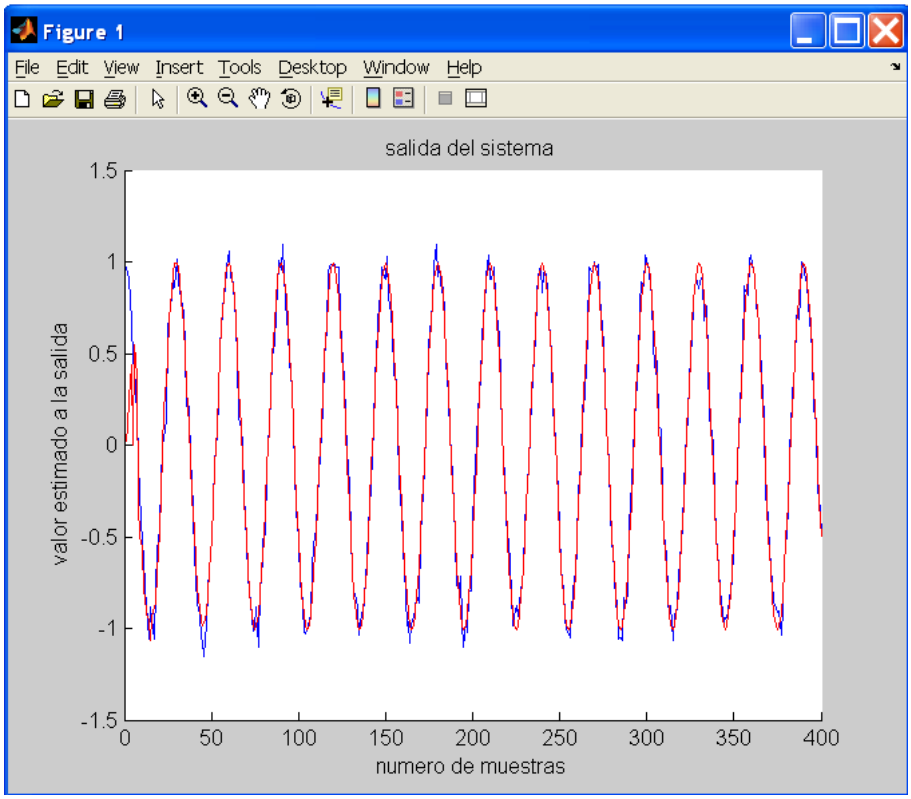
Pantalla 6.40 Ejemplo problema3_lms_x. Comparación entre la salida del sistema de una señal senosoidal (lms) con la señal de referencia del sistema.

Señal de referencia color azul, señal obtenida por el algoritmo lms color rojo



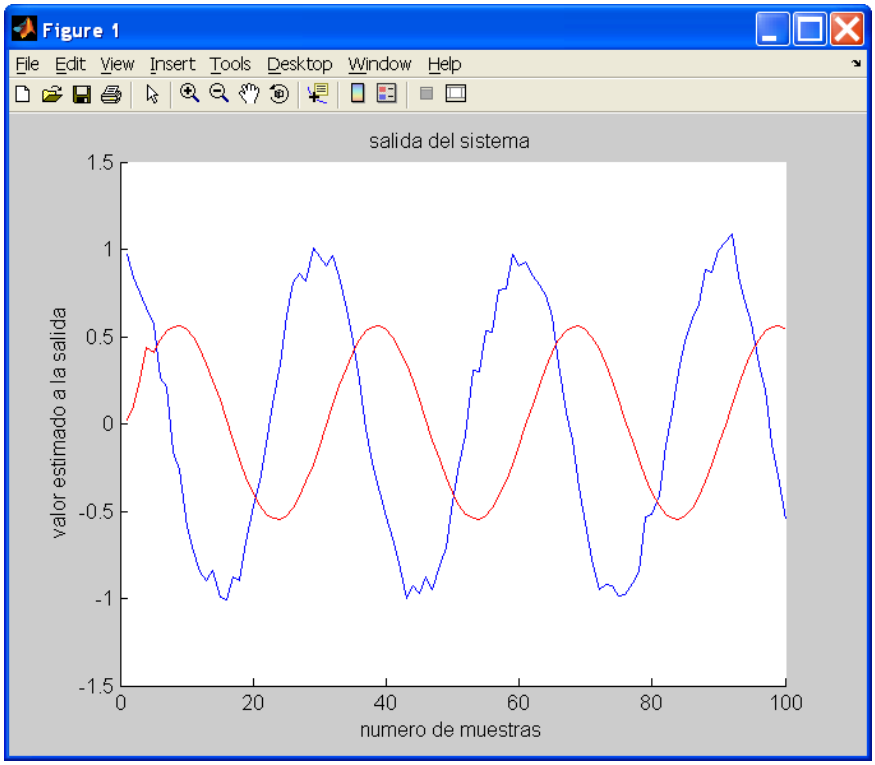
Pantalla 6.41 Ejemplo problema3_rls_x. Comparación entre la salida del sistema de una señal senosoidal (rls) con la señal de referencia del sistema

Señal de referencia color azul, señal obtenida por el algoritmo rls color rojo



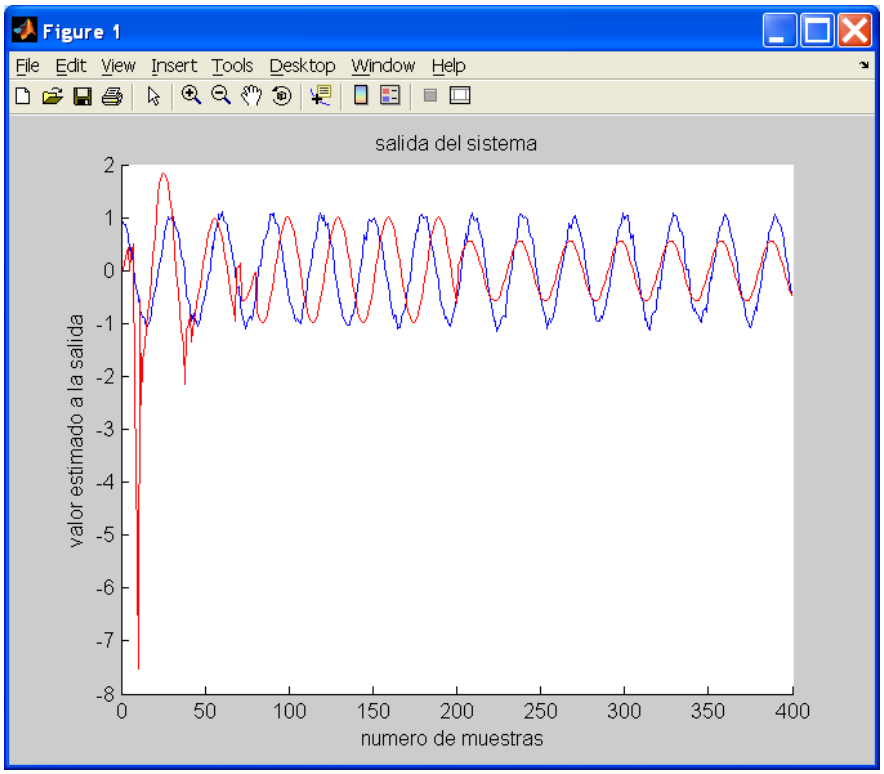
Pantalla 6.42 Ejemplo problema3_cma_x. Comparación entre la salida del sistema de una señal senosoidal (cma) con la señal de referencia del sistema

Señal de referencia color azul, señal obtenida por el algoritmo cma color rojo

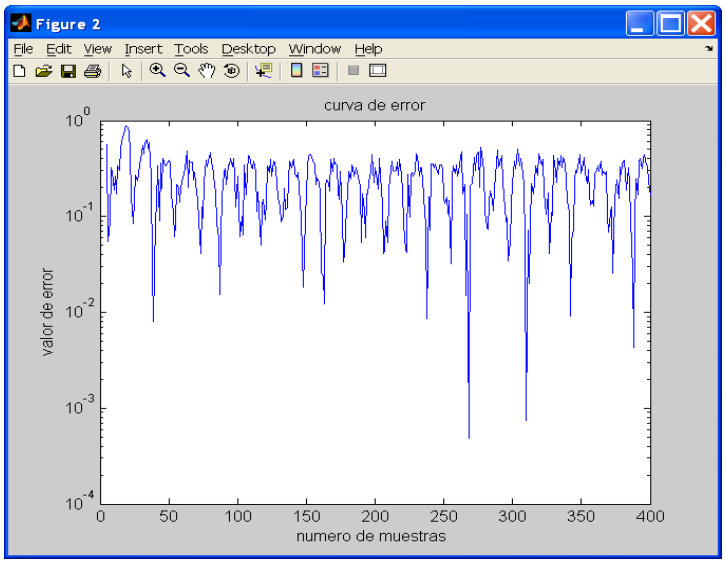


Pantalla 6.43 Ejemplo problema3_dmi_x. Comparación entre la salida del sistema de una señal senosoidal (dmi) con la señal de referencia del sistema

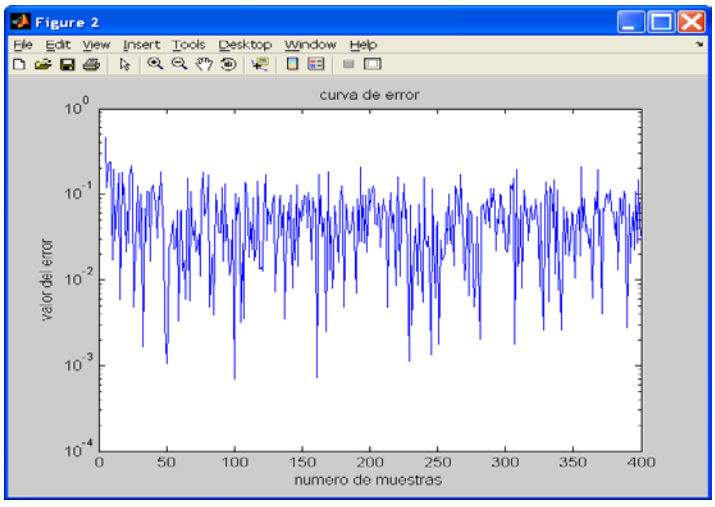
Señal de referencia color azul, señal obtenida por el algoritmo dmi color rojo



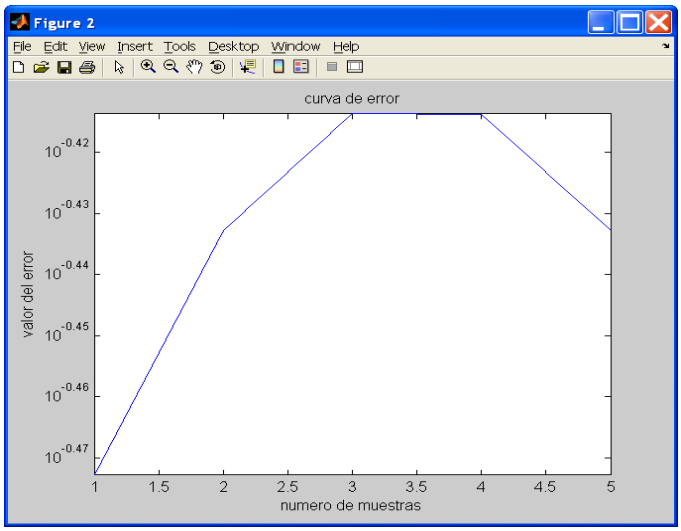
Pantalla 6.44 Ejemplo problema3_lms_x. Curva de error de una señal senosoidal en un algoritmo lms



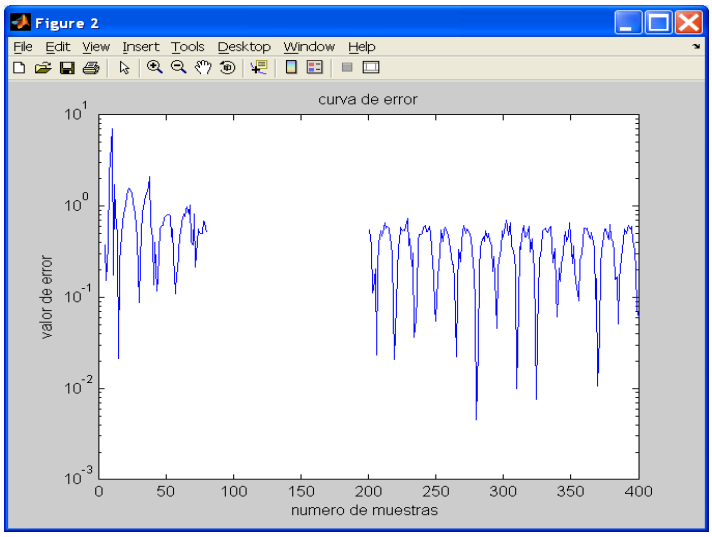
Pantalla 6.45 Ejemplo problema3_rls_x. Curva de error de una señal senosoidal en un algoritmo rls



Pantalla 6.46 Ejemplo problema3_cma_x. Curva de error de una señal senosoidal en un algoritmo cma



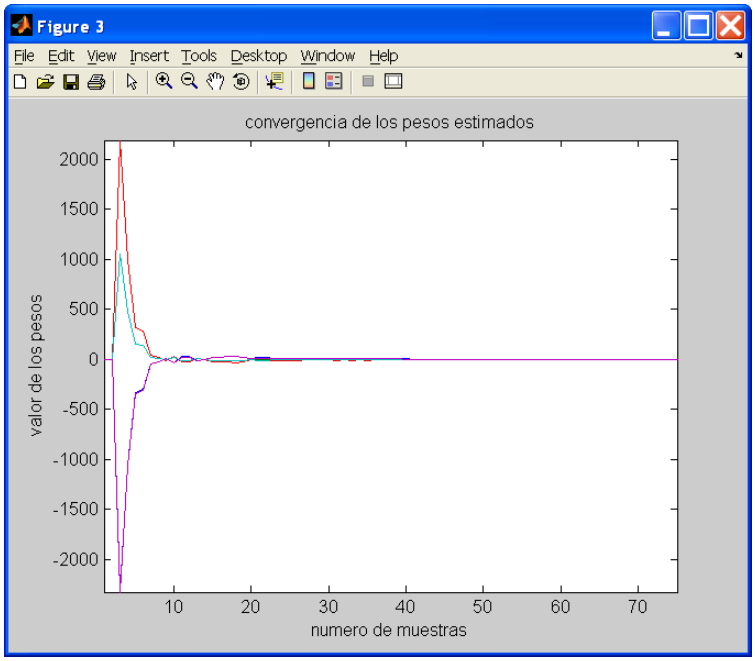
Pantalla 6.47 Ejemplo problema3_dmi_x. Curva de error de una señal senosoidal en un algoritmo dmi



el algoritmo lms no tiene grafico de convergencia de los pesos estimados los algoritmos cma y dmi sus graficos de convergencia de pesos estimados siempre dan como si hubiera una convergencia intantanea lo cual no siempre es cierto y puede dar datos erroneos sobre el desarrollo de la curva.

este grafico funciona mejor es en el algoritmo rls.

Pantalla 6.48 Ejemplo problema3_rls_x. Convergencia de los pesos estimados para una señal senosoidal en un algoritmo rls



Comparacion de resultados entre los ejemplos problema3_lms_z, Problema3_rls_z, problema3_cma_z y problema3_dmi_z.

Pantalla 6.49 Datos del programa problema3_lms_z, problema3_rls_z, problema3_cma_z y problema3_dmi_z

```
 ingrese el orden del filtro, cuyo coeficiente se va ha ajustar....5

orden =

     5

 restriccion: ingrese un numero de antenas igual al del orden del filtro ingresado)
 ingrese el numero de antenas entre 1,2,3,4,5,6,7,8,9,10,11,12,13,14 o 15....5
ingrese el voltaje1 (en v)..... voltaje1=0.10
 ingrese la voltaje2 (en V).... voltaje2=0.20
 ingrese la voltaje3 (en V).....voltaje3=0.22
 ingrese la voltaje4 (en v)..... voltaje4=0.15
 ingrese la voltaje5 (en v)..... voltaje5=0.09
ingrese el ruido en la antena1.....ruido1=0.01
ingrese el ruido en la antena2.....ruido2=0.01
ingrese el ruido en la antena3.....ruido3=0.01
ingrese el ruido en la antena4.....ruido4=0.02
ingrese el ruido en la antena5.....ruido5=0.02
si desea que el ruido adicional vaya a la señal a la entrada digite 1
 de lo contrario se considerara que el ruido adicional se agregara a la señal de referencia
opcion_ruido=1
ingrese el error del sistema=0.0000001
si desea que la señal de entrada se use como señal de referencia digite 1
de lo contrario se considerara otra señal diferente como señal de referencia
opcion_señal=2
```

Pantalla 6.50 Potencias obtenidas con el programa problema3_lms_z

```
potencia =

    0.4983

potencia1 =

    0.0164

potencia2 =

    0.0478

potencia3 =

    0.0564

potencia4 =

    0.0328

potencia5 =

    0.0164
```

Pantalla 6.51 Potencias obtenidas con el programa problema3_rls_z

```
potencia =

    0.4959

potencia1 =

    0.0162

potencia2 =

    0.0474

potencia3 =

    0.0561

potencia4 =
```

0.0326

potencia5 =

0.0162

Pantalla 6.52 Potencias obtenidas con el programa problema3_cma_z

potencia =

0.4866

potencia1 =

0.0158

potencia2 =

0.0466

potencia3 =

0.0552

potencia4 =

0.0319

potencia5 =

0.0158

Pantalla 6.53 Potencias obtenidas con el programa problema3_dmi_z

potencia =

0.5111

potencia1 =

0.0198

potencia2 =

0.0153

potencia3 =

0.0067

potencia4 =

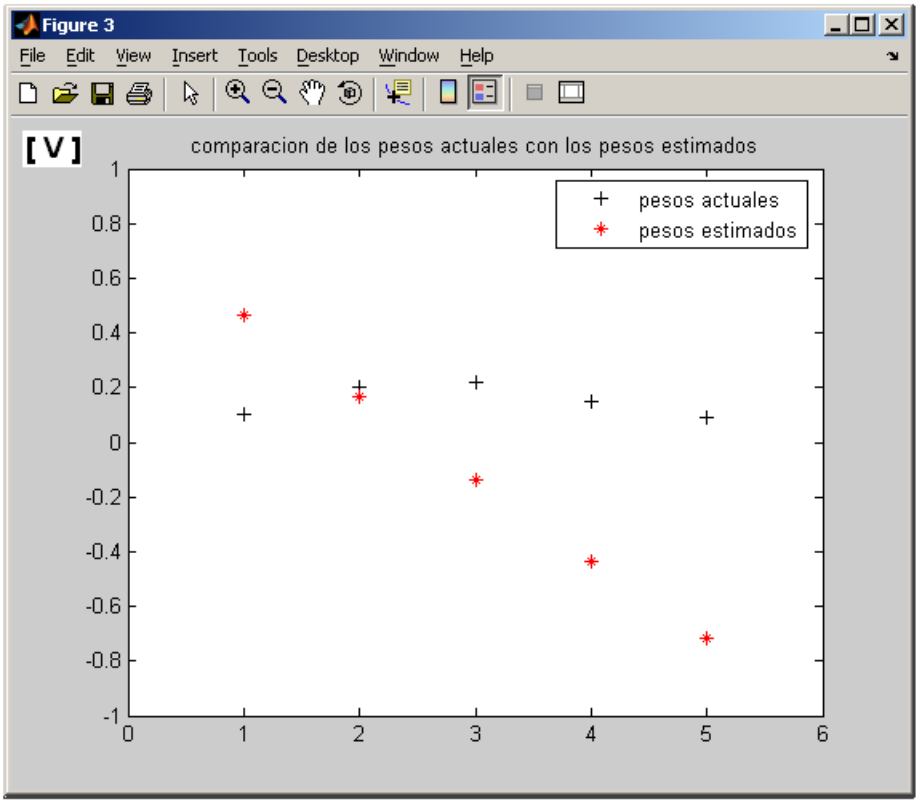
0.0116

potencia5 =

0.0223

Pantalla 6.54 Ejemplo problema3_lms_z. Comparación de los pesos estimados

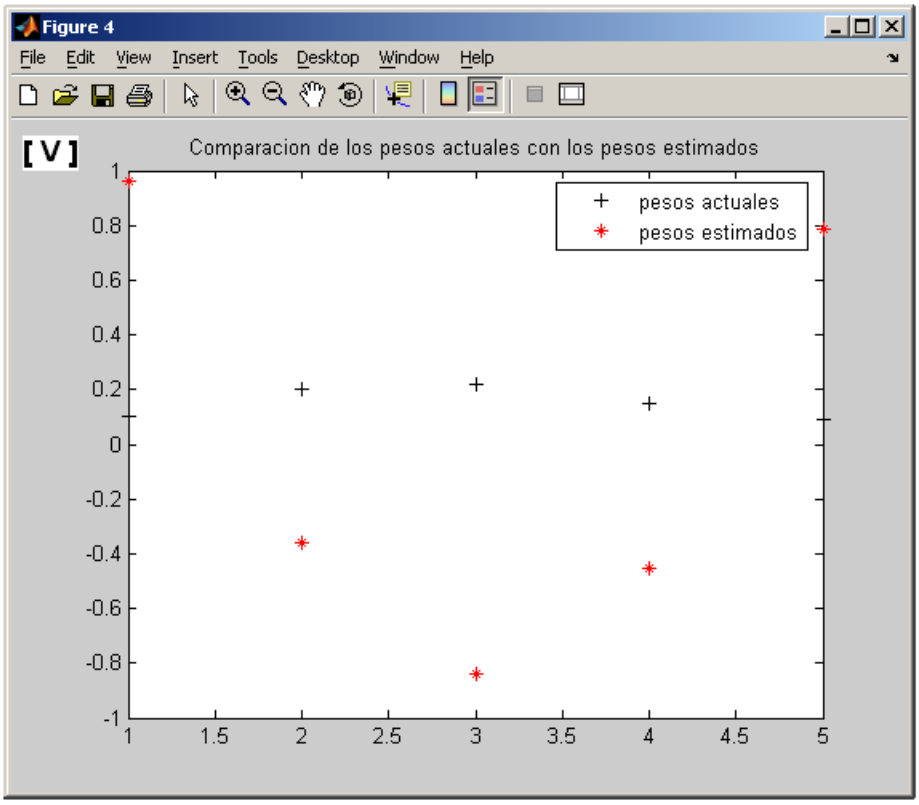
Ejemplo problema3_lms_z
comparacion de los pesos estimados de una señal senosoidal con los pesos reales o actuales de una señal senosoidal utilizando un algoritmo lms,orden=5, numero pesos=5



Eje de las x constante, eje de las y (amplitud de la señal en voltios). Amplitud_señal vs Valor constante

Pantalla 6.55 Ejemplo problema3_rls_z. Comparación de los pesos estimados

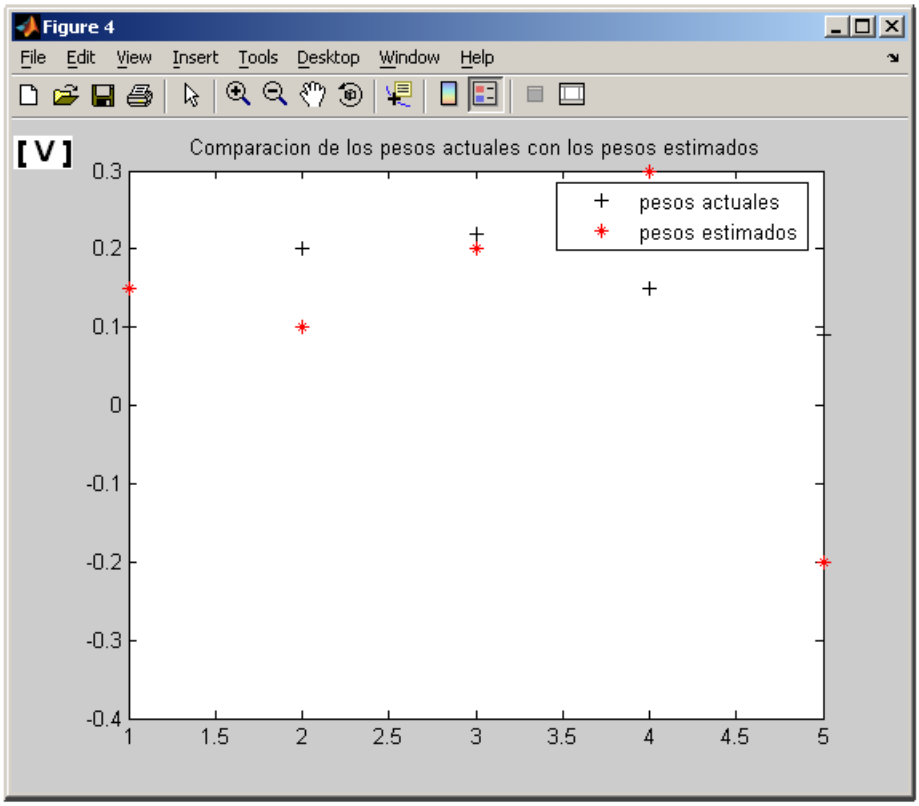
Ejemplo problema3_rls_z
comparacion de los pesos estimados de una señal senosoidal con los pesos reales o actuales de una señal senosoidal utilizando un algoritmo rls ,orden=5, numero pesos=5



Eje de las x constante, eje de las y (amplitud de la señal en voltios). Amplitud_señal vs Valor constante

Pantalla 6.56 Ejemplo problema3_cma_z. Comparación de los pesos estimados

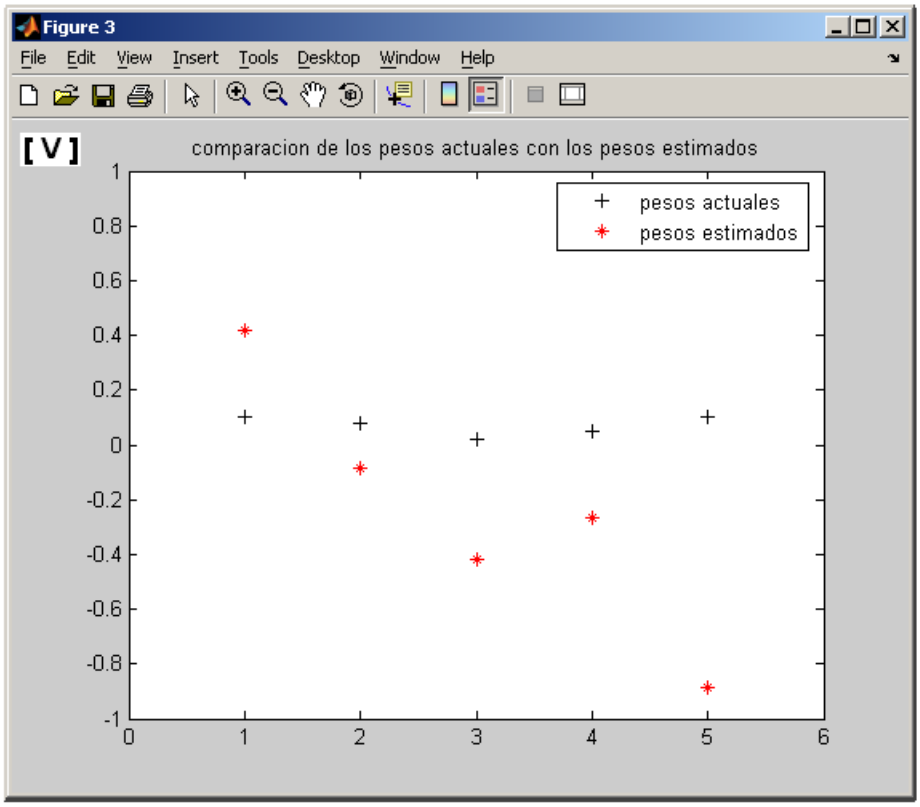
Ejemplo problema3_cma_z
comparacion de los pesos estimados de una señal senosoidal con los pesos reales o actuales de una señal senosoidal utilizando un algoritmo cma,orden=5, numero pesos=5



Eje de las x constante, eje de las y (amplitud de la señal en voltios). Amplitud_señal vs Valor constante

Pantalla 6.57 Ejemplo problema3_dmi_z. Comparación de los pesos estimados

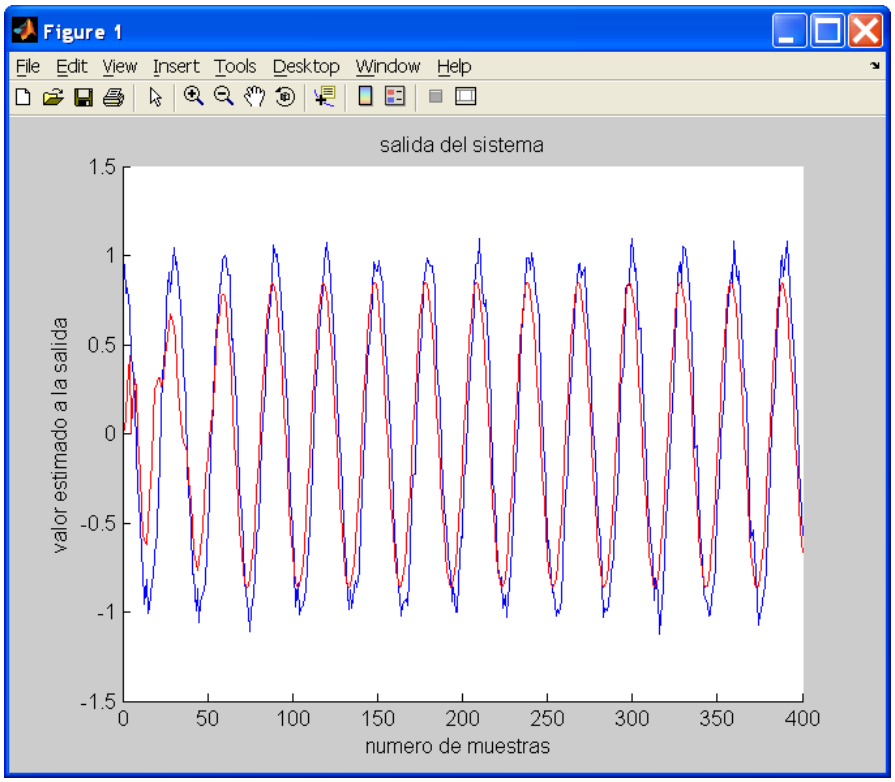
Ejemplo problema3_dmi_z
comparacion de los pesos estimados de una señal senosoidal con los pesos reales o actuales de una señal senosoidal utilizando un algoritmo dmi,orden=5, numero pesos=5



Eje de las x constante, eje de las y (amplitud de la señal en voltios). Amplitud_señal vs Valor constante

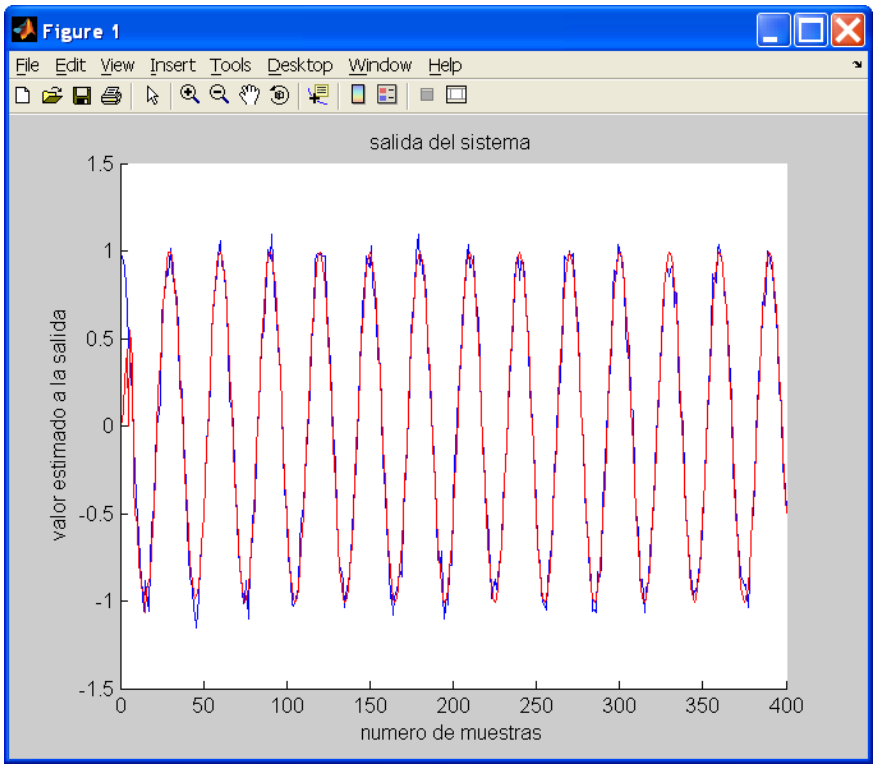
Pantalla 6.58 Ejemplo problema3_lms_z. Comparación entre la salida del sistema de una señal senosoidal (lms) con la señal de referencia del sistema

Señal de referencia color azul, señal obtenida por el algoritmo lms color rojo



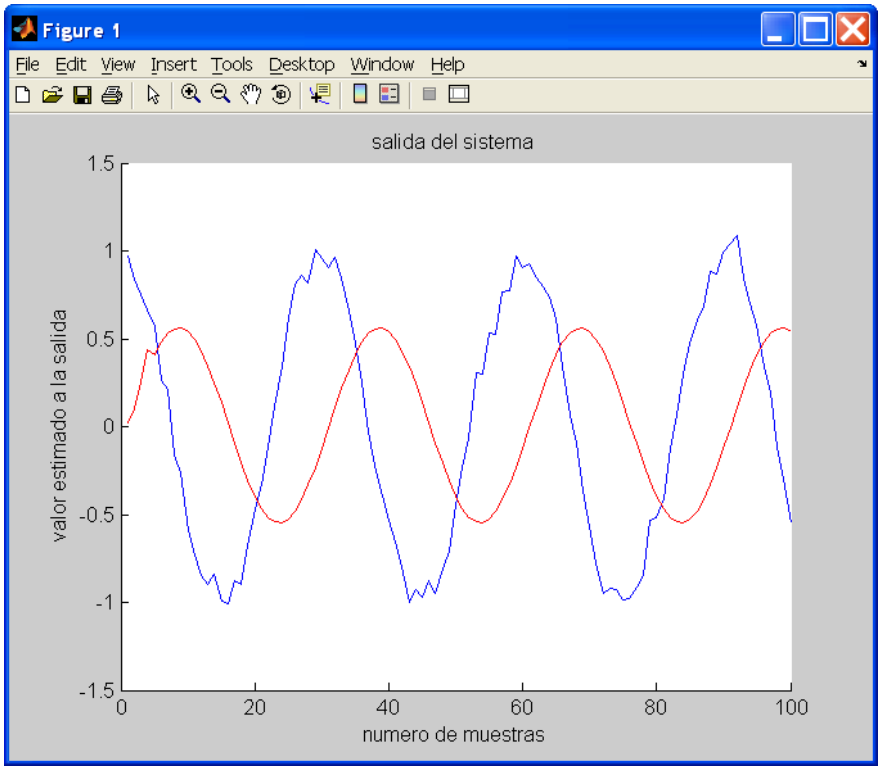
Pantalla 6.59 Ejemplo problema3_rls_z. Comparación entre la salida del sistema de una señal senosoidal (rls) con la señal de referencia del sistema

Señal de referencia color azul, señal obtenida por el algoritmo rls color rojo



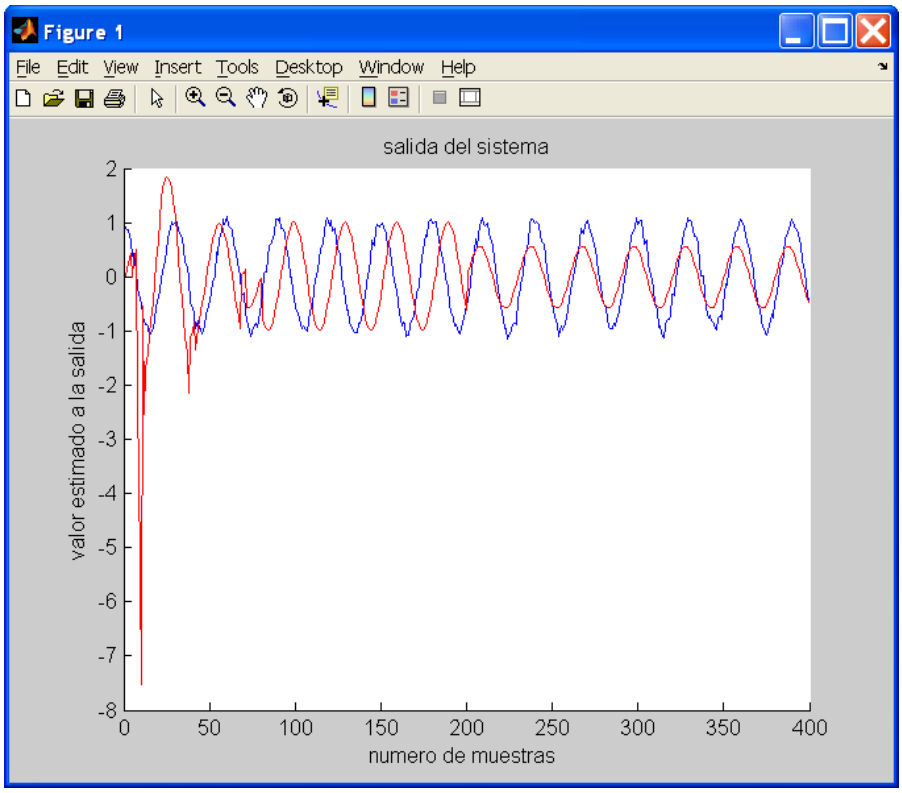
Pantalla 6.60 Ejemplo problema3_cma_z. Comparación entre la salida del sistema de una señal senosoidal (cma) con la señal de referencia del sistema

Señal de referencia color azul, señal obtenida por el algoritmo cma color rojo

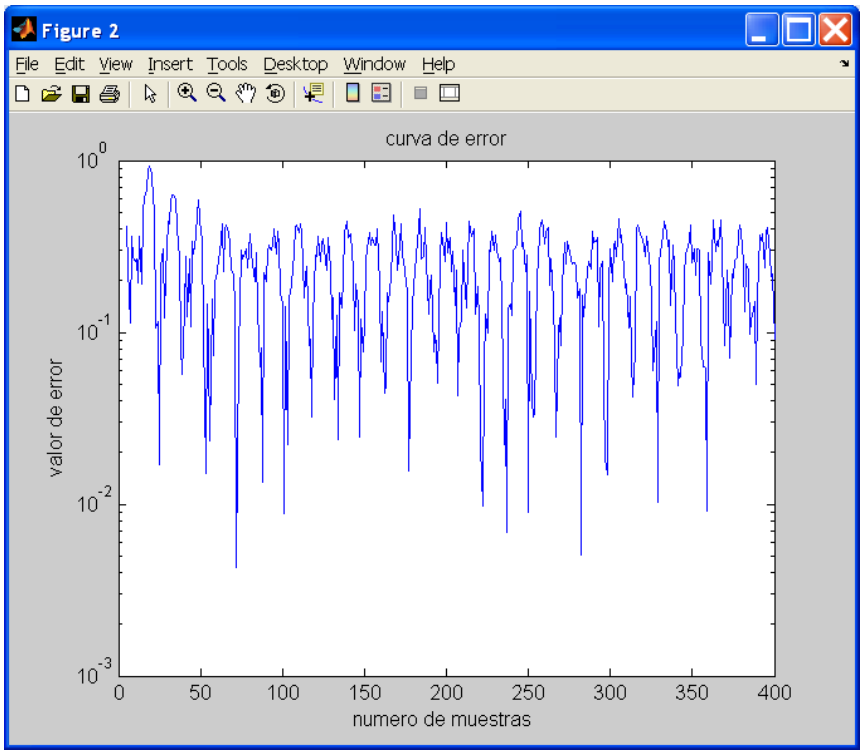


Pantalla 6.61 Ejemplo problema3_dmi_z. Comparación entre la salida del sistema de una señal senosoidal (dmi) con la señal de referencia del sistema

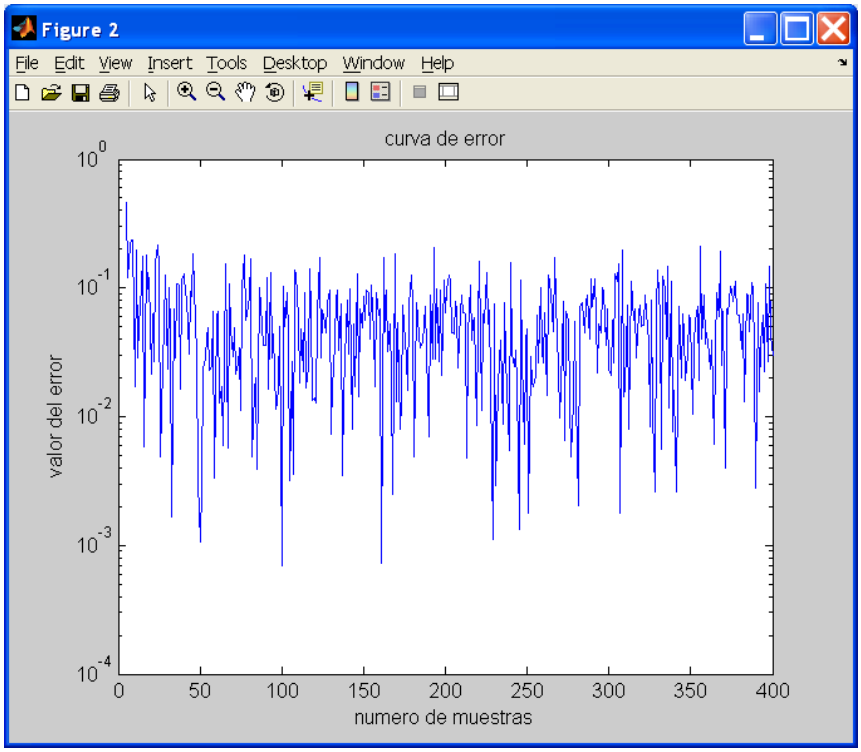
Señal de referencia color azul, señal obtenida por el algoritmo dmi color rojo



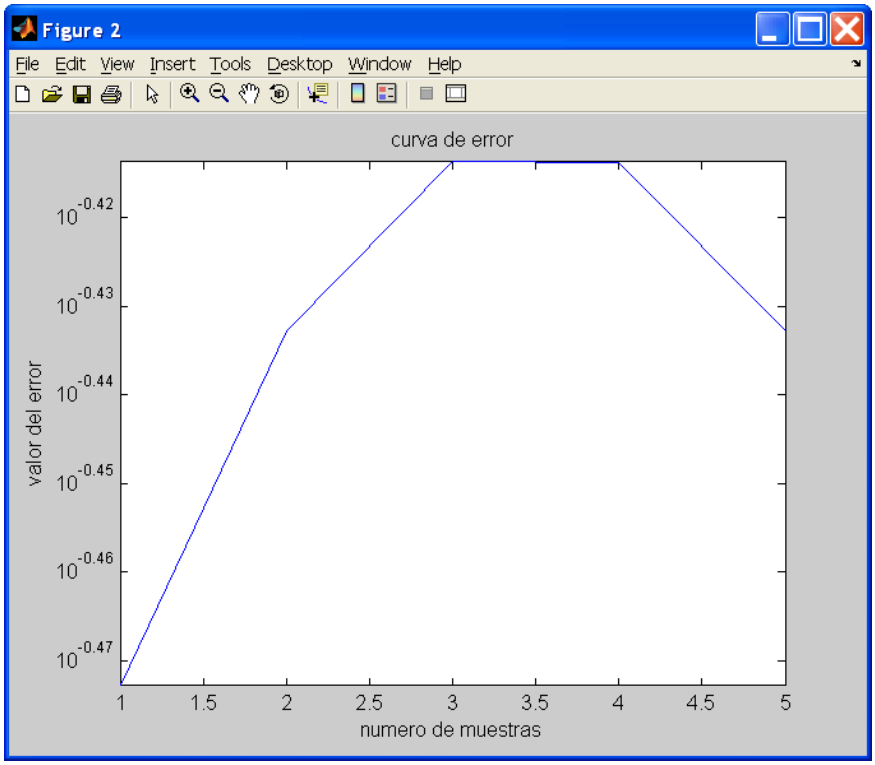
Pantalla 6.62 Ejemplo problema3_lms_z. Curva de error de una señal senosoidal en un algoritmo lms



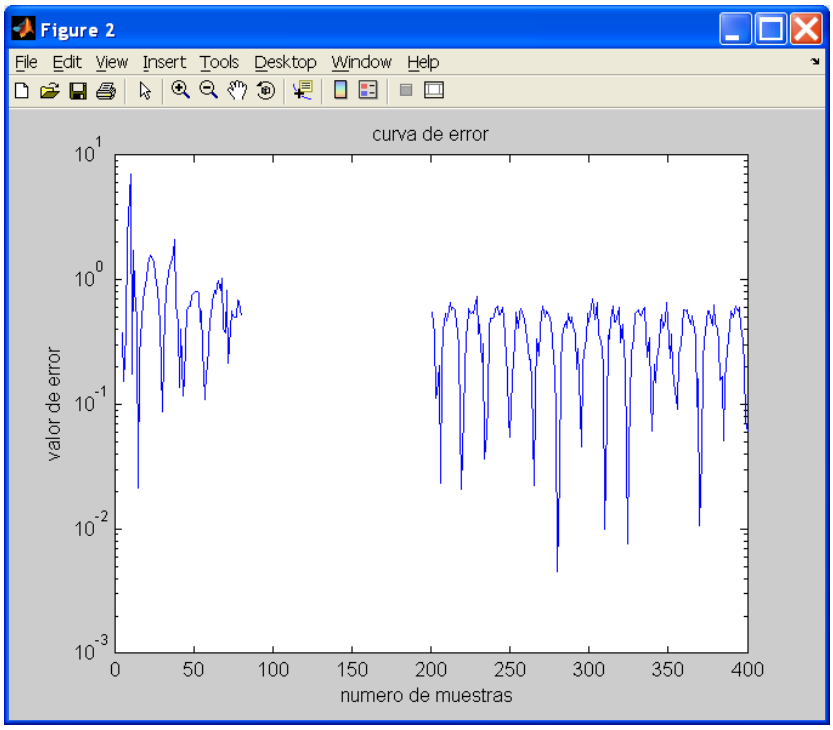
Pantalla 6.63 Ejemplo problema3_rls_z. Curva de error de una señal senosoidal en un algoritmo rls



Pantalla 6.64 Ejemplo problema3_cma_z. Curva de error de una señal senosoidal en un algoritmo cma



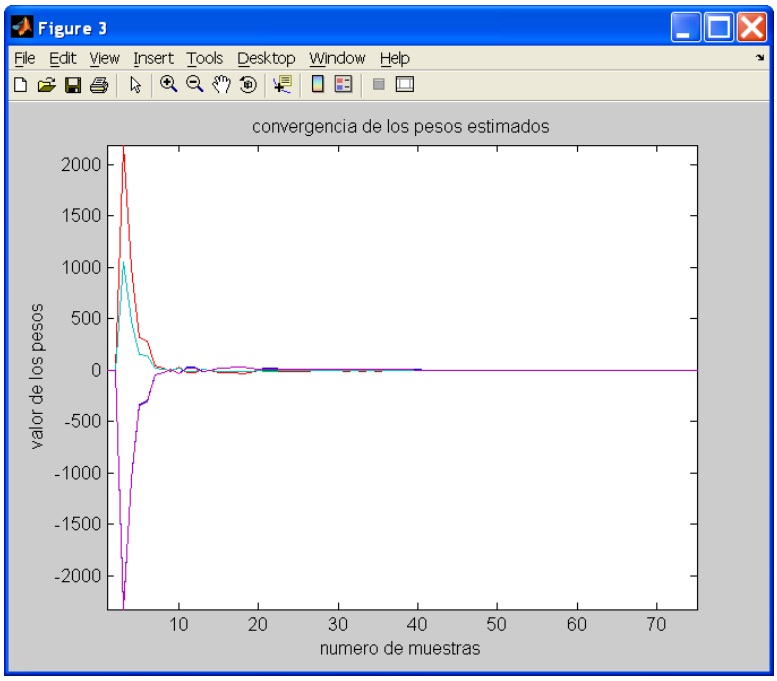
Pantalla 6.65 Ejemplo problema3_dmi_z. Curva de error de una señal senosoidal en un algoritmo dmi



el algoritmo lms no tiene grafico de convergencia de los pesos estimados los algoritmos cma y dmi sus graficos de convergencia de pesos estimados siempre dan como si hubiera una convergencia instantanea lo cual no siempre es cierto y puede dar datos erroneos sobre el desarrollo de la curva.

este grafico funciona mejor es en el algoritmo rls.

Pantalla 6.66 Ejemplo problema3_rls_z. Convergencia de los pesos estimados para una señal senosoidal en un algoritmo rls



Comparación de resultados entre los ejemplos problema2_lms_x, Problema2_rls_x, problema2_cma_x y problema2_dmi_x.

Pantalla 6.67 Datos del programa problema2_lms_x, problema2_rls_x, Problema2_cma_x y problema2_dmi_x

```
 ingrese el orden del filtro, cuyo coeficiente se va ha ajustar....5

orden =

     5

 ingrese el numero de antenas entre 1 y 15
 ingrese el numero de antenas entre 1,2,3,4,5,6,7,8,9,10,11,12,13,14 o 15....5
ingrese el voltaje1 (en v)..... voltaje1=0.09763
 ingrese la voltaje2 (en V).... voltaje2=0.28731
 ingrese la voltaje3 (en V).....voltaje3=0.335965
 ingrese la voltaje4 (en v)..... voltaje4=0.2209
 ingrese la voltaje5 (en v)..... voltaje5=0.0963
ingrese el ruido en la antena1.....ruido1=0.01
ingrese el ruido en la antena2.....ruido2=0.01
ingrese el ruido en la antena3.....ruido3=0.01
ingrese el ruido en la antena4.....ruido4=0.02
ingrese el ruido en la antena5.....ruido5=0.02
si desea que el ruido adicional vaya a la señal a la entrada digite 1
 de lo contrario se considerara que el ruido adicional se agregara a la señal de referencia
opcion_ruido=1
ingrese el error del sistema=0.0000001

h =

    0.0976
    0.2873
    0.3360
    0.2209
    0.0963
```

Pantalla 6.68 Potencias obtenidas con el programa problema2_lms_x

```
potencia =

    0.2997

potencia1 =

    0.0145

potencia2 =

    0.0912

potencia3 =

    0.1225

potencia4 =

    0.0609

potencia5 =

    0.0164
```

Pantalla 6.69 Potencias obtenidas con el programa problema2_rls_x

```
potencia =

    0.2624

potencia1 =

    0.0143

potencia2 =

    0.0913

potencia3 =

    0.1226

potencia4 =
```

0.0609

potencia5 =

0.0162

Pantalla 6.70 Potencias obtenidas con el programa problema2_cma_x

potencia =

0.3306

potencia1 =

0.0151

potencia2 =

0.0925

potencia3 =

0.1240

potencia4 =

0.0620

potencia5 =

0.0171

Pantalla 6.71 Potencias obtenidas con el programa problema2_dmi_x

potencia =

0.5111

potencia1 =

0.0198

potencia2 =

0.0153

potencia3 =

0.0067

potencia4 =

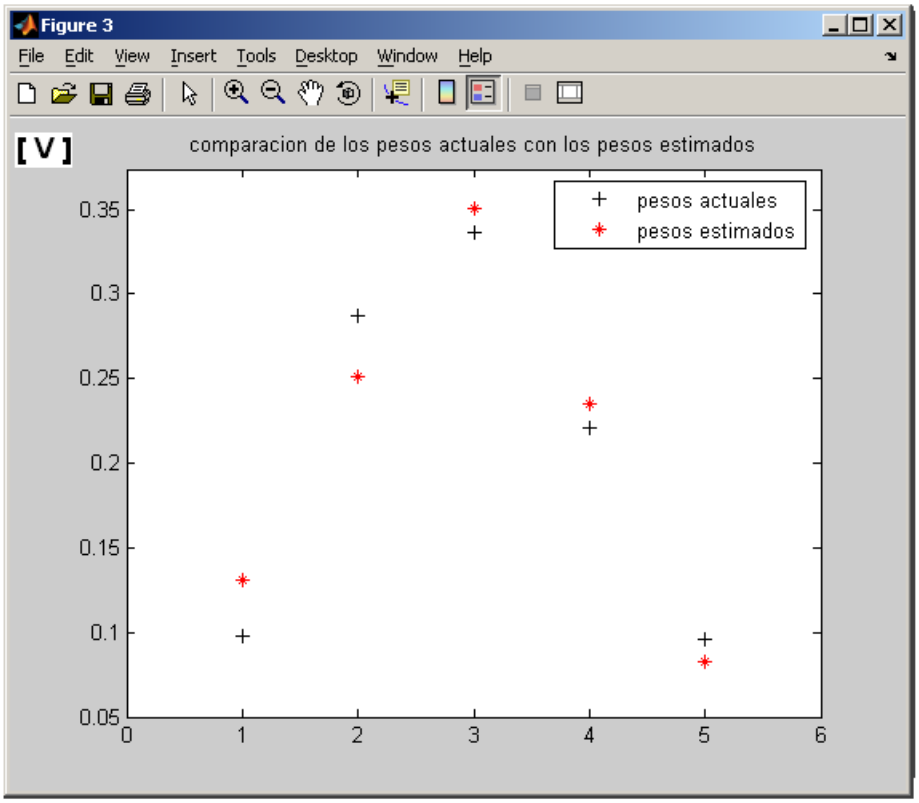
0.0116

potencia5 =

0.0223

Pantalla 6.72 Ejemplo problema2_lms_x. Comparación de los pesos estimados

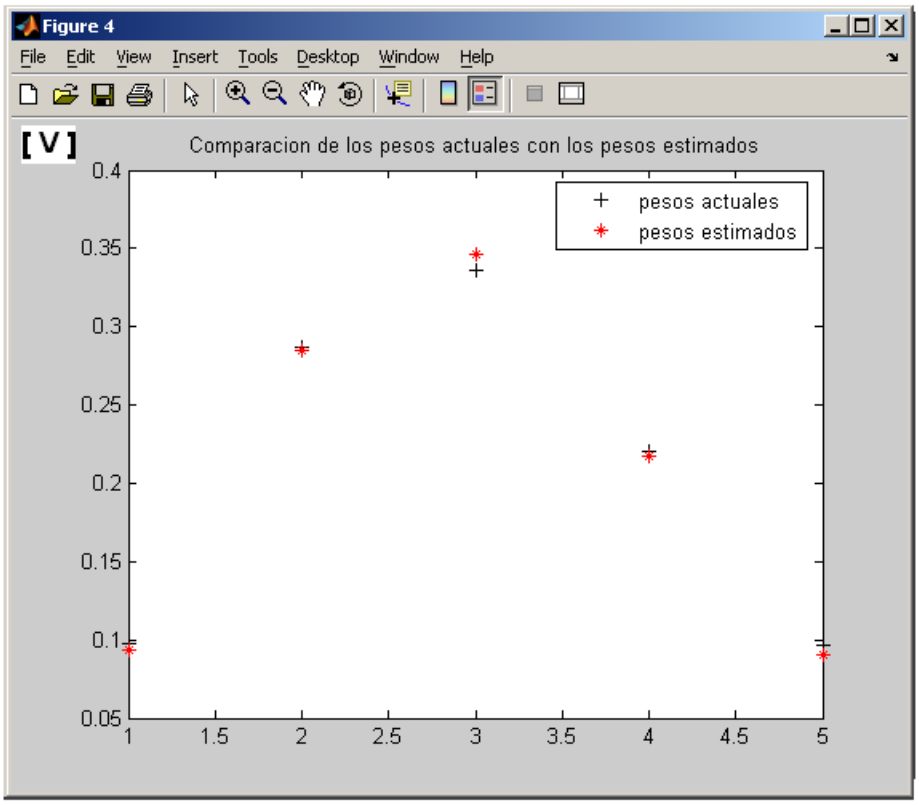
Ejemplo problema2_lms_x
comparacion de los pesos estimados de una señal gaussiana con los pesos reales o actuales de una señal gaussiana utilizando un algoritmo lms ,orden=5, numero pesos=5



Eje de las x constante, eje de las y (amplitud de la señal en voltios). Amplitud_señal vs Valor constante

Pantalla 6.73 Ejemplo problema2_rls_x. Comparación de los pesos estimados

Ejemplo problema2_rls_x
comparacion de los pesos estimados de una señal gaussiana con los pesos reales o actuales de una señal gaussiana utilizando un algoritmo rls,orden=5, numero pesos=5

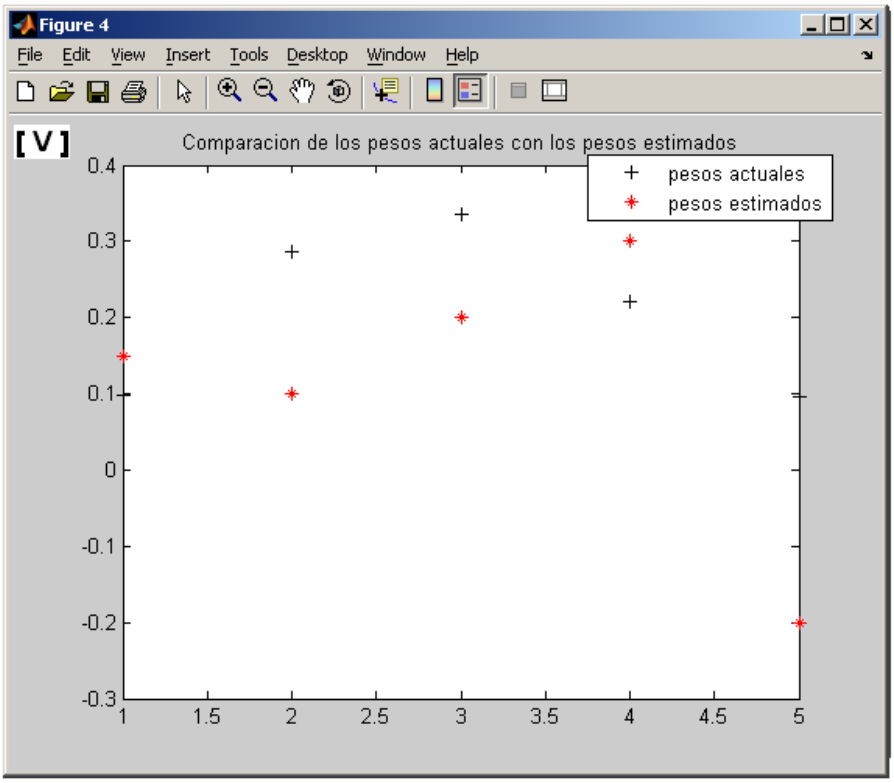


Eje de las x constante, eje de las y (amplitud de la señal en voltios). Amplitud_señal vs Valor constante

Se ve que al utilizar el algoritmo rls con valores gaussianos la convergencia es mejor y el error casi nulo.

Pantalla 6.74 Ejemplo problema2_cma_x. Comparación de los pesos estimados

Ejemplo problema2_cma_x
comparacion de los pesos estimados de una señal gaussiana con los pesos reales o actuales de una señal gaussiana utilizando un algoritmo cma,orden=5, numero pesos=5

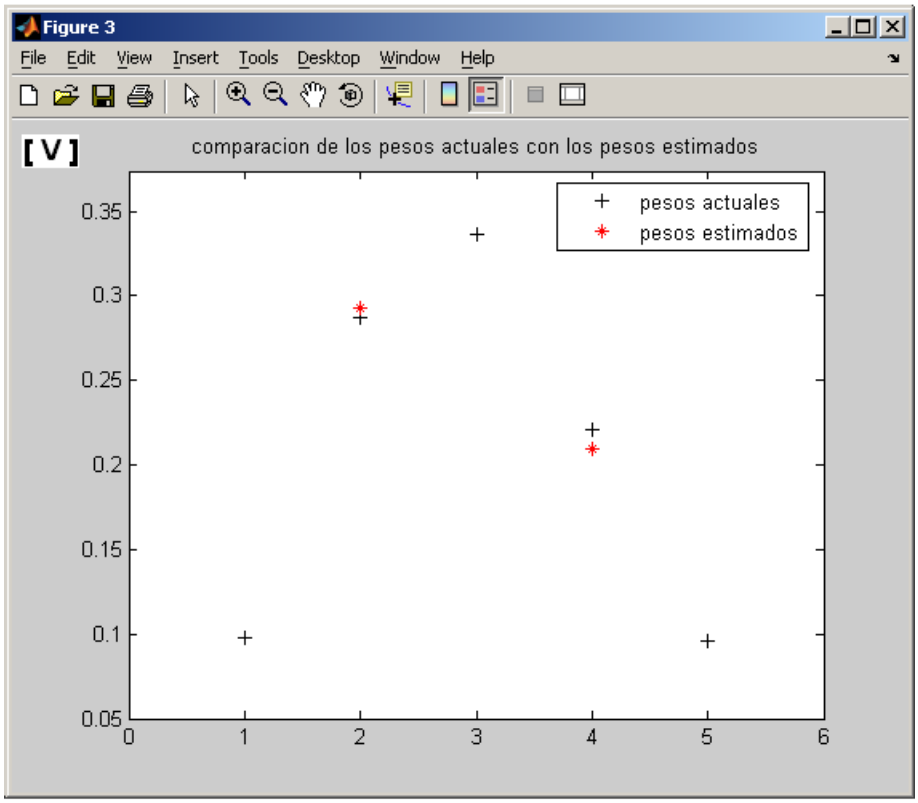


Eje de las x constante, eje de las y (amplitud de la señal en voltios). Amplitud_señal vs Valor constante

Vemos que la exactitud del programa al usar el algoritmo cma, disminuyo considerablemente en relación al uso del algoritmo lms y rls.

Pantalla 6.75 Ejemplo problema2_dmi_x. Comparación de los pesos estimados

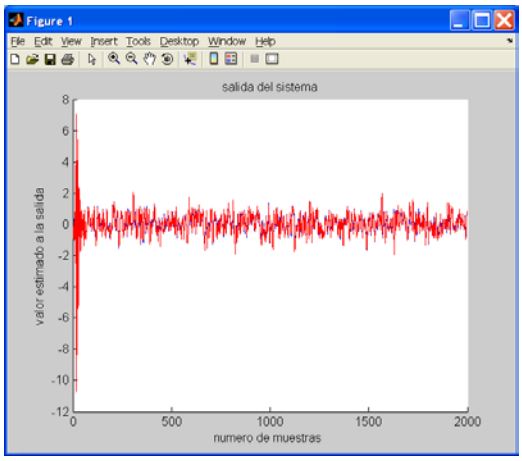
Ejemplo problema2_dmi_x
comparacion de los pesos estimados de una señal gaussiana con los pesos reales o actuales de una señal gaussiana utilizando un algoritmo dmi ,orden=5, numero pesos=5



Eje de las x constante, eje de las y (amplitud de la señal en voltios). Amplitud_señal vs Valor constante

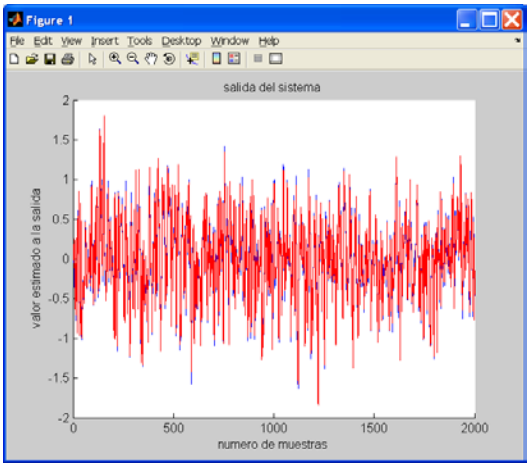
Pantalla 6.76 Ejemplo problema2_lms_x. Comparación entre la salida del sistema de una señal gaussiana (lms) con la señal de referencia del sistema

Señal de referencia color azul, señal obtenida por el algoritmo lms color rojo



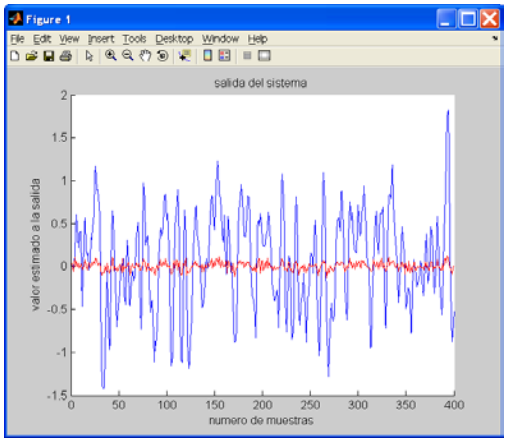
Pantalla 6.77 Ejemplo problema2_rls_x. Comparación entre la salida del sistema de una señal gaussiana (rls) con la señal de referencia del sistema

Señal de referencia color azul, señal obtenida por el algoritmo rls color rojo



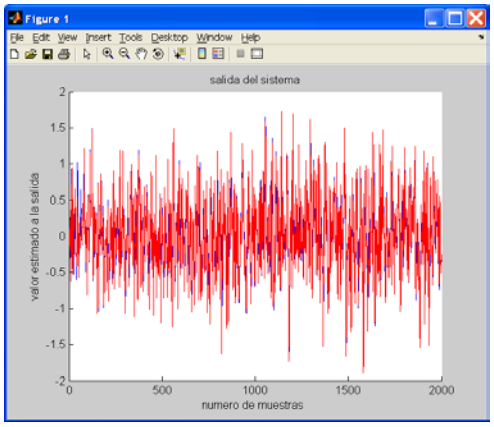
Pantalla 6.78 Ejemplo problema2_cma_x. Comparación entre la salida del sistema de una señal gaussiana (cma) con la señal de referencia del sistema.

Señal de referencia color azul, señal obtenida por el algoritmo cma color rojo

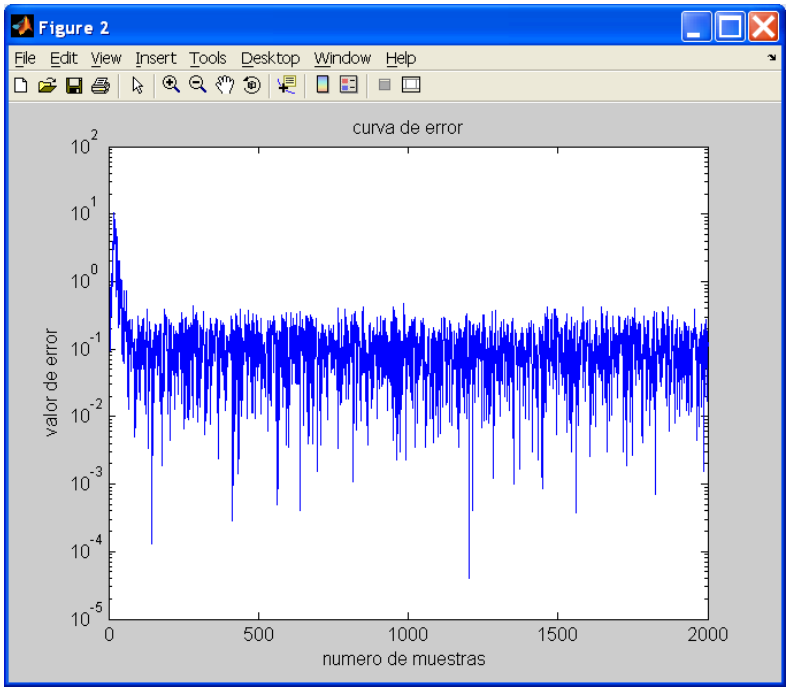


Pantalla 6.79 Ejemplo problema2_dmi_x. Comparación entre la salida del sistema de una señal gaussiana (dmi) con la señal de referencia del sistema

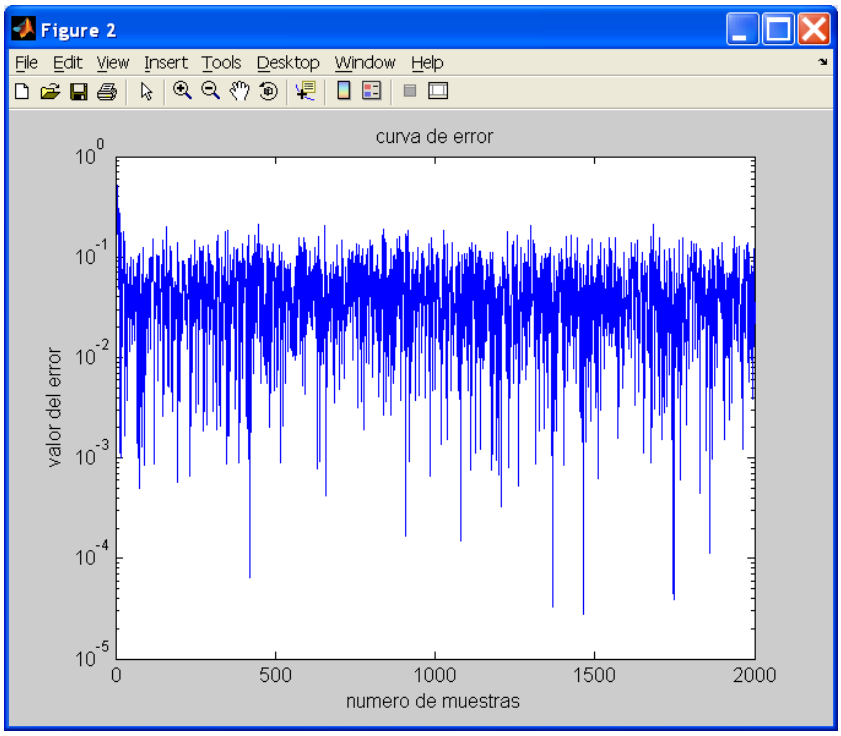
Señal de referencia color azul, señal obtenida por el algoritmo dmi color rojo



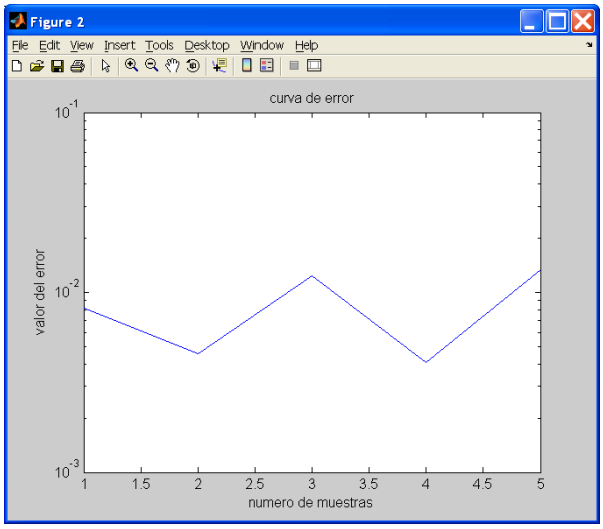
Pantalla 6.80 Ejemplo problema2_lms_x. Curva de error de una señal gaussiana en un algoritmo lms



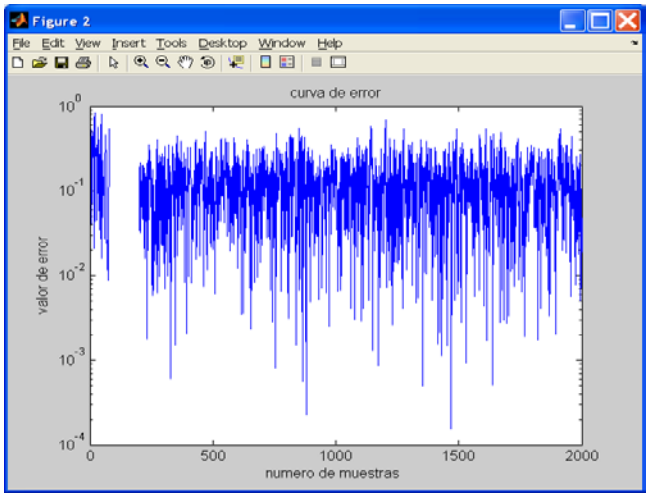
Pantalla 6.81 Ejemplo problema2_rls_x. Curva de error de una señal gaussiana en un algoritmo rls



Pantalla 6.82 Ejemplo problema2_cma_x. Curva de error de una señal gaussiana en un algoritmo cma



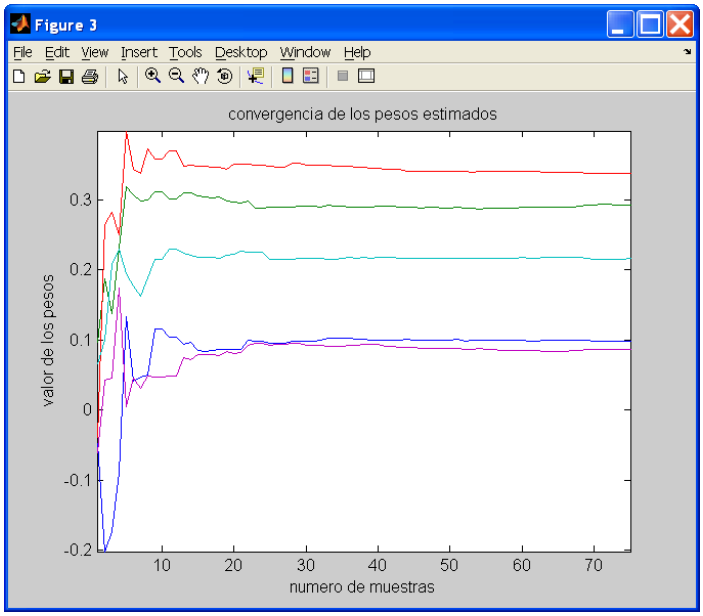
Pantalla 6.83 Ejemplo problema2_dmi_x. Curva de error de una señal gaussiana en un algoritmo dmi



el algoritmo lms no tiene grafico de convergencia de los pesos estimados los algoritmos cma y dmi sus graficos de convergencia de pesos estimados siempre dan como si hubiera una convergencia instantanea lo cual no siempre es cierto y puede dar datos erroneos sobre el desarrollo de la curva.

este grafico funciona mejor es en el algoritmo rls.

Pantalla 6.84 Ejemplo problema2_rls_x. Convergencia de los pesos estimados para una señal gaussiana en un algoritmo rls



Comparación de resultados entre los ejemplos problema1_lms_x, Problema1_rls_x, problema1_cma_x y problema1_dmi_x

Pantalla 6.85 Datos del programa problema1_lms_x, problema1_rls_x, Problema1_cma_x y problema1_dmi_x

```
 ingrese el orden del filtro, cuyo coeficiente se va ha ajustar....5

orden =

     5

 ingrese el numero de antenas entre 1 y 5
 ingrese el numero de antenas entre 1,2,3,4 o 5....5
ingrese el voltaje1 (en v)..... voltaje1=0.20
 ingrese la voltaje2 (en V).... voltaje2=0.25
 ingrese la voltaje3 (en V).....voltaje3=0.21
 ingrese la voltaje4 (en v)..... voltaje4=0.18
 ingrese la voltaje5 (en v)..... voltaje5=0.15
ingrese el ruido en la antena1.....ruido1=0.01
ingrese el ruido en la antena2.....ruido2=0.01
ingrese el ruido en la antena3.....ruido3=0.01
ingrese el ruido en la antena4.....ruido4=0.02
ingrese el ruido en la antena5.....ruido5=0.02
si desea que el ruido adicional vaya a la señal a la entrada digite 1
 de lo contrario se considerara que el ruido adicional se agregara a la señal de referencia
opcion_ruido=1
ingrese el error del sistema=0.0000001
```

Pantalla 6.86 Potencias obtenidas con el programa problema1_lms_x

```
potencia =

    0.5027

potencia1 =

    0.0478

potencia2 =

    0.0710

potencia3 =

    0.0520

potencia4 =

    0.0437

potencia5 =

    0.0328
```

Pantalla 6.87 Potencias obtenidas con el programa problema1_rls_x

```
potencia =

    0.4971

potencia1 =

    0.0491

potencia2 =

    0.0726

potencia3 =

    0.0534
```

```
potencia4 =

    0.0450

potencia5 =

    0.0339
```

Pantalla 6.88 Potencias obtenidas con el programa problema1_cma_x

```
potencia =

    0.4986

potencia1 =

    0.0498

potencia2 =

    0.0735

potencia3 =

    0.0542

potencia4 =

    0.0457

potencia5 =

    0.0345
```

Pantalla 6.89 Potencias obtenidas con el programa problema1_dmi_x

```
potencia =

    0.5002

potencia1 =
```

0.0493

potencia2 =

0.0728

potencia3 =

0.0536

potencia4 =

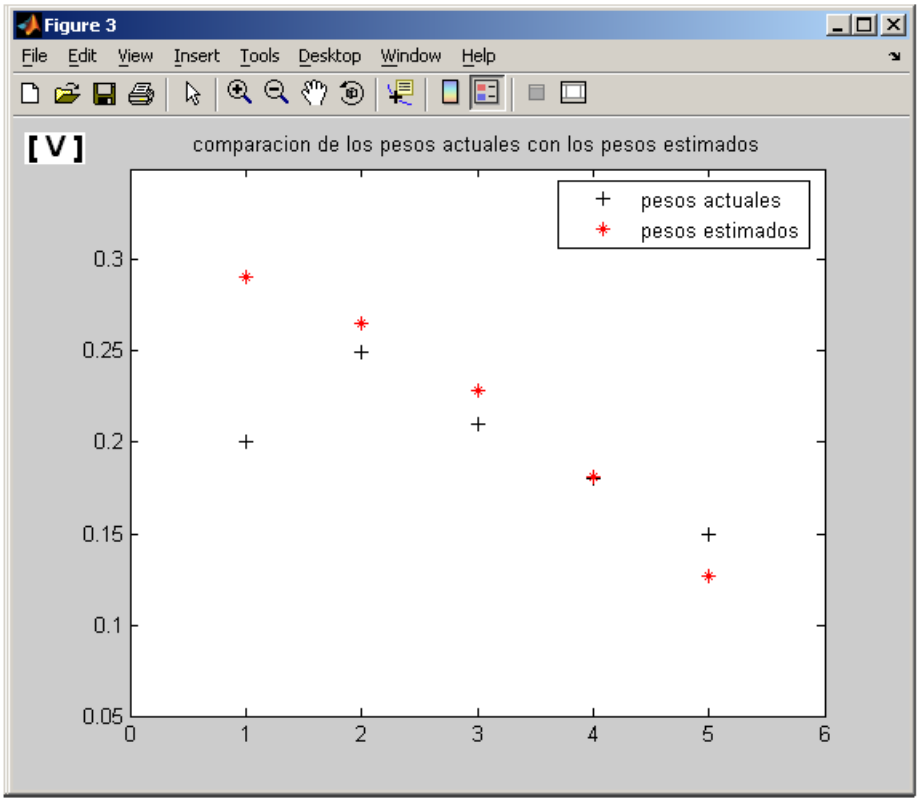
0.0451

potencia5 =

0.0340

Pantalla 6.90 Ejemplo problema1_lms_x. Comparación de los pesos estimados

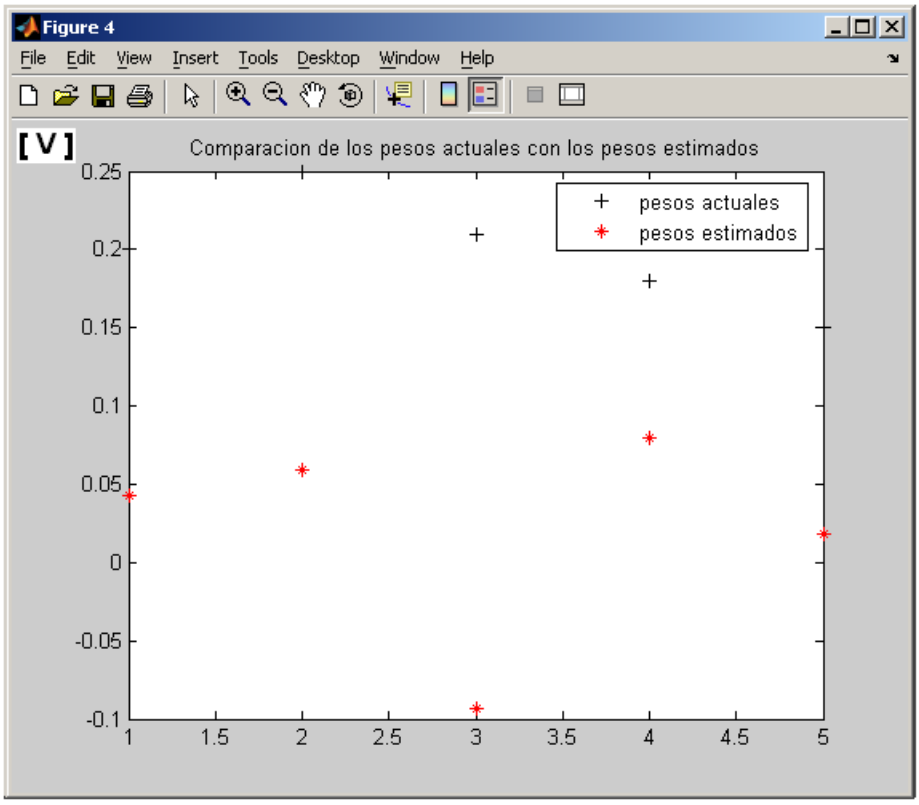
Ejemplo problema1_lms_x
comparacion de los pesos estimados de una señal senosoidal con los pesos reales o actuales de una señal senosoidal utilizando un algoritmo lms ,orden=5, numero pesos=5



Eje de las x constante, eje de las y (amplitud de la señal en voltios). Amplitud_señal vs Valor constante

Pantalla 6.91 Ejemplo problema1_rls_x. Comparación de los pesos estimados

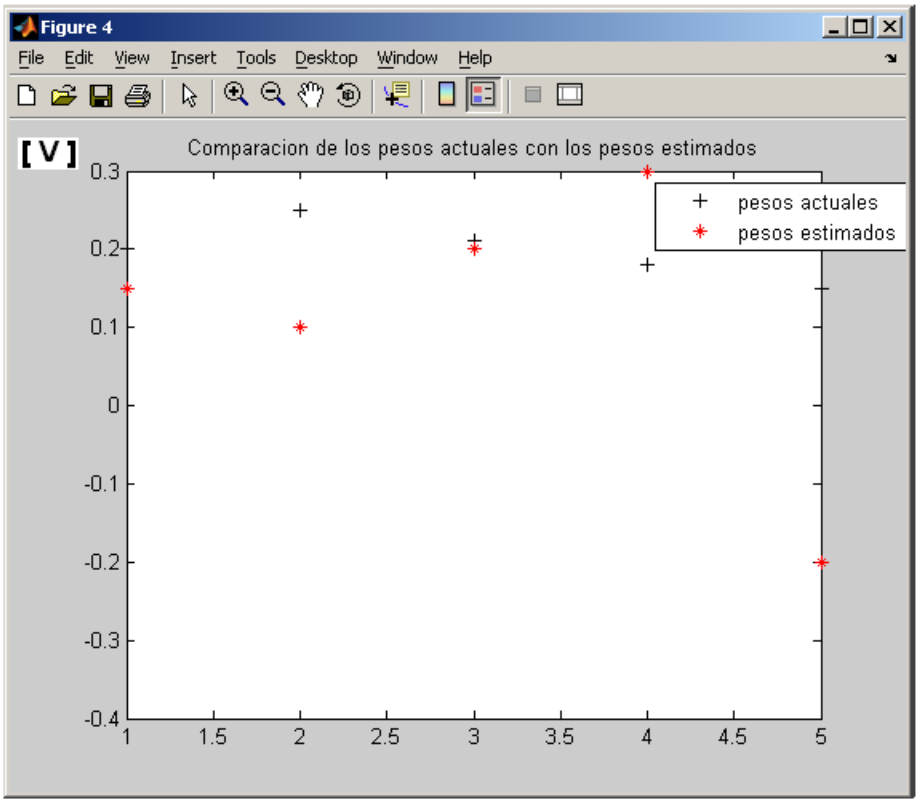
Ejemplo problema1_rls _x
comparacion de los pesos estimados de una señal senosoidal con los pesos reales o actuales de una señal senosoidal utilizando un algoritmo rls ,orden=5, numero pesos=5



Eje de las x constante, eje de las y (amplitud de la señal en voltios). Amplitud_señal vs Valor constante

Pantalla 6.92 Ejemplo problema1_cma_x. Comparación de los pesos estimados

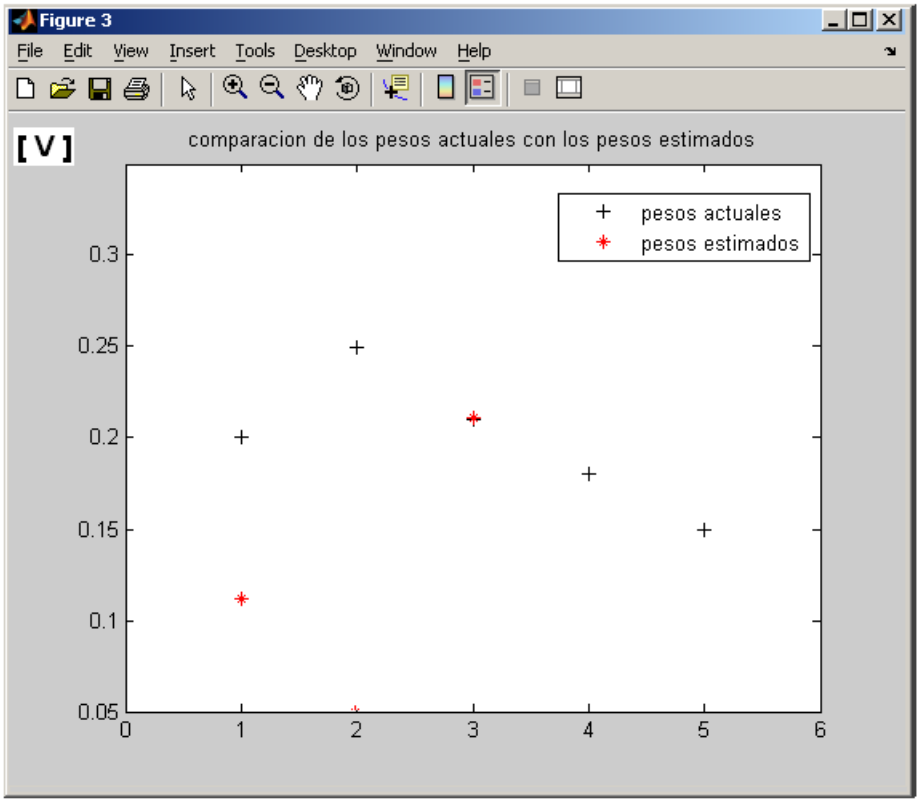
Ejemplo problema1_ cma _x
comparacion de los pesos estimados de una señal senosoidal con los pesos reales o actuales de una señal senosoidal utilizando un algoritmo cma,orden=5, numero pesos=5



Eje de las x constante, eje de las y (amplitud de la señal en voltios). Amplitud_señal vs Valor constante

Pantalla 6.93 Ejemplo problema1_dmi_x. Comparación de los pesos estimados

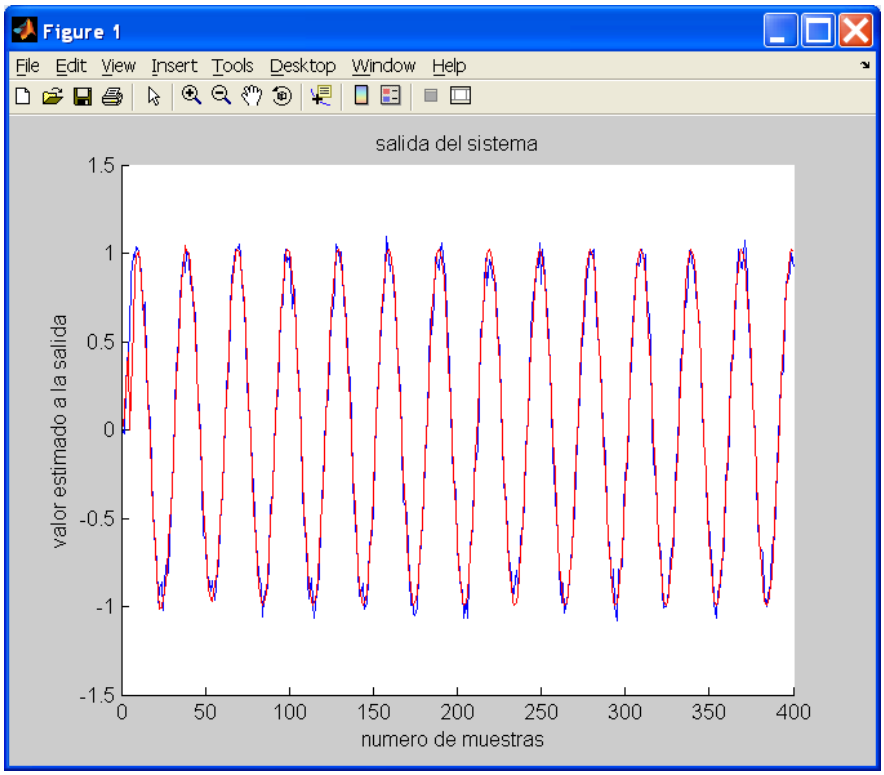
Ejemplo problema1_dmi_x
comparacion de los pesos estimados de una señal senosoidal con los pesos reales o actuales de una señal senosoidal utilizando un algoritmo dmi,orden=5, numero pesos=5



Eje de las x constante, eje de las y (amplitud de la señal en voltios). Amplitud_señal vs Valor constante

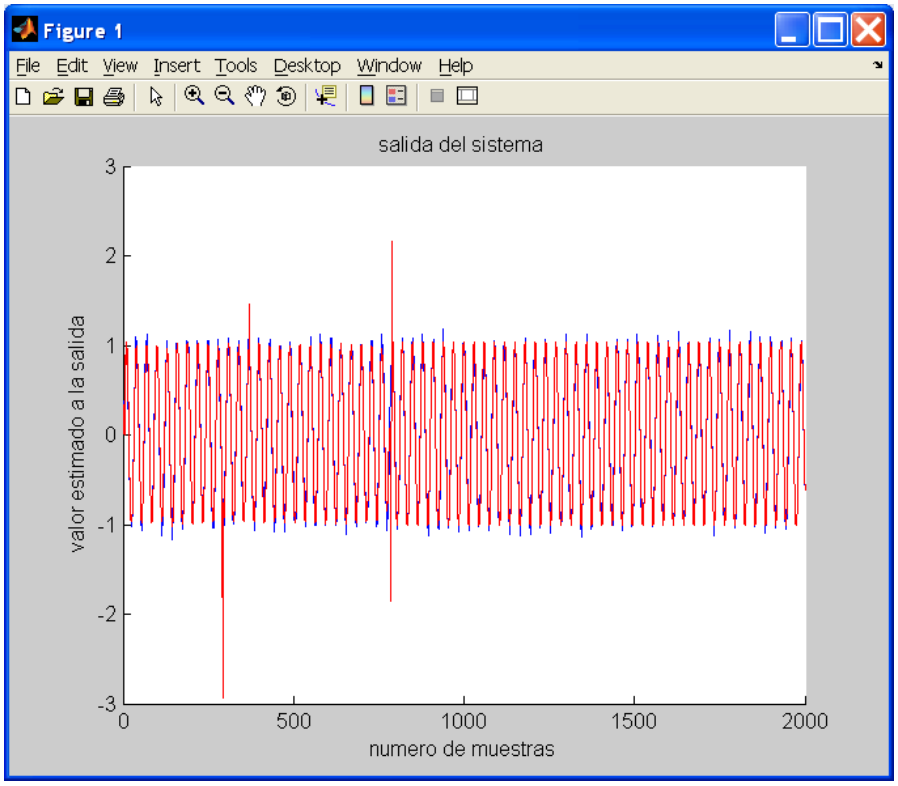
Pantalla 6.94 Ejemplo problema1_lms_x. Comparación entre la salida del sistema de una señal senosoidal (lms) con la señal de referencia del sistema

Señal de referencia color azul, señal obtenida por el algoritmo lms color rojo



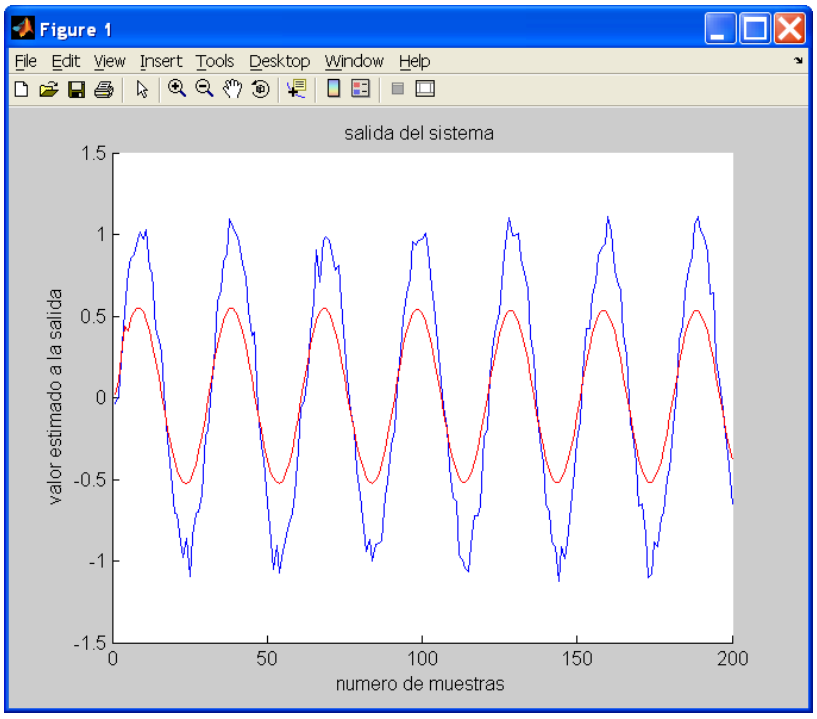
Pantalla 6.95 Ejemplo problema1_rls_x. Comparación entre la salida del sistema de una señal senosoidal (rls) con la señal de referencia del sistema

Señal de referencia color azul, señal obtenida por el algoritmo rls color rojo



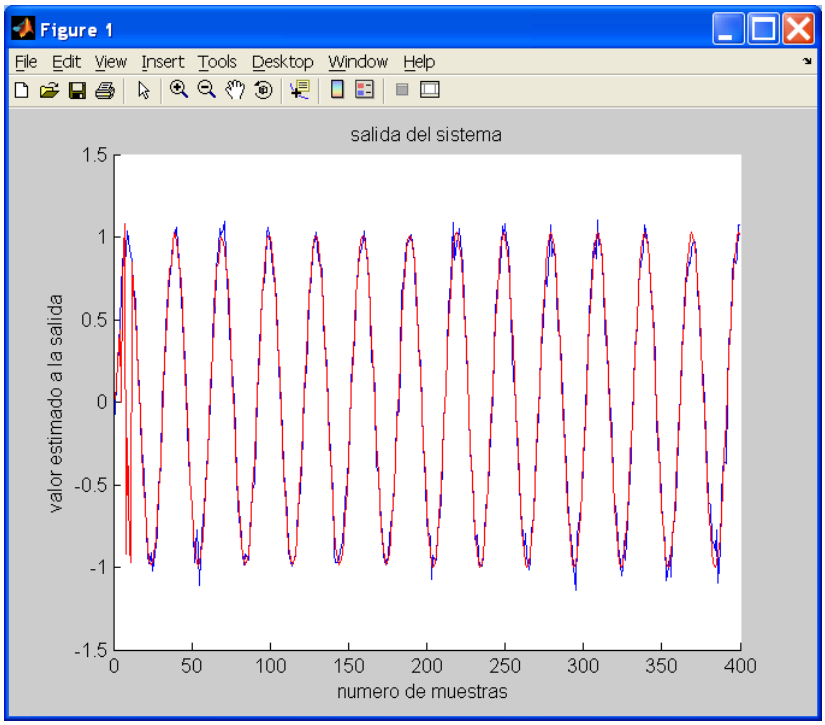
Pantalla 6.96 Ejemplo problema1_cma_x. Comparación entre la salida del sistema de una señal senosoidal (cma) con la señal de referencia del sistema.

Señal de referencia color azul, señal obtenida por el algoritmo cma color rojo

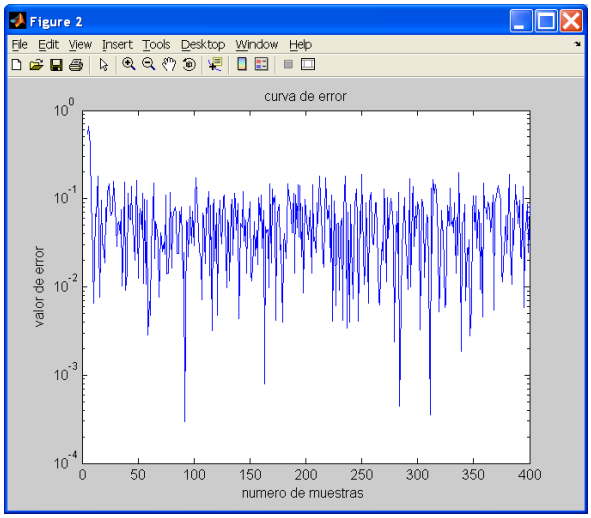


Pantalla 6.97 Ejemplo problema1_dmi_x. Comparación entre la salida del sistema de una señal senosoidal (dmi) con la señal de referencia del sistema

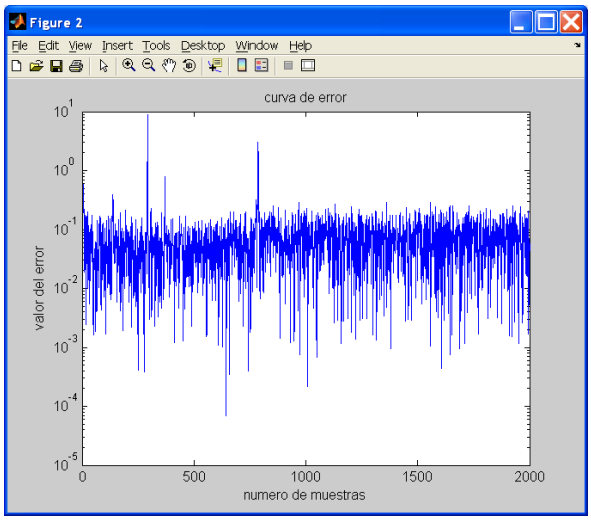
Señal de referencia color azul, señal obtenida por el algoritmo dmi color rojo



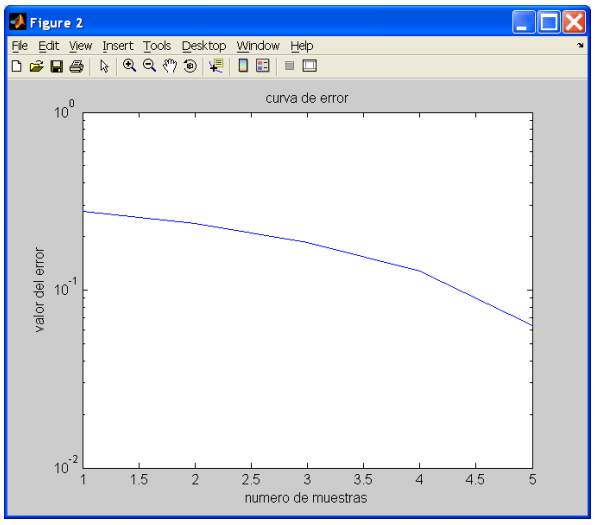
Pantalla 6.98 Ejemplo problema1_lms_x. Curva de error de una señal senosoidal en un algoritmo lms



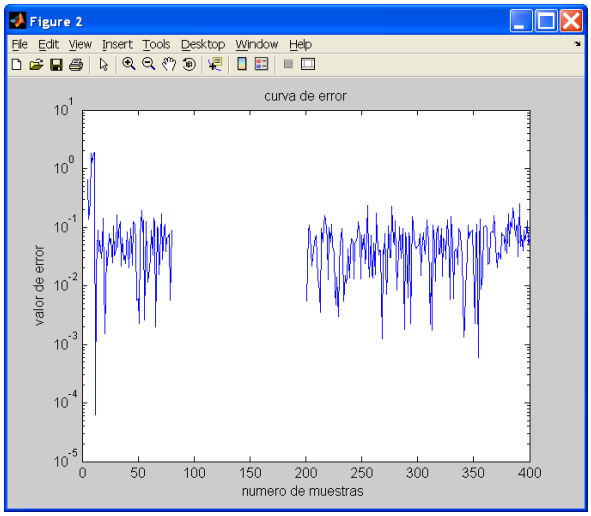
Pantalla 6.99 Ejemplo problema1_rls_x. Curva de error de una señal senosoidal en un algoritmo rls



Pantalla 6.100 Ejemplo problema1_cma_x. Curva de error de una señal senosoidal en un algoritmo cma



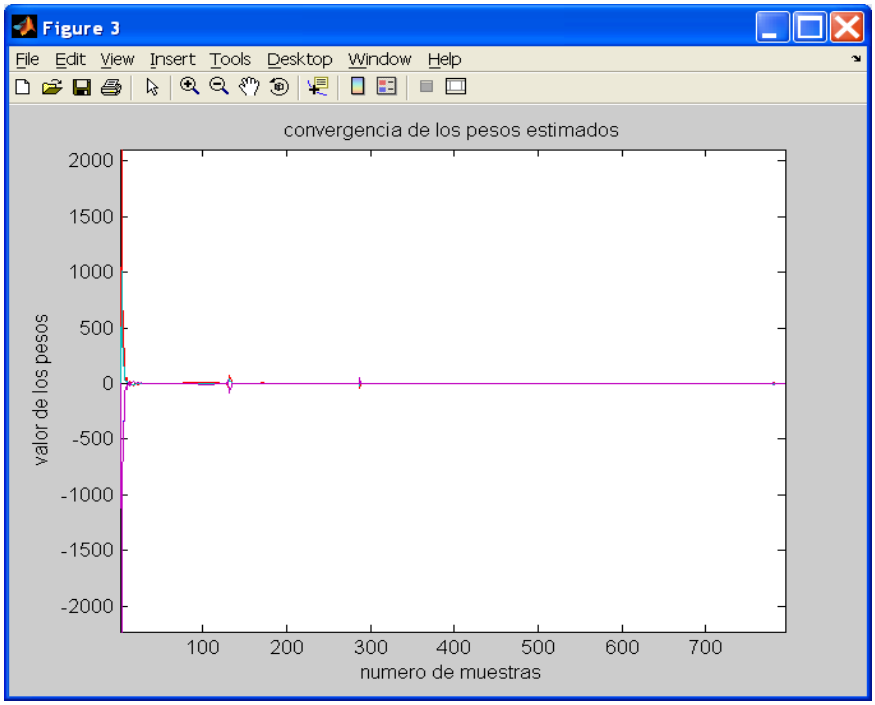
Pantalla 6.101 Ejemplo problema1_dmi_x. Curva de error de una señal senosoidal en un algoritmo dmi



el algoritmo lms no tiene grafico de convergencia de los pesos estimados los algoritmos cma y dmi sus graficos de convergencia de pesos estimados siempre dan como si hubiera una convergencia instantanea lo cual no siempre es cierto y puede dar datos erroneos sobre el desarrollo de la curva.

este grafico funciona mejor es en el algoritmo rls.

Pantalla 6.102 Ejemplo problema1_rls_x. Convergencia de los pesos estimados para una señal senosoidal en un algoritmo rls



Comparación de resultados entre los ejemplos problema1_lms_z, Problema1_rls_z, problema1_cma_z y problema1_dmi_z

Pantalla 6.103 Datos del programa problema1_lms_z, problema1_rls_z, Problema1_cma_z y problema1_dmi_z

```
 ingrese el orden del filtro, cuyo coeficiente se va ha ajustar....5

orden =

     5

 ingrese el numero de antenas entre 1 y 15
 ingrese el numero de antenas entre 1,2,3,4,5,6,7,8,9,10,11,12,13,14 o 15....5
ingrese el voltaje1 (en v)..... voltaje1=0.097631
 ingrese la voltaje2 (en V).... voltaje2=0.287310
 ingrese la voltaje3 (en V).....voltaje3=0.335965
 ingrese la voltaje4 (en v)..... voltaje4=0.2209
 ingrese la voltaje5 (en v)..... voltaje5=0.0963
ingrese el ruido en la antena1.....ruido1=0.01
ingrese el ruido en la antena2.....ruido2=0.01
ingrese el ruido en la antena3.....ruido3=0.01
ingrese el ruido en la antena4.....ruido4=0.01
ingrese el ruido en la antena5.....ruido5=0.02
si desea que el ruido adicional vaya a la señal a la entrada digite 1
 de lo contrario se considerara que el ruido adicional se agregara a la señal de referencia
opcion_ruido=1
ingrese el error del sistema=0.0000000000001

h =

    0.0976
    0.2873
    0.3360
    0.2209
    0.0963
```

Pantalla 6.104 Potencias obtenidas con el programa problema1_lms_z

```
potencia =

    0.5027

potencia1 =

    0.0478

potencia2 =

    0.0710

potencia3 =

    0.0520

potencia4 =

    0.0437

potencia5 =

    0.0328
```

Pantalla 6.105 Potencias obtenidas con el programa problema1_rls_z

```
potencia =

    0.4971

potencia1 =

    0.0491

potencia2 =

    0.0726

potencia3 =

    0.0534

potencia4 =
```

0.0450

potencia5 =

0.0339

Pantalla 6.106 Potencias obtenidas con el programa problema1_cma_z

potencia =

0.4986

potencia1 =

0.0498

potencia2 =

0.0735

potencia3 =

0.0542

potencia4 =

0.0457

potencia5 =

0.0345

Pantalla 6.107 Potencias obtenidas con el programa problema1_dmi_z

potencia =

0.5002

potencia1 =

0.0493

potencia2 =

0.0728

potencia3 =

0.0536

potencia4 =

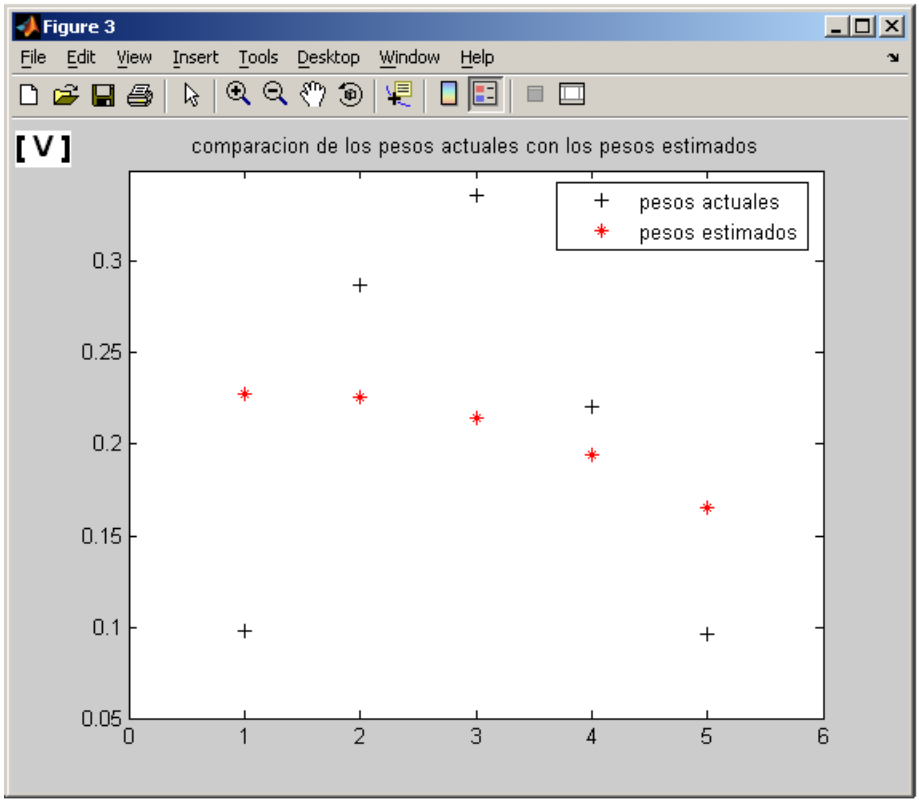
0.0451

potencia5 =

0.0340

Pantalla 6.108 Ejemplo problema1_lms_z. Comparación de los pesos estimados

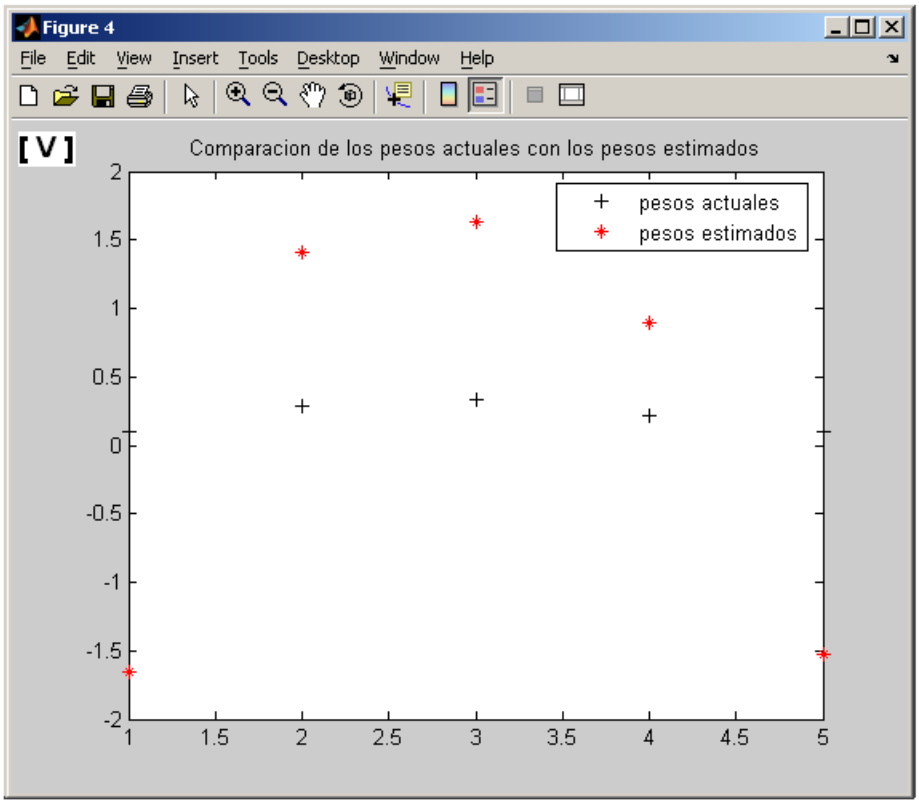
Ejemplo problema1_lms_z
comparacion de los pesos estimados de una señal senosoidal con los pesos reales o actuales de una señal senosoidal utilizando un algoritmo lms ,orden=5, numero pesos=5



Eje de las x constante, eje de las y (amplitud de la señal en voltios). Amplitud_señal vs Valor constante

Pantalla 6.109 Ejemplo problema1_rls_z. Comparación de los pesos estimados

Ejemplo problema1_rls _z
comparacion de los pesos estimados de una señal senosoidal con los pesos reales o actuales de una señal senosoidal utilizando un algoritmo rls ,orden=5, numero pesos=5

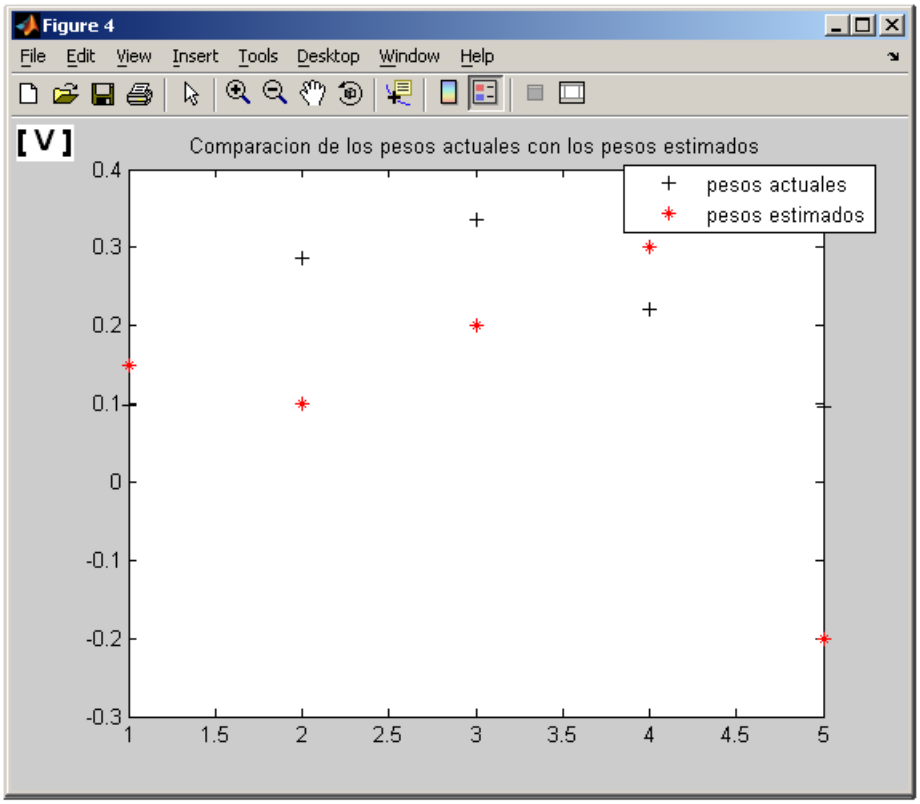


Eje de las x constante, eje de las y (amplitud de la señal en voltios). Amplitud_señal vs Valor constante

En este ejemplo vemos que la convergencia del algoritmo rls fue mejor que en el lms, también nos hemos dado cuenta que el algoritmo rls trabaja mucho mejor para valores gaussianos discretos.

Pantalla 6.110 Ejemplo problema1_cma_z. Comparación de los pesos estimados

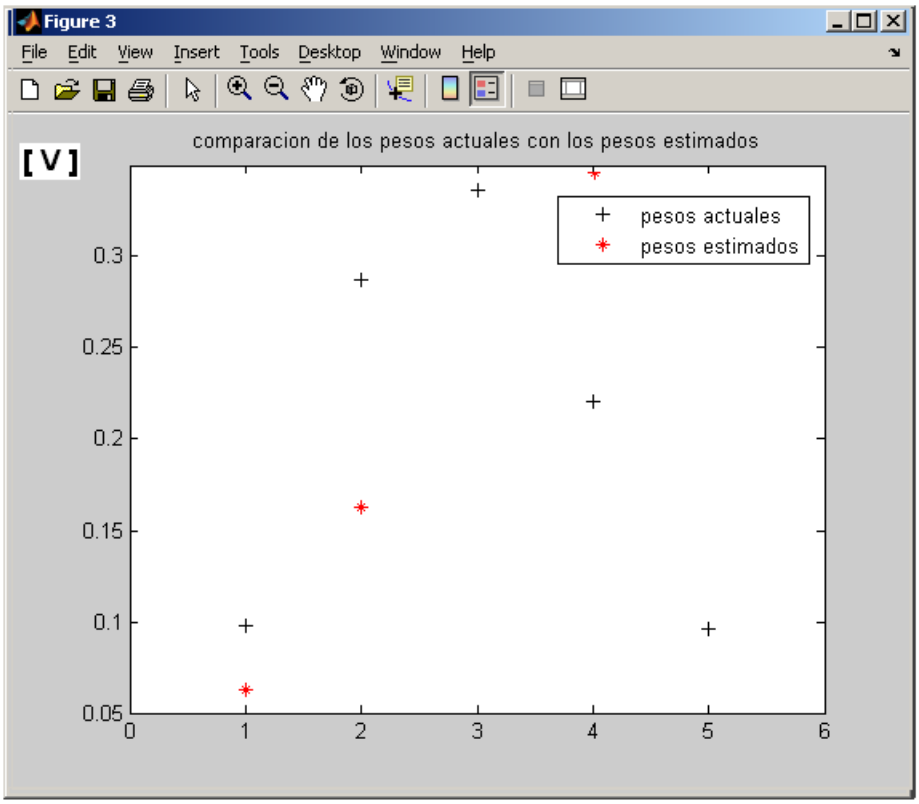
Ejemplo problema1_cma_z
comparacion de los pesos estimados de una señal senosoidal con los pesos reales o actuales de una señal senosoidal utilizando un algoritmo cma,orden=5, numero pesos=5



Eje de las x constante, eje de las y (amplitud de la señal en voltios). Amplitud_señal vs Valor constante

Pantalla 6.111 Ejemplo problema1_dmi_z. Comparación de los pesos estimados

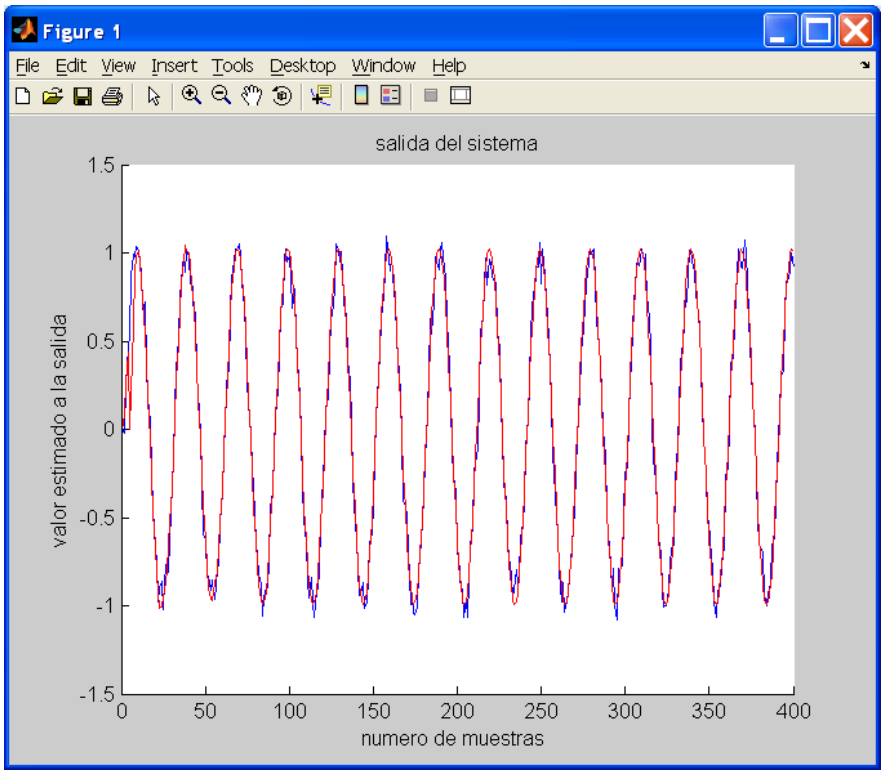
Ejemplo problema1_dmi_z
comparacion de los pesos estimados de una señal senosoidal con los pesos reales o actuales de una señal senosoidal utilizando un algoritmo dmi,orden=5, numero pesos=5



Eje de las x constante, eje de las y (amplitud de la señal en voltios). Amplitud_señal vs Valor constante

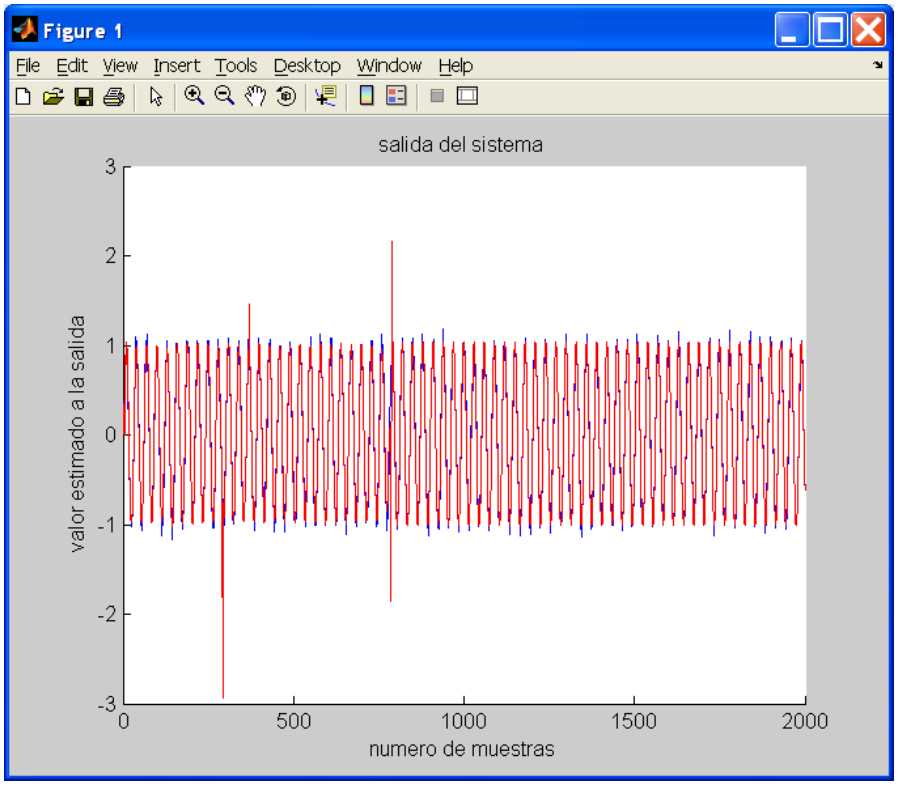
Pantalla 6.112 Ejemplo problema1_lms_z. Comparación entre la salida del sistema de una señal senosoidal (lms) con la señal de referencia del sistema

Señal de referencia color azul, señal obtenida por el algoritmo lms color rojo



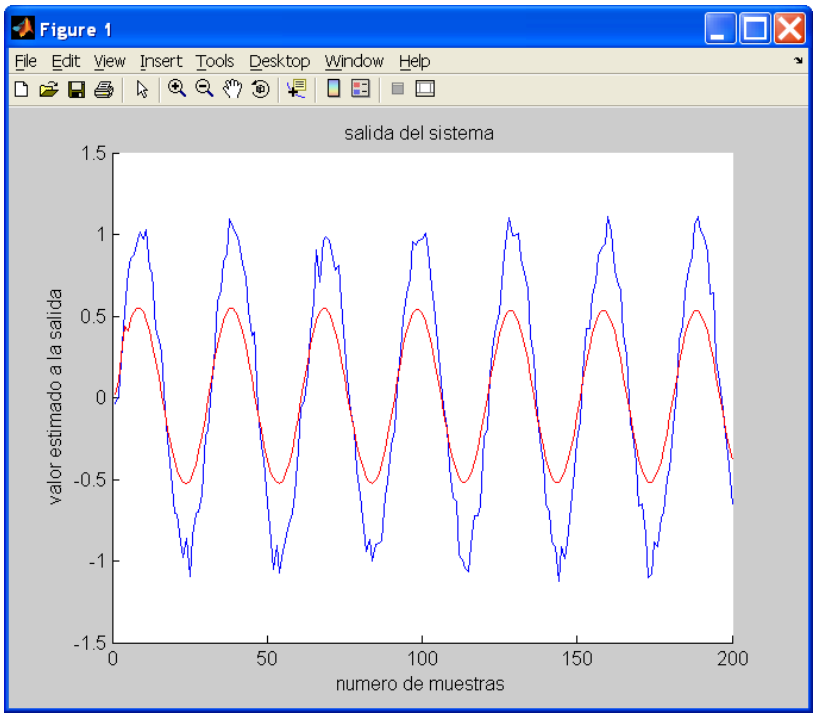
Pantalla 6.113 Ejemplo problema1_rls_z. Comparación entre la salida del sistema de una señal senosoidal (rls) con la señal de referencia del sistema

Señal de referencia color azul, señal obtenida por el algoritmo rls color rojo



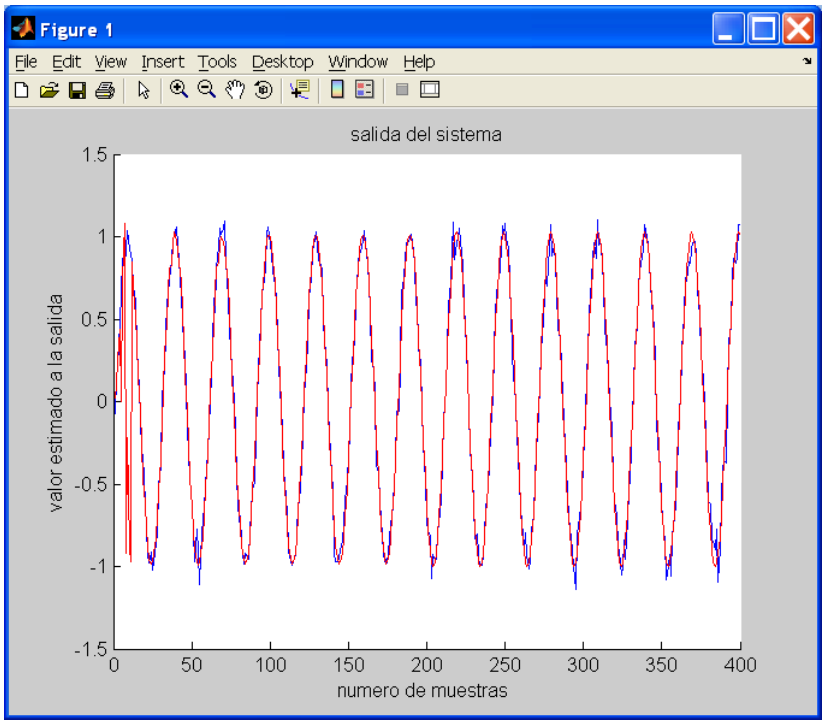
Pantalla 6.114 Ejemplo problema1_cma_z. Comparación entre la salida del sistema de una señal senosoidal (cma) con la señal de referencia del sistema.

Señal de referencia color azul, señal obtenida por el algoritmo cma color rojo

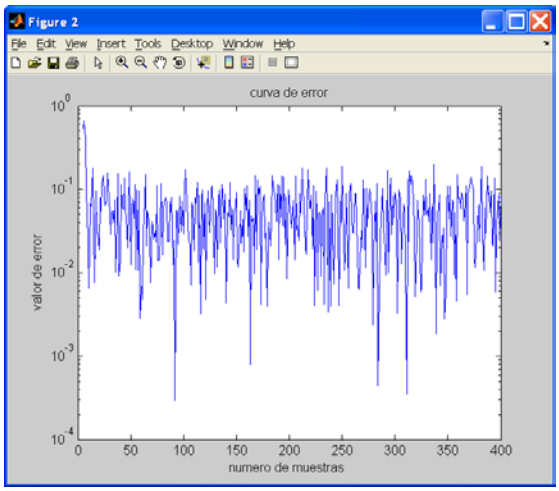


Pantalla 6.115 Ejemplo problema1_dmi_z. Comparación entre la salida del sistema de una señal senosoidal (dmi) con la señal de referencia del sistema

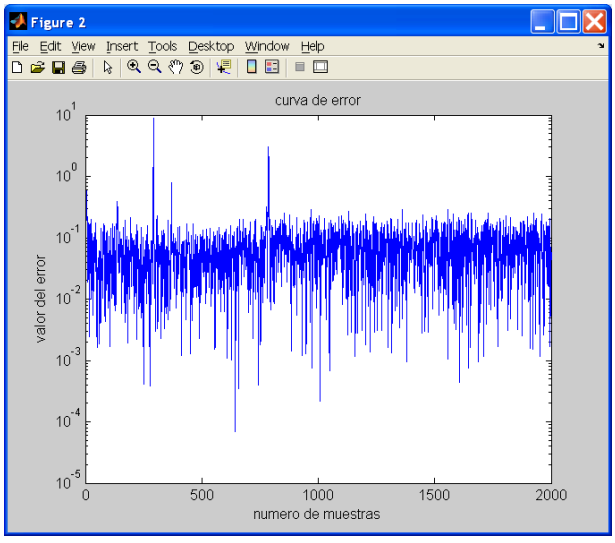
Señal de referencia color azul, señal obtenida por el algoritmo dmi color rojo



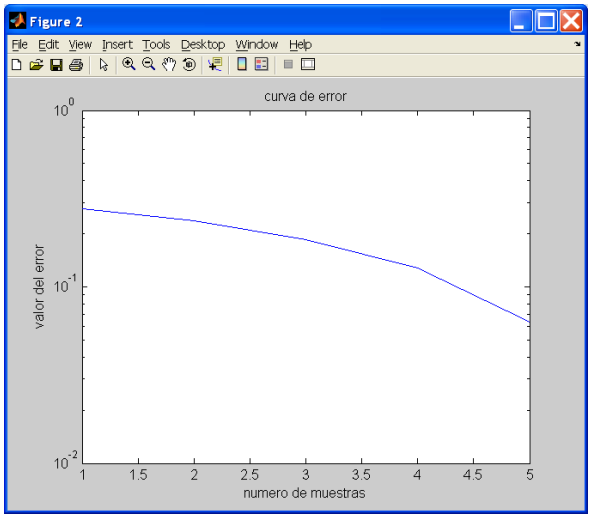
Pantalla 6.116 Ejemplo problema1_lms_z. Curva de error de una señal senosoidal en un algoritmo lms



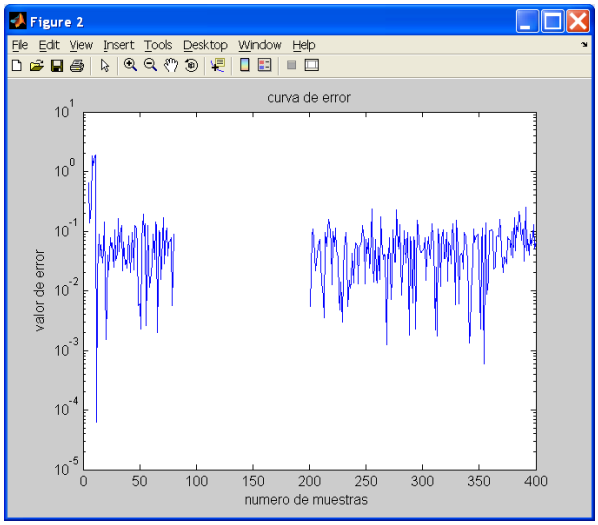
Pantalla 6.117 Ejemplo problema1_rls_z. Curva de error de una señal senosoidal en un algoritmo rls



Pantalla 6.118 Ejemplo problema1_cma_z. Curva de error de una señal senosoidal en un algoritmo cma



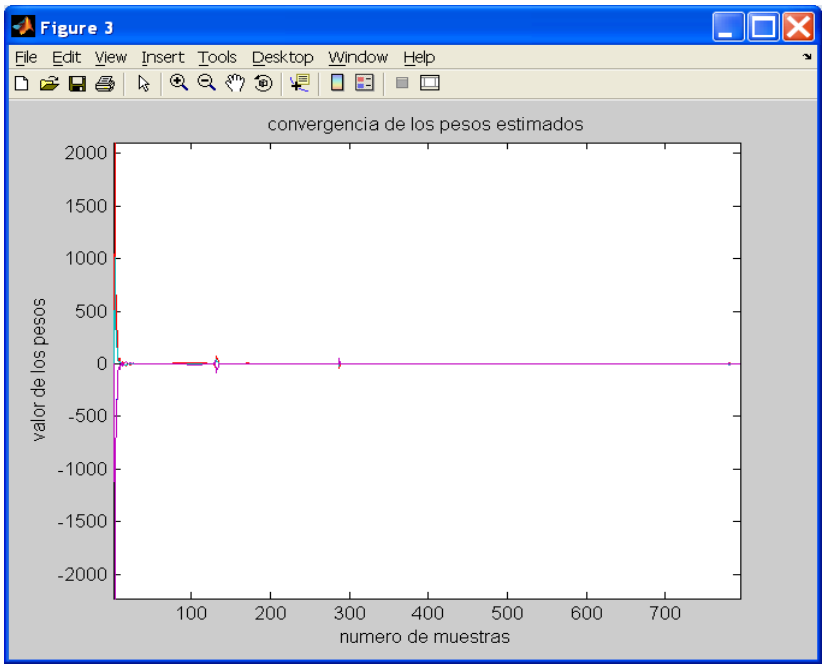
Pantalla 6.119 Ejemplo problema1_dmi_z. Curva de error de una señal senosoidal en un algoritmo dmi



el algoritmo lms no tiene grafico de convergencia de los pesos estimados los algoritmos cma y dmi sus graficos de convergencia de pesos estimados siempre dan como si hubiera una convergencia instantanea lo cual no siempre es cierto y puede dar datos erroneos sobre el desarrollo de la curva.

este grafico funciona mejor es en el algoritmo rls.

Pantalla 6.120 Ejemplo problema1_rls_x. Convergencia de los pesos estimados para una señal senosoidal en un algoritmo rls



ANEXO G

PROGRAMAS EN MATLAB DE LOS ALGORITMOS LMS, RLS, CMA, DMI

Problema1_lms, Problema1_rls , Problema1_cma y Problema1_dmi se usa una señal senosoidal de entrada, se usa la misma señal senosoidal como señal de referencia.

Problema2_lms , Problema2_rls , Problema2_cma y Problema2_dmi se usa una señal gaussiana aleatoria de entrada, se usa otra señal gaussiana aleatoria como señal de referencia

Problema3_lms y Problema3_rls se usa una función senosoidal de entrada por ejemplo Xn=sen(2*pi*k)/N y se usa como señal de referencia otra señal senosoidal distinta a la señal de entrada, pero que tenga una alta correlación a la señal de entrada por ejemplo d(k)=cos((2*pi*k)/N) el problema 3 se encuentra en los respectivos capítulos de la tesis

Problema1_lms

```
function[caratula]=problema1_lms;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
% Algoritmo LMS
% Autores: Juan Alvarez y Maribel Chuez
%
%Escuela Superior Politécnica del Litoral
%Facultad de Electricidad y Computacion
%Guayaquil, Ecuador
% email :juan_alvarez8@latinmail.com
% email : mchuez@ceibo.espol.edu.ec
% Webpage : proximamente
% Date    : Viernes 30 de abril del 2009
% Version : 1.0.1
% Referencias : S. Haykin, Adaptive Filter Theory. 3rd edition, Upper
Saddle River, NJ: Prentice-Hall, 1996.
%
 %clear all % esta funcion borra los valores anteriores que esten
memoria %
 %close all % esta funcion borra cualquier ventana anterior activa %
 %hold off   % esta funcion encera cualquier grafico que estuviera
activo %
fprintf('*****************************************************.\n')
fprintf('*                                                   *.\n')
fprintf('* PROBLEMA1_lms.- Determinar los pesos, el numero de *.\n')
fprintf('* iteraciones y la potencia                         *.\n')
fprintf('* de un grupo de antenas inteligentes utilizando el  *.\n')
fprintf('* algoritmo LMS (algoritmo minimo cuadrado, las      *.\n')
fprintf('* antenas estan distribuidas en forma de celdas para *.\n')
fprintf('* lo cual requeriremos los siguientes datos:         *.\n')
fprintf('*                                                   *.\n')
fprintf('* • la señal aplicada a la entrada del filtro        *.\n')
fprintf('*   ej  Xn=sin((2*pi*k)/N)                          *.\n')
fprintf('* • el numero de antenas  x                         *.\n')
fprintf('* • la señal de referencia, d(k)=2*cos((2*pi*k)/N);  *.\n')
fprintf('* • el ruido de las diferentes antenas, (hay que     *.\n')
fprintf('*   tener presente que el algoritmo LMS en su        *.\n')
fprintf('*   estructura internamente produce un ruido del 10  *.\n')
fprintf('*   por ciento, aparte de este ruido puede haber     *.\n')
fprintf('*   otros ruidos adicionales.  n(f)                 *.\n')
fprintf('* • el orden del filtro FIR cuyos coeficientes se    *.\n')
fprintf('*   van a ajustar, orden                            *.\n')
fprintf('* • el número de iteraciones en el aprendizaje, N    *.\n')
fprintf('*  (este número se hará hasta que cumpla el error)   *.\n')
```

```
fprintf('* • la constante de ajuste   u                        *.\n')
fprintf('* • el vector de pesos inicial. Los voltajes de las  *.\n')
fprintf('*   antenas V(f)                                      *.\n')
fprintf('* • el error deseado (puede ser ingresado por        *.\n')
fprintf('*   teclado) error                                    *.\n')
fprintf('*                                                     *.\n')
fprintf('*   A su salida devolverá:                            *.\n')
fprintf('* • la salida en cada iteración durante el proceso   *.\n')
fprintf('*   de ajuste, y(t)                                   *.\n')
fprintf('* • la curva de aprendizaje (grafico de la curva)    *.\n')
fprintf('* • los vectores de pesos obtenidos durante el       *.\n')
fprintf('*   proceso. W(t)                                     *.\n')
fprintf('* • la Potencia de la antena P(k)                    *.\n')
fprintf('* • el numero de iteraciones en que converge la curva*.\n')
fprintf('*   y se cumple el valor del error.  j                *.\n')
fprintf('*                                                     *.\n')
fprintf('*   alumno: Juan Francisco Alvarez Alvarado           *.\n')
fprintf('*   alumno: Maribel Del Rosario Chuez Gonzales        *.\n')
fprintf('                                                      *.\n')
fprintf('*   Profesor :Ing. Pedro Vargas                       *.\n')
fprintf('*                                                     *.\n')
fprintf('*******************************************************.\n')
 t=input('teclee enter para proseguir');
 clc
% ingresamos el orden del filtro, cuyo coeficiente se va ha ajustar%
orden=input(' ingrese el orden del filtro, cuyo coeficiente se va ha
ajustar....')
% los valores de h (pesos iniciales que recomendamos usar para el
ejemplo
% son h= 0.26; 0.25; 0.21 ;0.18; 0.15
% valores recomendados por Juan Alvarez y Maribel Chuez
% para este ejemplo se pueden ingresar valores cercanos a estos, de lo
contrario aumenta el error
% nota :ingresar valores que estén dentro del máximo de la curva,
% por lógica si ingresamos como peso por ejemplo 2 y la curva llega %
hasta 1 nunca lo encontrara, nunca habrá convergencia.
% ingresamos el numero de antenas del sistema
disp(' ingrese el numero de antenas entre 1 y 5');
x=input(' ingrese el numero de antenas entre 1,2,3,4 o 5....');
if x==1
    D(1)=input('ingrese el voltaje1 (en V)..... voltaje1=');
end

if x==2
    D(1)=input('ingrese el voltaje1 (en V)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
end

if x==3
```

```
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
end

if x==4
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
end
if x==5
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
end
% ingresamos el ruido que presenta cada antena del sistema
% Si existe ruido adicional al ocasionado por el sistema
if x==1
    E(1)=input(' ingrese el ruido en la antena1.....ruido1=');
    ruido=E(1);
end

if x==2
    E(1)=input('ingrese el ruido en la antena1.....ruido1==');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    ruido=(E(1)+E(2))/2;
end

if x==3
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    ruido=(E(1)+E(2)+E(3))/3;
end

if x==4
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    ruido=(E(1)+E(2)+E(3)+E(4))/4;
end
if x==5
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
```

```
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    ruido=(E(1)+E(2)+E(3)+E(4)+E(5))/5;
end
opcion_ruido=2;
disp('si desea que el ruido adicional vaya a la señal a la entrada
digite 1');
disp(' de lo contrario se considerara que el ruido adicional se
agregara a la señal de referencia');
opcion_ruido=input('opcion_ruido=');
error=input('ingrese el error del sistema=');
% numero de iteraciones del proceso hasta que se cumpla el error
N=2000;
j=0;
% vamos a considerar que todas las antenas tienen la misma señal a la
% entrada del filtro por ejemplo  entrada=sin((2*pi*k)/N)
if x==1
    h=[D(1)]
    p=[D(1)]
end
if x==2
    h=[D(1);D(2)]
    p=[(D(1)+D(2))/2]+[0.8*(D(1)+D(2))/2];
end
if x==3
    h=[D(1);D(2);D(3)]
    p=[(D(1)+D(2)+D(3))/3]+[0.8*(D(1)+D(2)+D(3))/3];
end
if x==4
    h=[D(1);D(2);D(3);D(4)]
    p=[(D(1)+D(2)+D(3)+D(4))/4]+[0.8*(D(1)+D(2)+D(3)+D(4))/4];
end
if x==5
    h=[D(1);D(2);D(3);D(4);D(5)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5))/5]+[0.8*(D(1)+D(2)+D(3)+D(4)+D(5))/5];
end
M=30;
for k=1:400
entrada(k)=sin((2*pi*k)/M);
% senal_referencia(k)=cos((2*pi*k)/M); % señal de referencia en algunos
libros se la llama señal deseada
end
% señal a la entrada del filtro, genera 400 valores aleatorios de
muestra
entrada';
entrada=entrada';
if opcion_ruido==1
    entrada=entrada+ruido;
```

```
end
n = sqrt(0.05)*randn(400,1); % ruido a la entrada del filtro, genera
400 valores aleatorios de muestra
[b,a] = butter(2,0.25) % usaremos otra forma para calcular a y b
Gz = tf(b,a,-1) % la funcion de transferencia tf funciona en matlab 7
% esta funcion permite resolver la transformada z,
% el primer orden evalua valores de pesos con la funcion ldiv
%h=ldiv(b,a,orden)';
% if you use ldiv this will give h :filter weights to be
% ingreso de los voltajes en la señal de entrada (estos voltajes seran
% valores de pesos iniciales que serviran para hallar los pesos W
y = lsim(Gz,entrada) % simula respuestas arbitrarias de entrada,
relaciona la señal Gz con la entrada
% para agregar el ruido
% este es el ruido que se genera internamente por el metodo LMS
% por lo general el valor de ruido que genera el algoritmo LMS puede
llegar al 10 % de la señal
n = n * std(y)/(10*std(n)); % aqui se genera el ruido del sistema
% el valor a la entrada del filtro es la suma de la señal entrante mas
el ruido
senal_referencia=y+n; % en la mayoría de los programas lms,rls,
if opcion_ruido>1 % filtros, se coloca el ruido del sistema en la señal
de referencia
    senal_referencia=senal_referencia+ruido;
end
totallength=size(senal_referencia,1) % da el largo de la funcion de %
entrada% conociendo los voltajes de las antenas podemos determinar % la
potencia en ellas
% potencia de la señal a la salida (potencia del sistema)
potencia=mean(senal_referencia.^2)
if x==1
% tambien calcularemos la potencia para cada una de las antenas
% antena1
entrada1=y+n+E(1);
potencia1=mean(entrada1.^2)
end
if x==2
% antena1
entrada1=D(1)+n+E(1);
potencia1=mean(entrada1.^2)
% antena2
entrada2=D(2)+n+E(2);
potencia2=mean(entrada2.^2)
end
if x==3
% antena1
entrada1=D(1)+n+E(1);
potencia1=mean(entrada1.^2)
% antena2
```

```
entrada2=D(2)+n+E(2);
potencia2=mean(entrada2.^2)
% antena3
entrada3=D(3)+n+E(3);
potencia3=mean(entrada3.^2);
end
if x==4
% antena1
entrada1=D(1)+n+E(1);
potencia1=mean(entrada1.^2)
% antena2
entrada2=D(2)+n+E(2);
potencia2=mean(entrada2.^2);
% antena3
entrada3=D(3)+n+E(3);
potencia3=mean(entrada3.^2)
% antena4
entrada4=D(4)+n+E(4);
potencia4=mean(entrada4.^2)
end
if x==5
% antena1
entrada1=D(1)+n+E(1);
potencia1=mean(entrada1.^2)
% antena2
entrada2=D(2)+n+E(2);
potencia2=mean(entrada2.^2)
% antena3
entrada3=D(3)+n+E(3);
potencia3=mean(entrada3.^2)
% antena4
entrada4=D(4)+n+E(4);
potencia4=mean(entrada4.^2)
% antena5
entrada5=D(5)+n+E(5);
potencia5=mean(entrada5.^2)
end

% calculo de la cota de la contante de ajuste
cota=1/((orden+1)*potencia);
%para 60 puntos  por corrida determinamos los pesos w
N=60
%begin of algorithm
w = zeros ( orden  , 1 )
for n = orden : N
    u = entrada(n:-1:n-orden+1);
    y(n)= w' * u;  % y(n) señal a la salida
    % e(n) = senal_referencia(n) - y(n);
    e(n)=senal_referencia(n)-y(n);
```

```
    j=j+1 % nos da el numero de iteraciones hasta que se cumpla el
error
     if e(n)<error
        fprintf('el numero de iteraciones hasta que se completo el
error ');
        j
     end

    % aqui determinamos el error del sistema
% mientras mayor sea el valor de constante de ajuste obtenemos una
% convergencia rapida pero menor exactitud en la respuesta de los
% mientras que si mu es pequeño la convergencia es mas lenta pero
% calcula mejor los pesos
% Start with big mu for speeding the convergence then slow down to
reach the correct weights
% se puede lograr mayor exactitud en la curva variando el valor de mu
% si se coloca valores de mu muy altos se pierde la señal de ajuste
% con valores de mu más pequeños se hace más lenta la convergencia
    if n < 20
        mu=cota/2
    else
        mu=cota/4
    end
    w = w + mu * u * e(n);
    % algunos autores usan el valor de w = w + 2*mu * u * e(n); para
    % evaluar los pesos
end
% chequeo de resultados de error
for n =  N+1 : totallength
    u = entrada(n:-1:n-orden+1) ;
    y(n) = w' * u ;   % y(n) señal a la salida
    % e(n) = senal_referencia(n) - y(n) ;
    e(n)=senal_referencia(n)-y(n);
end
hold on
plot(senal_referencia)
plot(y,'r');
title('salida del sistema') ;
xlabel('numero de muestras ')
ylabel('valor estimado a la salida ')
figure
semilogy((abs(e))) ;
title('curva de error') ;
xlabel('numero de muestras ')
ylabel('valor de error')
figure
plot(h, 'k+')
hold on
plot(w, 'r*')
```

```
legend('pesos actuales','pesos estimados')
title('comparacion de los pesos actuales con los pesos estimados') ;
axis([0 6 0.05 0.35])
```

```
function[caratula]=problema2_lms;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
% Algoritmo LMS
% Autores: Juan Alvarez y Maribel Chuez
%
%Escuela Superior Politecnica del Litoral
%Facultad de Electricidad y Computacion
%Guayaquil, Ecuador
% email :juan_alvarez8@latinmail.com
% email : mchuez@ceibo.espol.edu.ec
% Webpage : proximamente
% Date    : 18-04-2009
% Version : 1.0.2
% Referencias : S. Haykin, Adaptive Filter Theory. 3rd edition, Upper
Saddle River, NJ: Prentice-Hall, 1996.
%
% clear all % esta funcion borra los valores anteriores que esten
memoria %
% close all % esta funcion borra cualquier ventana anterior activa %
% hold off   % esta funcion encera cualquier grafico que estuviera
activo %
 fprintf('*****************************************************.\n')
 fprintf('*     ESCUELA SUPERIOR POLITECNICA DEL LITORAL      *.\n')
 fprintf('*                                                   *.\n')
 fprintf('*PROBLEMA LMS2.- Determinar los pesos, el numero de *.\n')
 fprintf('*iteraciones y la potencia de un grupo de antenas   *.\n')
 fprintf('*inteligentes utilizando el algoritmo LMS (algoritmo*.\n')
 fprintf('*mínimo cuadrado, las antenas están distribuidas en *.\n')
 fprintf('*forma de celdas para lo cual requeriremos          *.\n')
 fprintf('*los siguientes datos:                              *.\n')
 fprintf('*                                                   *.\n')
 fprintf('*  • la señal aplicada a la entrada del filtro,     *.\n')
 fprintf('*    ej  Xn=sin((2*pi*k)/N)                         *.\n')
 fprintf('*  • el numero de antenas  x                        *.\n')
 fprintf('*  • la señal de referencia,                        *.\n')
 fprintf('*    ejemplo d(k)=cos((2*pi*k)/N);                  *.\n')
 fprintf('*  • el ruido de las diferentes antenas,(hay que    *.\n')
 fprintf('*    tener presente que el algoritmo LMS en su      *.\n')
 fprintf('*    estructura internamente produce un ruido del   *.\n')
 fprintf('*    10 por ciento aparte de este ruido puede haber *.\n')
 fprintf('*    otros ruidos.                                  *.\n')
 fprintf('*  • el orden del filtro FIR cuyos coeficientes se  *.\n')
 fprintf('*    van a ajustar, orden                           *.\n')
 fprintf('*  • el número de iteraciones en el aprendizaje, N  *.\n')
 fprintf('*    (este número se hará                           *.\n')
 fprintf('*    hasta que cumpla el error)                     *.\n')
 fprintf('*  • la constante de ajuste   u                     *.\n')
 fprintf('*  • el vector de pesos inicial. Los voltajes de    *.\n')
```

```
 fprintf('*    las antenas V(f)                                    *.\n')
 fprintf('*  • el error deseado (puede ser ingresado por           *.\n')
 fprintf('*    teclado) error                                      *.\n')
 fprintf('*                                                        *.\n')
 fprintf('*    A su salida devolverá:                              *.\n')
 fprintf('*  • la salida en cada iteración durante el proceso      *.\n')
 fprintf('*    de ajuste, y(t)                                     *.\n')
 fprintf('*  • la curva de aprendizaje (grafico de la curva)       *.\n')
 fprintf('*  • los vectores de pesos obtenidos durante el          *.\n')
 fprintf('*    proceso. W(t)                                       *.\n')
 fprintf('*  • la Potencia de la antena P(k)                       *.\n')
 fprintf('*  • el numero de iteraciones en que converge la         *.\n')
 fprintf('*    curva y se cumple el valor del error. j             *.\n')
 fprintf('*                                                        *.\n')
 fprintf('*    alumno: Juan Francisco Alvarez Alvarado             *.\n')
 fprintf('*    alumno: Maribel del Rosario Chuez Gonzales          *.\n')
 fprintf('*                                                        *.\n')
 fprintf('*    Profesor :Ing. Pedro Vargas                         *.\n')
 fprintf('*                                                        *.\n')
fprintf('**********************************************************.\n')
 t=input('teclee enter para proseguir');
 clc
% ingresamos el orden del filtro, cuyo coeficiente se va ha ajustar%
orden=input(' ingrese el orden del filtro, cuyo coeficiente se va ha
ajustar....')

% ingresamos el numero de antenas del sistema
disp(' ingrese el numero de antenas entre 1 y 5');
x=input(' ingrese el numero de antenas entre 1,2,3,4 o 5....');
% los valores recomendado por la central matlab como valores de los
pesos, son
% h=0.0976;0.2873;0.3360;0.2210;0.0964
if x==1
    D(1)=input('ingrese el voltaje1 (en V)..... voltaje1=');
end

if x==2
    D(1)=input('ingrese el voltaje1 (en V)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
end

if x==3
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
end

if x==4
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
```

```
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
end
if x==5
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
end
% ingresamos el ruido que presenta cada antena del sistema
% Si existe ruido adicional al ocasionado por el sistema
if x==1
    E(1)=input(' ingrese el ruido en la antena1.....ruido1=');
    ruido=E(1);
end

if x==2
    E(1)=input('ingrese el ruido en la antena1.....ruido1==');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    ruido=(E(1)+E(2))/2;
end

if x==3
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    ruido=(E(1)+E(2)+E(3))/3;
end

if x==4
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    ruido=(E(1)+E(2)+E(3)+E(4))/4;
end
if x==5
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    ruido=(E(1)+E(2)+E(3)+E(4)+E(5))/5;
end
opcion_ruido=2;
disp('si desea que el ruido adicional vaya a la señal a la entrada
digite 1');
```

```
disp('de lo contrario se considerara que el ruido adicional se agregara
a la señal de referencia');
opcion_ruido=input('opcion_ruido=');
error=input('ingrese el error del sistema=');
% numero de iteraciones del proceso hasta que se cumpla el error
N=2000;
j=0;
% vamos a considerar que todas las antenas tienen la misma señal a la
% entrada del filtro por ejemplo  entrada=sin((2*pi*k)/N)
if x==1
    h=[D(1)]
    p=[D(1)]
end
if x==2
    h=[D(1);D(2)]
    p=[(D(1)+D(2))/2]+[0.8*(D(1)+D(2))/2];
end
if x==3
    h=[D(1);D(2);D(3)]
    p=[(D(1)+D(2)+D(3))/3]+[0.8*(D(1)+D(2)+D(3))/3];
end
if x==4
    h=[D(1);D(2);D(3);D(4)]
    p=[(D(1)+D(2)+D(3)+D(4))/4]+[0.8*(D(1)+D(2)+D(3)+D(4))/4];
end
if x==5
    h=[D(1);D(2);D(3);D(4);D(5)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5))/5]+[0.8*(D(1)+D(2)+D(3)+D(4)+D(5))/5];
end
entrada = randn(N,1); % señal a la entrada del filtro, genera 2000
valores aleatorios de muestra
if opcion_ruido==1
    entrada=entrada+ruido
end
n = randn(N,1); % ruido a la entrada del filtro, genera 2000 valores
aleatorios de muestra
[b,a] = butter(2,0.25);
Gz = tf(b,a,-1) % la funcion de transferencia tf funciona en matlab 7
% esta funcion permite resolver la transformada z,
% el primer orden evalua valores de pesos con la funcion ldiv
%h=ldiv(b,a,orden)';
% if you use ldiv this will give h :filter weights to be
% ingreso de los voltajes en la señal de entrada (estos voltajes seran
% valores de pesos iniciales que serviran para hallar los pesos W
y = lsim(Gz,entrada); % simula respuestas arbitrarias de entrada,
relaciona la señal Gz con inp
% para agregar el ruido
% este es el ruido que se genera internamente por el metodo LMS
```

```
% por lo general el valor de ruido que genera el algoritmo LMS puede
llegar al 10 % de la señal
n = n * std(y)/(10*std(n)); % aqui se genera el ruido del sistema
% el valor a la entrada del filtro es la suma de la señal entrante mas
el ruido
senal_referencia = y + n; % en la mayoría de los programs lms, rls,
filtros se coloca el ruido del sistema en la señal de referncia
if opcion_ruido>1
    senal_referencia=senal_referencia+ruido
end
totallength=size(senal_referencia,1); % da el largo de la funcion d
% conociendo los voltajes de las antenas podemos determinar la potencia
% en ellas
% potencia de la señal a la salida (potencia del sistema
potencia=mean(senal_referencia.^2)
if x==1
% tambien calcularemos la potencia para cada una de las antenas
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
end
if x==2
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
end
if x==3
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
end
if x==4
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
```

```
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
end
if x==5
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
end
% calculo de la cota de la contante de ajuste
cota=1/((orden+1)*potencia);
%para 60 puntos  por corrida determinamos los pesos w
N=60
%begin of algorithm
w = zeros ( orden  , 1 )
for n = orden : N
    u = entrada(n:-1:n-orden+1);
    y(n)= w' * u;
    e(n) = senal_referencia(n) - y(n);
    j=j+1 % nos da el numero de iteraciones hasta que se cumpla el
error
     if e(n)<error
        fprintf('el numero de iteraciones hasta que se completo el
error ');
        j
     end

    % aqui determinamos el error del sistema
% mientras mayor sea el valor de constante de ajuste
% obtenemos una convergencia rapida pero menor exactitud
% en la respuesta de los pesos, mientras que si mu es
% pequeño la convergencia es más lenta pero calcula mejor los pesos
    if n < 20
% mu=cota/2% en algunos libros recomiendan utilizar esta formula
% pasa la contante de ajuste
         mu=0.32 % valor recomendado por la biblioteca de matlab
```

```
% en caso que los valores de pesos no convergan bien cambiar el
% valor de mu
    else
        % mu=cota/4
         mu=0.15 % valor recomendado por la biblioteca de matlab
        % en caso que los valores de pesos no convergan bien cambiar el
        % valor de mu
    end
    w = w + mu * u * e(n); % algunos libros recomiendan utlilizar  w =
w + 2* mu * u * e(n)
    % para calcular los pesos
end
% chequeo de resultados de error
for n =  N+1 : totallength
    u = entrada(n:-1:n-orden+1) ;
    y(n) = w' * u ;
    e(n) = senal_referencia(n) - y(n) ;
end
hold on
plot(senal_referencia)
plot(y,'r');
title('salida del sistema') ;
xlabel('numero de muestras ')
ylabel('valor estimado a la salida ')
figure
semilogy((abs(e))) ;
title('curva de error') ;
xlabel('numero de muestras ')
ylabel('valor de error')
figure
plot(h, 'k+')
hold on
plot(w, 'r*')
legend('pesos actuales','pesos estimados')
title('comparacion de los pesos actuales con los pesos estimados') ;
axis([0 6 0.05 p])
```

```
function[caratula]=problema3_lms;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
% Algoritmo LMS
% Autores: Juan Alvarez , Maribel Chuez
%
%Escuela Superior Politecnica del Litoral
%Facultad de Electricidad y Computacion
%Guayaquil, Ecuador
% email :juan_alvarez8@latinmail.com
% email : mchuez@ceibo.espol.edu.ec
% Webpage : proximamente
% Date    : Viernes 30 de abril del 2009
% Version : 1.0.3
% Referencias : S. Haykin, Adaptive Filter Theory. 3rd edition, Upper
Saddle River, NJ: Prentice-Hall, 1996.
%
%clear all % esta función borra los valores anteriores que estén
memoria %
 %close all % esta función borra cualquier ventana anterior activa %
 %hold off   % esta función encera cualquier grafico que estuviera
activo %
 fprintf('*******************************************************.\n')
fprintf('*                                                     *.\n')
fprintf('*      ESCUELA SUPERIOR POLITECNICA DEL LITORAL       *.\n')
fprintf('*                                                     *.\n')
fprintf('* PROBLEMA3_lms.- Determinar los pesos, el numero de *.\n')
fprintf('* iteraciones y la potencia de un grupo de antenas   *.\n')
fprintf('* inteligentes utilizando el algoritmo LMS (algoritmo*.\n')
fprintf('* mínimo cuadrado, las antenas están distribuidas en *.\n')
fprintf('* forma de celdas para lo cual requeriremos los      *.\n')
fprintf('* siguientes datos:                                  *.\n')
fprintf('*                                                     *.\n')
fprintf('*  • la señal aplicada a la entrada del filtro,      *.\n')
fprintf('*         ej  Xn=2* sin((2*pi*k)/N)                  *.\n')
fprintf('*  • el numero de antenas  x                         *.\n')
fprintf('*  • la señal de referencia, d(k)=2*cos((2*pi*k)/N); *.\n')
fprintf('*  • el ruido de las diferentes antenas, (hay que    *.\n')
fprintf('*  tener presente que el algoritmo LMS en su         *.\n')
fprintf('*  extructura internamente produce un ruido del 10   *.\n')
fprintf('*  por ciento,aparte de este ruido puede haber otros.*.\n')
fprintf('*  ruidos adicionales que ingresaremos por teclado   *.\n')
fprintf('*  • el orden del filtro FIR cuyos coeficientes se   *.\n')
fprintf('*  van a ajustar, orden                              *.\n')
fprintf('*  • el número de iteraciones en el aprendizaje, N   *.\n')
fprintf('*  (este número se hará hasta que cumpla el error)   *.\n')
fprintf('*  • la constante de ajuste   mu                     *.\n')
fprintf('*  • el vector de pesos inicial. Los voltajes de las *.\n')
fprintf('*  antenas V(f)                                      *.\n')
```

```
fprintf('*  • el error deseado (puede ser ingresado error     *.\n')
fprintf('*   por teclado)                                     *.\n')
fprintf('*                                                    *.\n')
fprintf('*  A su salida devolverá:                            *.\n')
fprintf('*  • la salida en cada iteración durante el proceso  *.\n')
fprintf('*  de ajuste, y(t)                                   *.\n')
fprintf('*  • la curva de aprendizaje (grafico de la curva)   *.\n')
fprintf('*  • los vectores de pesos obtenidos durante el      *.\n')
fprintf('*  proceso. W(t)                                     *.\n')
fprintf('*  • la Potencia de la antena P(k)                   *.\n')
fprintf('*  • el numero de iteraciones en que converge la     *.\n')
fprintf('*  curva y se cumple el valor                        *.\n')
fprintf('*       del error.  j                                *.\n')
fprintf('*    alumno: Juan Francisco Alvarez Alvarado         *.\n')
fprintf('*    alumno: Maribel del Rosario Chuez Gonzales      *.\n')
fprintf('*                                                    *.\n')
fprintf('*    Profesor :Ing. Pedro Vargas                     *.\n')
fprintf('*                                                    *.\n')
fprintf('******************************************************.\n')
t=input('teclee enter para proseguir');
 clc
% ingresamos el orden del filtro, cuyo coeficiente se va ha ajustar%
orden=input(' ingrese el orden del filtro, cuyo coeficiente se va ha
ajustar....')
% los valores de pesos que recomendamos para en estas dos curvas son:
% h=0.5; 0.2; -0.2; -0.4; -0.7
% valores recomendado por Juan Alvarez y Maribel Chuez
% se tiene que ingresar valores que estén dentro del rango de la curva
% se puede ingresar valores cercanos a estos, pero el error aumentara
% mientras más alejados sean de estos valores.
% ingresamos el número de antenas del sistema
disp(' ingrese el numero de antenas entre 1 y 5');
x=input(' ingrese el numero de antenas entre 1,2,3,4 o 5....');
if x==1
    D(1)=input('ingrese el voltaje1 (en V)..... voltaje1=');
end

if x==2
    D(1)=input('ingrese el voltaje1 (en V)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
end

if x==3
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
end

if x==4
```

```
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
end
if x==5
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
end
% ingresamos el ruido que presenta cada antena del sistema
% Si existe ruido adicional al ocasionado por el sistema
if x==1
    E(1)=input(' ingrese el ruido en la antena1.....ruido1=');
    ruido=E(1);
end

if x==2
    E(1)=input('ingrese el ruido en la antena1.....ruido1==');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    ruido=(E(1)+E(2))/2;
end

if x==3
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    ruido=(E(1)+E(2)+E(3))/3;
end

if x==4
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    ruido=(E(1)+E(2)+E(3)+E(4))/4;
end
if x==5
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    ruido=(E(1)+E(2)+E(3)+E(4)+E(5))/5;
end
opcion_ruido=2;
```

```
disp('si desea que el ruido adicional vaya a la señal a la entrada
digite 1');
disp(' de lo contrario se considerara que el ruido adicional se
agregara a la señal de referencia');
opcion_ruido=input('opcion_ruido=');
error=input('ingrese el error del sistema=');
% numero de iteraciones del proceso hasta que se cumpla el error
opcion_senal=2;
disp('si desea que la señal de entrada se use como señal de referencia
digite 1');
disp(' de lo contrario se considerara otra señal diferente como señal
de referencia ');
opcion_senal=input('opcion_ruido=');
% numero de iteraciones del proceso hasta que se cumpla el error
N=2000;
j=0;
% vamos a considerar que todas las antenas tienen la misma señal a la
% entrada del filtro por ejemplo  entrada=2*sin((2*pi*k)/N)
if x==1
    h=[D(1)]
    p=[D(1)]
end
if x==2
    h=[D(1);D(2)]
    p=[(D(1)+D(2))/2]+[0.8*(D(1)+D(2))/2];
end
if x==3
    h=[D(1);D(2);D(3)]
    p=[(D(1)+D(2)+D(3))/3]+[0.8*(D(1)+D(2)+D(3))/3];
end
if x==4
    h=[D(1);D(2);D(3);D(4)]
    p=[(D(1)+D(2)+D(3)+D(4))/4]+[0.8*(D(1)+D(2)+D(3)+D(4))/4];
end
if x==5
    h=[D(1);D(2);D(3);D(4);D(5)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5))/5]+[0.8*(D(1)+D(2)+D(3)+D(4)+D(5))/5];
end
M=30;
for k=1:400
entrada(k)=sin((2*pi*k)/M);
% senal_referencia(k)=cos((2*pi*k)/M); % señal de referencia en algunos
libros se la llama señal deseada
end
for k=1:400
%entrada(k)=sin((2*pi*k)/M);
senal_referencia(k)=cos((2*pi*k)/M); % señal de referencia en algunos
libros se la llama señal deseada
```

```
end
% señal a la entrada del filtro, genera 2000 valores aleatorios de
muestra
entrada';
entrada=entrada'
if opcion_ruido==1
    entrada=entrada+ruido
end
n = sqrt(0.05)*randn(400,1); % ruido a la entrada del filtro, genera
2000 valores aleatorios de muestra
[b,a] = butter(2,0.25) % usaremos otra forma para calcular a y b
Gz = tf(b,a,-1) % la funcion de transferencia tf funciona en matlab 7
% esta funcion permite resolver la transformada z,
% el primer orden evalua valores de pesos con la funcion ldiv
%h=ldiv(b,a,orden)';
% if you use ldiv this will give h :filter weights to be
% ingreso de los voltajes en la señal de entrada (estos voltajes serán
% los valores de pesos iniciales que serviran para hallar los pesos W
y = lsim(Gz,entrada) % simula respuestas arbitrarias de entrada,
relaciona la señal Gz con la entrada
% para agregar el ruido
% este es el ruido que se genera internamente por el metodo LMS
% por lo general el valor de ruido que genera el algoritmo LMS puede
llegar al 10 % de la señal
n = n * std(y)/(10*std(n)); % aqui se genera el ruido del sistema
% el valor a la entrada del filtro es la suma de la señal entrante mas
el ruido
if opcion_senal==1
senal_referencia = y + n;
end
if opcion_senal>1
    senal_referencia'
    senal_referencia=senal_referencia'
    senal_referencia=senal_referencia+n
end
if opcion_ruido>1
    senal_referencia=senal_referencia+ruido
end
totallength=size(senal_referencia,1) % da el largo de la función de
entrada

% conociendo los voltajes de las antenas podemos determinar la potencia
en ellas
% potencia de la señal a la salida (potencia del sistema
potencia=mean(senal_referencia.^2)
if x==1
% tambien calcularemos la potencia para cada una de las antenas
% antena1
entrada1=D(1)+n+E(1);
```

```
potencia1=mean(entrada1.^2)
end
if x==2
% antena1
entrada1=D(1)+n+E(1);
potencia1=mean(entrada1.^2)
% antena2
entrada2=D(2)+n+E(2);
potencia1=mean(entrada2.^2)
end
if x==3
% antena1
entrada1=D(1)+n+E(1);
potencia1=mean(entrada1.^2)
% antena2
entrada2=D(2)+n+E(2);
potencia1=mean(entrada2.^2)
% antena3
entrada3=D(3)+n+E(3);
potencia3=mean(entrada3.^2)
end
if x==4
% antena1
entrada1=D(1)+n+E(1);
potencia1=mean(entrada1.^2)
% antena2
entrada2=D(2)+n+E(2);
potencia1=mean(entrada2.^2)
% antena3
entrada3=D(3)+n+E(3);
potencia3=mean(entrada3.^2)
% antena4
entrada4=D(4)+n+E(4);
potencia4=mean(entrada4.^2)
end
if x==5
% antena1
entrada1=D(1)+n+E(1);
potencia1=mean(entrada1.^2)
% antena2
entrada2=D(2)+n+E(2);
potencia2=mean(entrada2.^2)
% antena3
entrada3=D(3)+n+E(3);
potencia3=mean(entrada3.^2)
% antena4
entrada4=D(4)+n+E(4);
potencia4=mean(entrada4.^2)
% antena5
```

```
entrada5=D(5)+n+E(5);
potencia5=mean(entrada5.^2)
end

% calculo de la cota de la contante de ajuste
cota=1/((orden+1)*potencia);
%para 60 puntos  por corrida determinamos los pesos w
N=60
%begin of algorithm
w = zeros ( orden  , 1 )
for n = orden : N
    u = entrada(n:-1:n-orden+1);
    y(n)= w' * u;  % y(n) señal a la salida
    % e(n) = senal_referencia(n) - y(n);
    e(n)=senal_referencia(n)-y(n);
    j=j+1 % nos da el numero de iteraciones hasta que se cumpla el
error
     if e(n)<error
        fprintf('el numero de iteraciones hasta que se completo el
error ');
        j
     end

    % aquí determinamos el error del sistema
% mientras mayor sea el valor de constante de ajuste
% obtenemos una convergencia rápida pero menor exactitud
% en la respuesta de los pesos, mientras que si mu es
% pequeño la convergencia es más lenta pero calcula mejor los pesos
% Start with big mu for speeding the convergence then slow down to
reach the correct weights
% se puede lograr mayor exactitud en la curva variando el valor de mu
% si se coloca valores de mu muy altos se pierde la señal de ajuste
% con valores de mu más pequeños se hace más lenta la convergencia
    if n < 20
        mu=cota/2
    else
        mu=cota/4
    end
    w = w + mu * u * e(n);
    % algunos autores usan el valor de w = w + 2*mu * u * e(n); para
    % evaluar los pesos, y hay personas que están tratando de mejorar
%esta ecuacion
end
% chequeo de resultados de error
for n =  N+1 : totallength
    u = entrada(n:-1:n-orden+1) ;
    y(n) = w' * u ;   % y(n) señal a la salida
    % e(n) = senal_referencia(n) - y(n) ;
    e(n)=senal_referencia(n)-y(n);
```

```
end
hold on
plot(senal_referencia)
plot(y,'r');
title('salida del sistema') ;
xlabel('numero de muestras ')
ylabel('valor estimado a la salida ')
figure
semilogy((abs(e))) ;
title('curva de error') ;
xlabel('numero de muestras ')
ylabel('valor de error')
figure
plot(h, 'k+')
hold on
plot(w, 'r*')
legend('pesos actuales','pesos estimados')
title('comparacion de los pesos actuales con los pesos estimados') ;
% axis([coordenada eje x- coordenada eje x+  coordenada y- coordenada
y+])
axis([0 6 -1 1])
```

```
function[caratula]=problema1_rls;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
% Algoritmo RLS
% Autores: Juan Alvarez y Maribel Chuez
%
%Escuela Superior Politécnica del Litoral
%Facultad de Electricidad y Computacion
%Guayaquil, Ecuador
% email :juan_alvarez8@latinmail.com
% email : mchuez@ceibo.espol.edu.ec
% Webpage : proximamente
% Date    : Viernes 30 de abril del 2009
% Version : 1.0.1
% Referencias : S. Haykin, Adaptive Filter Theory. 3rd edition, Upper
Saddle River, NJ: Prentice-Hall, 1996.
%
%clear all % esta funcion borra los valores anteriores que esten
memoria %
%close all % esta funcion borra cualquier ventana anterior activa %
%hold off   % esta funcion encera cualquier grafico que estuviera
activo %
fprintf('*****************************************************.\n')
fprintf('*    ESCUELA SUPERIOR POLITECNICA DEL LITORAL        *.\n')
fprintf('*                                                    *.\n')
fprintf('* PROBLEMA1_rls.- Determinar los pesos, el numero de *.\n')
fprintf('* iteraciones y la potencia de un grupo de antenas   *.\n')
fprintf('* inteligentes utilizando el algoritmo RLS (algoritmo*.\n')
fprintf('* minimo cuadrado recursivo, las antenas estan distri*.\n')
fprintf('* buidas en forma de celdas                          *.\n')
fprintf('* para lo cual requeriremos de los siguientes datos: *.\n')
fprintf('*                                                    *.\n')
fprintf('*  • la señal aplicada a la entrada del filtro,      *.\n')
fprintf('*   ej  Xn=sin((2*pi*k)/N)                           *.\n')
fprintf('*  • el numero de antenas  x                         *.\n')
fprintf('*  • la señal de referencia, d(k)=cos((2*pi*k)/N);   *.\n')
fprintf('*  • el ruido de las diferentes antenas, (hay que    *.\n')
fprintf('*  tener presente que el algoritmo RLS en su         *.\n')
fprintf('*  estructura internamente produce un ruido que puede*.\n')
fprintf('*  ser de 2, 3 y hasta 10 %  aparte de este ruido    *.\n')
fprintf('*  generado por el mismo programa RLS se puede       *.\n')
fprintf('*  ingresar ruidos adicionales en el programa a la   *.\n')
fprintf('*  entrada o a la salida del programa.  n(f)         *.\n')
fprintf('*  • el orden del filtro FIR cuyos coeficientes se   *.\n')
fprintf('*  van a ajustar, orden                              *.\n')
fprintf('*  • el número de iteraciones en el aprendizaje, N   *.\n')
fprintf('*  (este número se hará  hasta que cumpla el error   *.\n')
fprintf('*  • la constante de ajuste   u                      *.\n')
```

```
fprintf('*  • el vector de pesos inicial. Los voltajes de las *.\n')
fprintf('*  antenas V(f)                                       *.\n')
fprintf('*  • el error deseado puede ser ingresado por teclado*.\n')
fprintf('*    error                                            *.\n')
fprintf('*                                                     *.\n')
fprintf('*  A su salida devolverá:                             *.\n')
fprintf('*  • la salida en cada iteración durante el proceso   *.\n')
fprintf('*  de ajuste, y(t)                                    *.\n')
fprintf('*  • la curva de aprendizaje (grafico de la curva)    *.\n')
fprintf('*  • los vectores de pesos obtenidos durante el pro-  *.\n')
fprintf('*  ceso. W(t)                                         *.\n')
fprintf('*  • la Potencia de la antena P(k)                    *.\n')
fprintf('*  • el numero de iteraciones en que converge la      *.\n')
fprintf('*    curva y se cumple el valor del error             *.\n')
fprintf('*                                                     *.\n')
fprintf('* alumno: Juan Francisco Alvarez Alvarado             *.\n')
fprintf('* alumno: Maribel del Rosario Chuez Gonzales          *.\n')
fprintf('*                                                     *.\n')
fprintf('* Profesor :Ing. Pedro Vargas                         *.\n')
fprintf('*                                                     *.\n')
fprintf('*******************************************************.\n')
t=input('teclee enter para proseguir');
 clc
% ingresamos el orden del filtro, cuyo coeficiente se va ha ajustar%
orden=input(' ingrese el orden del filtro, cuyo coeficiente se va ha
ajustar....')

% ingresamos el numero de antenas del sistema
disp(' ingrese el numero de antenas entre 1 y 15');
x=input(' ingrese el numero de antenas entre
1,2,3,4,5,6,7,8,9,10,11,12,13,14 o 15....');
if x==1
    D(1)=input('ingrese el voltaje1 (en V)..... voltaje1=');
end

if x==2
    D(1)=input('ingrese el voltaje1 (en V)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
end

if x==3
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
end

if x==4
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
```

```
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
end
if x==5
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
end
if x==6
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
end
if x==7
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
    D(7)=input(' ingrese la voltaje7 (en v)..... voltaje7=');
end
if x==8
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
    D(7)=input(' ingrese la voltaje7 (en v)..... voltaje7=');
    D(8)=input(' ingrese la voltaje8 (en v)..... voltaje8=');
end
if x==9
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
    D(7)=input(' ingrese la voltaje7 (en v)..... voltaje7=');
    D(8)=input(' ingrese la voltaje8 (en v)..... voltaje8=');
    D(9)=input(' ingrese la voltaje9 (en v)..... voltaje9=');
end
if x==10
```

```
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
    D(7)=input(' ingrese la voltaje7 (en v)..... voltaje7=');
    D(8)=input(' ingrese la voltaje8 (en v)..... voltaje8=');
    D(9)=input(' ingrese la voltaje9 (en v)..... voltaje9=');
    D(10)=input(' ingrese la voltaje10 (en v)..... voltaje10=');
end
if x==11
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
    D(7)=input(' ingrese la voltaje7 (en v)..... voltaje7=');
    D(8)=input(' ingrese la voltaje8 (en v)..... voltaje8=');
    D(9)=input(' ingrese la voltaje9 (en v)..... voltaje9=');
    D(10)=input(' ingrese la voltaje10 (en v)..... voltaje10=');
    D(11)=input(' ingrese la voltaje11 (en v)..... voltaje11=');
end
if x==12
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
    D(7)=input(' ingrese la voltaje7 (en v)..... voltaje7=');
    D(8)=input(' ingrese la voltaje8 (en v)..... voltaje8=');
    D(9)=input(' ingrese la voltaje9 (en v)..... voltaje9=');
    D(10)=input(' ingrese la voltaje10 (en v)..... voltaje10=');
    D(11)=input(' ingrese la voltaje11 (en v)..... voltaje11=');
    D(12)=input(' ingrese la voltaje12 (en v)..... voltaje12=');
end
if x==13
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
    D(7)=input(' ingrese la voltaje7 (en v)..... voltaje7=');
    D(8)=input(' ingrese la voltaje8 (en v)..... voltaje8=');
    D(9)=input(' ingrese la voltaje9 (en v)..... voltaje9=');
    D(10)=input(' ingrese la voltaje10 (en v)..... voltaje10=');
```

```
    D(11)=input(' ingrese la voltaje11 (en v)..... voltaje11=');
    D(12)=input(' ingrese la voltaje12 (en v)..... voltaje12=');
    D(13)=input(' ingrese la voltaje13 (en v)..... voltaje13=');
end
if x==14
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
    D(7)=input(' ingrese la voltaje7 (en v)..... voltaje7=');
    D(8)=input(' ingrese la voltaje8 (en v)..... voltaje8=');
    D(9)=input(' ingrese la voltaje9 (en v)..... voltaje9=');
    D(10)=input(' ingrese la voltaje10 (en v)..... voltaje10=');
    D(11)=input(' ingrese la voltaje11 (en v)..... voltaje11=');
    D(12)=input(' ingrese la voltaje12 (en v)..... voltaje12=');
    D(13)=input(' ingrese la voltaje13 (en v)..... voltaje13=');
    D(14)=input(' ingrese la voltaje14 (en v)..... voltaje14=');
end
if x==15
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
    D(7)=input(' ingrese la voltaje7 (en v)..... voltaje7=');
    D(8)=input(' ingrese la voltaje8 (en v)..... voltaje8=');
    D(9)=input(' ingrese la voltaje9 (en v)..... voltaje9=');
    D(10)=input(' ingrese la voltaje10 (en v)..... voltaje10=');
    D(11)=input(' ingrese la voltaje11 (en v)..... voltaje11=');
    D(12)=input(' ingrese la voltaje12 (en v)..... voltaje12=');
    D(13)=input(' ingrese la voltaje13 (en v)..... voltaje13=');
    D(14)=input(' ingrese la voltaje14 (en v)..... voltaje14=');
    D(15)=input(' ingrese la voltaje15 (en v)..... voltaje15=');
end

% ingresamos el ruido que presenta cada antena del sistema
% Si existe ruido adicional al ocasionado por el sistema
if x==1
    E(1)=input(' ingrese el ruido en la antena1.....ruido1=');
    ruido=E(1);
end

if x==2
    E(1)=input('ingrese el ruido en la antena1.....ruido1==');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    ruido=(E(1)+E(2))/2;
```

```
end

if x==3
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    ruido=(E(1)+E(2)+E(3))/3;
end

if x==4
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    ruido=(E(1)+E(2)+E(3)+E(4))/4;
end
if x==5
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    ruido=(E(1)+E(2)+E(3)+E(4)+E(5))/5;
end
if x==6
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6))/6;
end
if x==7
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    E(7)=input('ingrese el ruido en la antena7.....ruido7=');
    ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6)+E(7))/7;
end
if x==8
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
```

```
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    E(7)=input('ingrese el ruido en la antena7.....ruido7=');
    E(8)=input('ingrese el ruido en la antena8.....ruido8=');
    ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6)+E(7)+E(8))/8;
end
if x==9
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    E(7)=input('ingrese el ruido en la antena7.....ruido7=');
    E(8)=input('ingrese el ruido en la antena8.....ruido8=');
    E(9)=input('ingrese el ruido en la antena9.....ruido9=');
    ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6)+E(7)+E(8)+E(9))/9;
end
if x==10
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    E(7)=input('ingrese el ruido en la antena7.....ruido7=');
    E(8)=input('ingrese el ruido en la antena8.....ruido8=');
    E(9)=input('ingrese el ruido en la antena9.....ruido9=');
    E(10)=input('ingrese el ruido en la antena10.....ruido10=');
    ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6)+E(7)+E(8)+E(9)+E(10))/10;
end
if x==11
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    E(7)=input('ingrese el ruido en la antena7.....ruido7=');
    E(8)=input('ingrese el ruido en la antena8.....ruido8=');
    E(9)=input('ingrese el ruido en la antena9.....ruido9=');
    E(10)=input('ingrese el ruido en la antena10.....ruido10=');
    E(11)=input('ingrese el ruido en la antena11.....ruido11=');

ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6)+E(7)+E(8)+E(9)+E(10)+E(11))/11;
end
if x==12
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
```

```
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    E(7)=input('ingrese el ruido en la antena7.....ruido7=');
    E(8)=input('ingrese el ruido en la antena8.....ruido8=');
    E(9)=input('ingrese el ruido en la antena9.....ruido9=');
    E(10)=input('ingrese el ruido en la antena10.....ruido10=');
    E(11)=input('ingrese el ruido en la antena11.....ruido11=');
    E(12)=input('ingrese el ruido en la antena12.....ruido12=');

ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6)+E(7)+E(8)+E(9)+E(10)+E(11)+E(12))/
12;
end
if x==13
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    E(7)=input('ingrese el ruido en la antena7.....ruido7=');
    E(8)=input('ingrese el ruido en la antena8.....ruido8=');
    E(9)=input('ingrese el ruido en la antena9.....ruido9=');
    E(10)=input('ingrese el ruido en la antena10.....ruido10=');
    E(11)=input('ingrese el ruido en la antena11.....ruido11=');
    E(12)=input('ingrese el ruido en la antena12.....ruido12=');
    E(13)=input('ingrese el ruido en la antena13.....ruido13=');

ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6)+E(7)+E(8)+E(9)+E(10)+E(11)+E(12)+E
(13))/13;
end
if x==14
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    E(7)=input('ingrese el ruido en la antena7.....ruido7=');
    E(8)=input('ingrese el ruido en la antena8.....ruido8=');
    E(9)=input('ingrese el ruido en la antena9.....ruido9=');
    E(10)=input('ingrese el ruido en la antena10.....ruido10=');
    E(11)=input('ingrese el ruido en la antena11.....ruido11=');
    E(12)=input('ingrese el ruido en la antena12.....ruido12=');
    E(13)=input('ingrese el ruido en la antena13.....ruido13=');
    E(14)=input('ingrese el ruido en la antena14.....ruido14=');

ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6)+E(7)+E(8)+E(9)+E(10)+E(11)+E(12)+E
(13)+E(14))/14;
```

```
end
if x==15
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    E(7)=input('ingrese el ruido en la antena7.....ruido7=');
    E(8)=input('ingrese el ruido en la antena8.....ruido8=');
    E(9)=input('ingrese el ruido en la antena9.....ruido9=');
    E(10)=input('ingrese el ruido en la antena10.....ruido10=');
    E(11)=input('ingrese el ruido en la antena11.....ruido11=');
    E(12)=input('ingrese el ruido en la antena12.....ruido12=');
    E(13)=input('ingrese el ruido en la antena13.....ruido13=');
    E(14)=input('ingrese el ruido en la antena14.....ruido14=');
    E(15)=input('ingrese el ruido en la antena15.....ruido15=');

ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6)+E(7)+E(8)+E(9)+E(10)+E(11)+E(12)+E
(13)+E(14)+E(15))/15;
end

opcion_ruido=2;
disp('si desea que el ruido adicional vaya a la señal a la entrada
digite 1');
disp(' de lo contrario se considerara que el ruido adicional se
agregara a la señal de referencia');
opcion_ruido=input('opcion_ruido=');
error=input('ingrese el error del sistema=');
% numero de iteraciones del proceso hasta que se cumpla el error
j=0;
% valores recomendado por la central matlab para los pesos en una
funcion gaussiana
% h=[0.097631   0.287310   0.335965   0.220981 0.096354 0.017183
% -0.015917  -0.020735  -0.014243  -0.006517 -0.001396   0.000856
%  0.001272  0.000914  0.000438 0.000108 -0.000044  -0.00008
% -0.000058 -0.000029];
% vamos a considerar que todas las antenas tienen la misma señal a
% la entrada del filtro por ejemplo  entrada=sin((2*pi*k)/N);
% para señales gaussiana se usa entrada=randn(N,1)
% en el presente programa se usa una señal senosoidal como señal de %
entrada y de referencia (problema1)
if x==1
    h=[D(1)]
    p=[D(1)]
end
if x==2
    h=[D(1);D(2)]
    p=[(D(1)+D(2))/2]+[0.8*(D(1)+D(2))/2];
```

```
end
if x==3
    h=[D(1);D(2);D(3)]
    p=[(D(1)+D(2)+D(3))/3]+[0.8*(D(1)+D(2)+D(3))/3];
end
if x==4
    h=[D(1);D(2);D(3);D(4)]
    p=[(D(1)+D(2)+D(3)+D(4))/4]+[0.8*(D(1)+D(2)+D(3)+D(4))/4];
end
if x==5
    h=[D(1);D(2);D(3);D(4);D(5)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5))/5]+[0.8*(D(1)+D(2)+D(3)+D(4)+D(5))/5];
end
if x==6
    h=[D(1);D(2);D(3);D(4);D(5);D(6)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6))/6]+[0.8*(D(1)+D(2)+D(3)+D(4)+D(5)+D(
6))/6];
end
if x==7
    h=[D(1);D(2);D(3);D(4);D(5);D(6);D(7)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7))/7]+[0.8*(D(1)+D(2)+D(3)+D(4)+D(
5)+D(6)+D(7))/7];
end
if x==8
    h=[D(1);D(2);D(3);D(4);D(5);D(6);D(7);D(8)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8))/8]+[0.8*(D(1)+D(2)+D(3)+D(
4)+D(5)+D(6)+D(7)+D(8))/8];
end
if x==9
    h=[D(1);D(2);D(3);D(4);D(5);D(6);D(7);D(8);D(9)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9))/9]+[0.8*(D(1)+D(2)+D(
3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9))/9];
end
if x==10
    h=[D(1);D(2);D(3);D(4);D(5);D(6);D(7);D(8);D(9);D(10)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10))/10]+[0.8*(D(1)+
D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10))/10];
end
if x==11
    h=[D(1);D(2);D(3);D(4);D(5);D(6);D(7);D(8);D(9);D(10);D(11)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10)+D(11))/11]+[0.8*
(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10)+D(11))/11];
```

```
end
if x==12
    h=[D(1);D(2);D(3);D(4);D(5);D(6);D(7);D(8);D(9);D(10);D(11);D(12)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10)+D(11)+D(12))/12]
+[0.8*(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10)+D(11)+D(12))/
12];
end
if x==13

h=[D(1);D(2);D(3);D(4);D(5);D(6);D(7);D(8);D(9);D(10);D(11);D(12);D(13)
]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10)+D(11)+D(12)+D(13
))/13]+[0.8*(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10)+D(11)+D
(12)+D(13))/13];
end
if x==14

h=[D(1);D(2);D(3);D(4);D(5);D(6);D(7);D(8);D(9);D(10);D(11);D(12);D(13)
;D(14)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10)+D(11)+D(12)+D(13
)+D(14))/14]+[0.8*(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10)+D
(11)+D(12)+D(13)+D(14))/14];
end
if x==15

h=[D(1);D(2);D(3);D(4);D(5);D(6);D(7);D(8);D(9);D(10);D(11);D(12);D(13)
;D(14);D(15)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10)+D(11)+D(12)+D(13
)+D(14)+D(15))/15]+[0.8*(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D
(10)+D(11)+D(12)+D(13)+D(14)+D(15))/15];
end
M=30;
for k=1:2000
    entrada(k)=sin((2*pi*k)/M);
    %senal de entrada
end
entrada=entrada';
if opcion_ruido==1
    entrada=entrada+ruido; % aqui agregamos el ruido adicional a la
entrada
end
n=sqrt(0.05)*randn(2000,1); % ruido a la entrada del filtro, genera
2000 valores
% aleatorios de muestra
[b,a] = butter(2,0.25); % esta funcion calcula los valores de a y b
```

```
Gz = tf(b,a,-1);
% la funcion de transferencia tf funciona en matlab 7
% esta funcion permite resolver la transformada z.

%Si no se tiene acceso a la caja de herramientas del matlab se puede
cargar valores de
%la central matlab usando el sample data
%load IIRsampledata;
%entrada= IIRsampledata(1:2000);
%d= IIRsampledata(1:2000);

% se puede usar ldiv para hallar las aproximaciones de pesos para un
filtro IIR (filtro de respuesta infinita)
% tambien se puede evaluar  y=h*u
%ldiv es una funcion que nos permite hallar la inversa de la
transformada Z (Matlab central file exchange)
%para hallar los valores de orden de los pesos se puede utilizar la
%siguiente formula
%use h=ldiv(b,a,sysorder)'; ==> here we use sysorder == 10

h=h(1:orden);
y = lsim(Gz,entrada);% simula respuestas arbitrarias de entrada,
relacion la señal Gz
% con la entrada
% para agregar el ruido, ruido adicional que genera el algoritmo rls
n = n * std(y)/(10*std(n));
senal_referencia = y + n; % en la mayoría de los programas lms, rls,
filtros se coloca el ruido del sistema en la señal de referencia

if opcion_ruido>1
    senal_referencia=senal_referencia+ruido; % agrega el ruido
adicional a la salida del filtro
end
totallength=size(senal_referencia,1);
% nos da el numero de valores de la funcion de entrada
% conociendo los voltajes de las antenas podemos determinar la
% potencia en ellas
% potencia de la señal a la salida (potencia del sistema)
potencia=mean(senal_referencia.^2)
if x==1
% tambien calcularemos la potencia para cada una de las antenas
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
end
if x==2
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
```

```
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
end
if x==3
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
end
if x==4
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia1=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
end
if x==5
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
end
if x==6
% antena1
d1=D(1)+n+E(1);
```

```
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
end
if x==7
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
% antena7
d7=D(7)+n+E(7);
potencia7=mean(d7.^2)
end
if x==8
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
```

```
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
% antena7
d7=D(7)+n+E(7);
potencia7=mean(d7.^2)
% antena8
d8=D(8)+n+E(8);
potencia8=mean(d8.^2)
end
if x==9
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
% antena7
d7=D(7)+n+E(7);
potencia7=mean(d7.^2)
% antena8
d8=D(8)+n+E(8);
potencia8=mean(d8.^2)
% antena9
d9=D(9)+n+E(9);
potencia9=mean(d9.^2)
end
if x==10
% antena1
d1=D(1)+n+E(1);
```

```
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
% antena7
d7=D(7)+n+E(7);
potencia7=mean(d7.^2)
% antena8
d8=D(8)+n+E(8);
potencia8=mean(d8.^2)
% antena9
d9=D(9)+n+E(9);
potencia9=mean(d9.^2)
% antena10
d10=D(10)+n+E(10);
potencia9=mean(d10.^2)
end
if x==11
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
% antena7
```

```
d7=D(7)+n+E(7);
potencia7=mean(d7.^2)
% antena8
d8=D(8)+n+E(8);
potencia8=mean(d8.^2)
% antena9
d9=D(9)+n+E(9);
potencia9=mean(d9.^2)
% antena10
d10=D(10)+n+E(10);
potencia10=mean(d10.^2)
% antena11
d11=D(11)+n+E(11);
potencia11=mean(d11.^2)
end
if x==12
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
% antena7
d7=D(7)+n+E(7);
potencia7=mean(d7.^2)
% antena8
d8=D(8)+n+E(8);
potencia8=mean(d8.^2)
% antena9
d9=D(9)+n+E(9);
potencia9=mean(d9.^2)
% antena10
d10=D(10)+n+E(10);
potencia10=mean(d10.^2)
% antena11
d11=D(11)+n+E(11);
potencia11=mean(d11.^2)
```

```
% antena12
d12=D(12)+n+E(12);
potencia12=mean(d12.^2)
end
if x==13
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
% antena7
d7=D(7)+n+E(7);
potencia7=mean(d7.^2)
% antena8
d8=D(8)+n+E(8);
potencia8=mean(d8.^2)
% antena9
d9=D(9)+n+E(9);
potencia9=mean(d9.^2)
% antena10
d10=D(10)+n+E(10);
potencia10=mean(d10.^2)
% antena11
d11=D(11)+n+E(11);
potencia11=mean(d11.^2)
% antena12
d12=D(12)+n+E(12);
potencia12=mean(d12.^2)
% antena13
d13=D(13)+n+E(13);
potencia13=mean(d13.^2)
end
if x==14
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
```

```
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
% antena7
d7=D(7)+n+E(7);
potencia7=mean(d7.^2)
% antena8
d8=D(8)+n+E(8);
potencia8=mean(d8.^2)
% antena9
d9=D(9)+n+E(9);
potencia9=mean(d9.^2)
% antena10
d10=D(10)+n+E(10);
potencia10=mean(d10.^2)
% antena11
d11=D(11)+n+E(11);
potencia11=mean(d11.^2)
% antena12
d12=D(12)+n+E(12);
potencia12=mean(d12.^2)
% antena13
d13=D(13)+n+E(13);
potencia13=mean(d13.^2)
% antena14
d14=D(14)+n+E(14);
potencia14=mean(d14.^2)
end
if x==15
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
```

```
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
% antena7
d7=D(7)+n+E(7);
potencia7=mean(d7.^2)
% antena8
d8=D(8)+n+E(8);
potencia8=mean(d8.^2)
% antena9
d9=D(9)+n+E(9);
potencia9=mean(d9.^2)
% antena10
d10=D(10)+n+E(10);
potencia10=mean(d10.^2)
% antena11
d11=D(11)+n+E(11);
potencia11=mean(d11.^2)
% antena12
d12=D(12)+n+E(12);
potencia12=mean(d12.^2)
% antena13
d13=D(13)+n+E(13);
potencia13=mean(d13.^2)
% antena14
d14=D(14)+n+E(14);
potencia14=mean(d14.^2)
% antena15
d15=D(15)+n+E(15);
potencia15=mean(d15.^2)
end
% calculo de la cota de la contante de ajuste
% cota=1/((orden+1)*potencia); el valor de cota se usa en el lms
%evalua 70 puntos por corrida ( N - orden 70 = 80 - 10 )
N=800;  % N=80
%begin of the algorithm
%forgetting factor valores recomendados por la biblioteca matlab
lamda = 0.999994115 ;% lamda= 0.9995
%initial P matrix
delta=100000000000000 % delta = 1e10 ;
P = delta * eye (orden);
w = zeros ( orden  , 1 ) ;
```

```
for n = orden : N
    u = entrada(n:-1:n-orden+1);
    phi = u' * P ;
    k = phi'/(lamda + phi * u );
    y(n)=w' * u;
    e(n) = senal_referencia(n) - y(n) ;
    w = (w) + k * e(n) ;
    % w=w/10;
    P = ( P - k * phi ) / lamda;
    % ajuste del ploteo, pesos
    Recordedw(1:orden,n)=w;
end
%check of results
for n =  N+1 : totallength
    u = entrada(n:-1:n-orden+1) ;
    y(n) = w' * u ;
    e(n) = senal_referencia(n) - y(n);
end
hold on
plot(senal_referencia)
plot(y,'r');
title('salida del sistema ') ;
xlabel('numero de muestras')
ylabel('valor estimado a la salida')
figure
semilogy((abs(e))) ;
title('curva de error') ;
xlabel('numero de muestras');
ylabel('valor del error');
figure
plot(Recordedw(1:orden,orden:N)');
title('convergencia de los pesos estimados') ;
xlabel('numero de muestras');
ylabel('valor de los pesos');
axis([1 N-orden min(min(Recordedw(1:orden,orden:N)'))
max(max(Recordedw(1:orden,orden:N)')) ]);
hold off
figure
plot(h, 'k+')
hold on
plot(w/10, 'r*')
legend('pesos actuales','pesos estimados');
title('Comparacion de los pesos actuales con los pesos estimados') ;
```

```
function[caratula]=problema2_rls;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
% Algoritmo RLS
% Autores: Juan Alvarez y Maribel chuez
%
%Escuela Superior Politecnica del Litoral
%Facultad de Electricidad y Computacion
%Guayaquil, Ecuador
% email :juan_alvarez8@latinmail.com
% email : mchuez@ceibo.espol.edu.ec
% Webpage : proximamente
% Date    : Miercoles 13 de Mayo del 2009
% Version : 1.0.2
% Referencias : S. Haykin, Adaptive Filter Theory. 3rd edition, Upper
Saddle River, NJ: Prentice-Hall, 1996.
%
%clear all % esta funcion borra los valores anteriores que esten
memoria %
%close all % esta funcion borra cualquier ventana anterior activa %
%hold off   % esta funcion encera cualquier grafico que estuviera
activo %
fprintf('*****************************************************.\n')
fprintf('*    ESCUELA SUPERIOR POLITECNICA DEL LITORAL        *.\n')
fprintf('*                                                    *.\n')
fprintf('* PROBLEMA2_rls.- Determinar los pesos, el numero de *.\n')
fprintf('* iteraciones y la potencia de un grupo de antenas   *.\n')
fprintf('* inteligentes utilizando el algoritmo RLS (algoritmo*.\n')
fprintf('* minimo cuadrado recursivo, las antenas estan distri*.\n')
fprintf('* buidas en forma de celdas                          *.\n')
fprintf('* para lo cual requeriremos de los siguientes datos: *.\n')
fprintf('*                                                    *.\n')
fprintf('*  • la señal aplicada a la entrada del filtro,      *.\n')
fprintf('*   ej  Xn=sin((2*pi*k)/N)                           *.\n')
fprintf('*  • el numero de antenas  x                         *.\n')
fprintf('*  • la señal de referencia, d(k)=cos((2*pi*k)/N);   *.\n')
fprintf('*  • el ruido de las diferentes antenas, (hay que    *.\n')
fprintf('*  tener presente que el algoritmo RLS en su         *.\n')
fprintf('*  estructura internamente produce un ruido que puede*.\n')
fprintf('*  ser de 2, 3 y hasta 10 %  aparte de este ruido    *.\n')
fprintf('*  generado por el mismo programa RLS se puede       *.\n')
fprintf('*  ingresar ruidos adicionales en el programa a la   *.\n')
fprintf('*  entrada o a la salida del programa.  n(f)         *.\n')
fprintf('*  • el orden del filtro FIR cuyos coeficientes se   *.\n')
fprintf('*  van a ajustar, orden                              *.\n')
fprintf('*  • el número de iteraciones en el aprendizaje, N   *.\n')
fprintf('*  (este número se hará  hasta que cumpla el error   *.\n')
fprintf('*  • la constante de ajuste   u                      *.\n')
```

```
fprintf('*  • el vector de pesos inicial. Los voltajes de las *.\n')
fprintf('*  antenas V(f)                                       *.\n')
fprintf('*  • el error deseado puede ser ingresado por teclado*.\n')
fprintf('*    error                                            *.\n')
fprintf('*                                                     *.\n')
fprintf('*  A su salida devolverá:                             *.\n')
fprintf('*  • la salida en cada iteración durante el proceso   *.\n')
fprintf('*  de ajuste, y(t)                                    *.\n')
fprintf('*  • la curva de aprendizaje (grafico de la curva)    *.\n')
fprintf('*  • los vectores de pesos obtenidos durante el pro-  *.\n')
fprintf('*  ceso. W(t)                                         *.\n')
fprintf('*  • la Potencia de la antena P(k)                    *.\n')
fprintf('*  • el numero de iteraciones en que converge la      *.\n')
fprintf('*    curva y se cumple el valor del error             *.\n')
fprintf('*                                                     *.\n')
fprintf('* alumno: Juan Francisco Alvarez Alvarado             *.\n')
fprintf('* alumno: Maribel del Rosaria Chuez Gonzales          *.\n')
fprintf('*                                                     *.\n')
fprintf('* Profesor :Ing. Pedro Vargas                         *.\n')
fprintf('*                                                     *.\n')
fprintf('*******************************************************.\n')
t=input('teclee enter para proseguir');
 clc
% ingresamos el orden del filtro, cuyo coeficiente se va ha ajustar%
orden=input(' ingrese el orden del filtro, cuyo coeficiente se va ha
ajustar....')
% valores recomendado por la central matlab para los pesos en una
función gaussiana son
% h=[0.097631   0.287310   0.335965   0.220981 0.096354 0.017183
% - 0.015917 -0.020735  -0.014243  -0.006517 -0.001396   0.000856
%   0.001272  0.000914   0.000438 0.000108 -0.000044  -0.00008
% -0.000058 -0.000029];
% vamos a considerar que todas las antenas tienen la misma señal a
% la entrada del filtro por ejemplo  entrada=sin((2*pi*k)/N); o
% entrada=randn(N,1),etc
% en el presente programa se usa una señal gaussiana aleatoria
% ingresamos el numero de antenas del sistema
disp(' ingrese el numero de antenas entre 1 y 15');
x=input(' ingrese el numero de antenas entre
1,2,3,4,5,6,7,8,9,10,11,12,13,14 o 15....');
if x==1
    D(1)=input('ingrese el voltaje1 (en V)..... voltaje1=');
end

if x==2
    D(1)=input('ingrese el voltaje1 (en V)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
end
```

```
if x==3
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
end

if x==4
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
end
if x==5
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
end
if x==6
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
end
if x==7
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
    D(7)=input(' ingrese la voltaje7 (en v)..... voltaje7=');
end
if x==8
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
    D(7)=input(' ingrese la voltaje7 (en v)..... voltaje7=');
    D(8)=input(' ingrese la voltaje8 (en v)..... voltaje8=');
end
if x==9
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
```

```
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
    D(7)=input(' ingrese la voltaje7 (en v)..... voltaje7=');
    D(8)=input(' ingrese la voltaje8 (en v)..... voltaje8=');
    D(9)=input(' ingrese la voltaje9 (en v)..... voltaje9=');
end
if x==10
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
    D(7)=input(' ingrese la voltaje7 (en v)..... voltaje7=');
    D(8)=input(' ingrese la voltaje8 (en v)..... voltaje8=');
    D(9)=input(' ingrese la voltaje9 (en v)..... voltaje9=');
    D(10)=input(' ingrese la voltaje10 (en v)..... voltaje10=');
end
if x==11
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
    D(7)=input(' ingrese la voltaje7 (en v)..... voltaje7=');
    D(8)=input(' ingrese la voltaje8 (en v)..... voltaje8=');
    D(9)=input(' ingrese la voltaje9 (en v)..... voltaje9=');
    D(10)=input(' ingrese la voltaje10 (en v)..... voltaje10=');
    D(11)=input(' ingrese la voltaje11 (en v)..... voltaje11=');
end
if x==12
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
    D(7)=input(' ingrese la voltaje7 (en v)..... voltaje7=');
    D(8)=input(' ingrese la voltaje8 (en v)..... voltaje8=');
    D(9)=input(' ingrese la voltaje9 (en v)..... voltaje9=');
    D(10)=input(' ingrese la voltaje10 (en v)..... voltaje10=');
    D(11)=input(' ingrese la voltaje11 (en v)..... voltaje11=');
    D(12)=input(' ingrese la voltaje12 (en v)..... voltaje12=');
end
if x==13
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
```

```
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
    D(7)=input(' ingrese la voltaje7 (en v)..... voltaje7=');
    D(8)=input(' ingrese la voltaje8 (en v)..... voltaje8=');
    D(9)=input(' ingrese la voltaje9 (en v)..... voltaje9=');
    D(10)=input(' ingrese la voltaje10 (en v)..... voltaje10=');
    D(11)=input(' ingrese la voltaje11 (en v)..... voltaje11=');
    D(12)=input(' ingrese la voltaje12 (en v)..... voltaje12=');
    D(13)=input(' ingrese la voltaje13 (en v)..... voltaje13=');
end
if x==14
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
    D(7)=input(' ingrese la voltaje7 (en v)..... voltaje7=');
    D(8)=input(' ingrese la voltaje8 (en v)..... voltaje8=');
    D(9)=input(' ingrese la voltaje9 (en v)..... voltaje9=');
    D(10)=input(' ingrese la voltaje10 (en v)..... voltaje10=');
    D(11)=input(' ingrese la voltaje11 (en v)..... voltaje11=');
    D(12)=input(' ingrese la voltaje12 (en v)..... voltaje12=');
    D(13)=input(' ingrese la voltaje13 (en v)..... voltaje13=');
    D(14)=input(' ingrese la voltaje14 (en v)..... voltaje14=');
end
if x==15
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
    D(7)=input(' ingrese la voltaje7 (en v)..... voltaje7=');
    D(8)=input(' ingrese la voltaje8 (en v)..... voltaje8=');
    D(9)=input(' ingrese la voltaje9 (en v)..... voltaje9=');
    D(10)=input(' ingrese la voltaje10 (en v)..... voltaje10=');
    D(11)=input(' ingrese la voltaje11 (en v)..... voltaje11=');
    D(12)=input(' ingrese la voltaje12 (en v)..... voltaje12=');
    D(13)=input(' ingrese la voltaje13 (en v)..... voltaje13=');
    D(14)=input(' ingrese la voltaje14 (en v)..... voltaje14=');
    D(15)=input(' ingrese la voltaje15 (en v)..... voltaje15=');
end

% ingresamos el ruido que presenta cada antena del sistema
% Si existe ruido adicional al ocasionado por el sistema
```

```
if x==1
    E(1)=input(' ingrese el ruido en la antena1.....ruido1=');
    ruido=E(1);
end

if x==2
    E(1)=input('ingrese el ruido en la antena1.....ruido1==');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    ruido=(E(1)+E(2))/2;
end

if x==3
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    ruido=(E(1)+E(2)+E(3))/3;
end

if x==4
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    ruido=(E(1)+E(2)+E(3)+E(4))/4;
end
if x==5
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    ruido=(E(1)+E(2)+E(3)+E(4)+E(5))/5;
end
if x==6
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6))/6;
end
if x==7
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
```

```
    E(7)=input('ingrese el ruido en la antena7.....ruido7=');
    ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6)+E(7))/7;
end
if x==8
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    E(7)=input('ingrese el ruido en la antena7.....ruido7=');
    E(8)=input('ingrese el ruido en la antena8.....ruido8=');
    ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6)+E(7)+E(8))/8;
end
if x==9
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    E(7)=input('ingrese el ruido en la antena7.....ruido7=');
    E(8)=input('ingrese el ruido en la antena8.....ruido8=');
    E(9)=input('ingrese el ruido en la antena9.....ruido9=');
    ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6)+E(7)+E(8)+E(9))/9;
end
if x==10
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    E(7)=input('ingrese el ruido en la antena7.....ruido7=');
    E(8)=input('ingrese el ruido en la antena8.....ruido8=');
    E(9)=input('ingrese el ruido en la antena9.....ruido9=');
    E(10)=input('ingrese el ruido en la antena10.....ruido10=');
    ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6)+E(7)+E(8)+E(9)+E(10))/10;
end
if x==11
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    E(7)=input('ingrese el ruido en la antena7.....ruido7=');
    E(8)=input('ingrese el ruido en la antena8.....ruido8=');
    E(9)=input('ingrese el ruido en la antena9.....ruido9=');
```

```
    E(10)=input('ingrese el ruido en la antena10.....ruido10=');
    E(11)=input('ingrese el ruido en la antena11.....ruido11=');

ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6)+E(7)+E(8)+E(9)+E(10)+E(11))/11;
end
if x==12
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    E(7)=input('ingrese el ruido en la antena7.....ruido7=');
    E(8)=input('ingrese el ruido en la antena8.....ruido8=');
    E(9)=input('ingrese el ruido en la antena9.....ruido9=');
    E(10)=input('ingrese el ruido en la antena10.....ruido10=');
    E(11)=input('ingrese el ruido en la antena11.....ruido11=');
    E(12)=input('ingrese el ruido en la antena12.....ruido12=');

ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6)+E(7)+E(8)+E(9)+E(10)+E(11)+E(12))/
12;
end
if x==13
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    E(7)=input('ingrese el ruido en la antena7.....ruido7=');
    E(8)=input('ingrese el ruido en la antena8.....ruido8=');
    E(9)=input('ingrese el ruido en la antena9.....ruido9=');
    E(10)=input('ingrese el ruido en la antena10.....ruido10=');
    E(11)=input('ingrese el ruido en la antena11.....ruido11=');
    E(12)=input('ingrese el ruido en la antena12.....ruido12=');
    E(13)=input('ingrese el ruido en la antena13.....ruido13=');

ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6)+E(7)+E(8)+E(9)+E(10)+E(11)+E(12)+E
(13))/13;
end
if x==14
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    E(7)=input('ingrese el ruido en la antena7.....ruido7=');
    E(8)=input('ingrese el ruido en la antena8.....ruido8=');
```

```
    E(9)=input('ingrese el ruido en la antena9.....ruido9=');
    E(10)=input('ingrese el ruido en la antena10.....ruido10=');
    E(11)=input('ingrese el ruido en la antena11.....ruido11=');
    E(12)=input('ingrese el ruido en la antena12.....ruido12=');
    E(13)=input('ingrese el ruido en la antena13.....ruido13=');
    E(14)=input('ingrese el ruido en la antena14.....ruido14=');

ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6)+E(7)+E(8)+E(9)+E(10)+E(11)+E(12)+E
(13)+E(14))/14;
end
if x==15
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    E(7)=input('ingrese el ruido en la antena7.....ruido7=');
    E(8)=input('ingrese el ruido en la antena8.....ruido8=');
    E(9)=input('ingrese el ruido en la antena9.....ruido9=');
    E(10)=input('ingrese el ruido en la antena10.....ruido10=');
    E(11)=input('ingrese el ruido en la antena11.....ruido11=');
    E(12)=input('ingrese el ruido en la antena12.....ruido12=');
    E(13)=input('ingrese el ruido en la antena13.....ruido13=');
    E(14)=input('ingrese el ruido en la antena14.....ruido14=');
    E(15)=input('ingrese el ruido en la antena15.....ruido15=');

ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6)+E(7)+E(8)+E(9)+E(10)+E(11)+E(12)+E
(13)+E(14)+E(15))/15;
end

opcion_ruido=2;
disp('si desea que el ruido adicional vaya a la señal a la entrada
digite 1');
disp(' de lo contrario se considerara que el ruido adicional se
agregara a la señal de referencia');
opcion_ruido=input('opcion_ruido=');
error=input('ingrese el error del sistema=');
% numero de iteraciones del proceso hasta que se cumpla el error
N=2000;
j=0;

if x==1
    h=[D(1)]
    p=[D(1)]
end
if x==2
    h=[D(1);D(2)]
    p=[(D(1)+D(2))/2]+[0.8*(D(1)+D(2))/2];
```

```
end
if x==3
    h=[D(1);D(2);D(3)]
    p=[(D(1)+D(2)+D(3))/3]+[0.8*(D(1)+D(2)+D(3))/3];
end
if x==4
    h=[D(1);D(2);D(3);D(4)]
    p=[(D(1)+D(2)+D(3)+D(4))/4]+[0.8*(D(1)+D(2)+D(3)+D(4))/4];
end
if x==5
    h=[D(1);D(2);D(3);D(4);D(5)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5))/5]+[0.8*(D(1)+D(2)+D(3)+D(4)+D(5))/5];
end
if x==6
    h=[D(1);D(2);D(3);D(4);D(5);D(6)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6))/6]+[0.8*(D(1)+D(2)+D(3)+D(4)+D(5)+D(
6))/6];
end
if x==7
    h=[D(1);D(2);D(3);D(4);D(5);D(6);D(7)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7))/7]+[0.8*(D(1)+D(2)+D(3)+D(4)+D(
5)+D(6)+D(7))/7];
end
if x==8
    h=[D(1);D(2);D(3);D(4);D(5);D(6);D(7);D(8)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8))/8]+[0.8*(D(1)+D(2)+D(3)+D(
4)+D(5)+D(6)+D(7)+D(8))/8];
end
if x==9
    h=[D(1);D(2);D(3);D(4);D(5);D(6);D(7);D(8);D(9)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9))/9]+[0.8*(D(1)+D(2)+D(
3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9))/9];
end
if x==10
    h=[D(1);D(2);D(3);D(4);D(5);D(6);D(7);D(8);D(9);D(10)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10))/10]+[0.8*(D(1)+
D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10))/10];
end
if x==11
    h=[D(1);D(2);D(3);D(4);D(5);D(6);D(7);D(8);D(9);D(10);D(11)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10)+D(11))/11]+[0.8*
(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10)+D(11))/11];
```

```
end
if x==12
    h=[D(1);D(2);D(3);D(4);D(5);D(6);D(7);D(8);D(9);D(10);D(11);D(12)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10)+D(11)+D(12))/12]
+[0.8*(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10)+D(11)+D(12))/
12];
end
if x==13

h=[D(1);D(2);D(3);D(4);D(5);D(6);D(7);D(8);D(9);D(10);D(11);D(12);D(13)
]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10)+D(11)+D(12)+D(13
))/13]+[0.8*(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10)+D(11)+D
(12)+D(13))/13];
end
if x==14

h=[D(1);D(2);D(3);D(4);D(5);D(6);D(7);D(8);D(9);D(10);D(11);D(12);D(13)
;D(14)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10)+D(11)+D(12)+D(13
)+D(14))/14]+[0.8*(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10)+D
(11)+D(12)+D(13)+D(14))/14];
end
if x==15

h=[D(1);D(2);D(3);D(4);D(5);D(6);D(7);D(8);D(9);D(10);D(11);D(12);D(13)
;D(14);D(15)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10)+D(11)+D(12)+D(13
)+D(14)+D(15))/15]+[0.8*(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D
(10)+D(11)+D(12)+D(13)+D(14)+D(15))/15];
end
j=0;
N=2000; % numero de muestras
% vamos a considerar que todas las antenas tienen la misma señal a
% la entrada del filtro por ejemplo  entrada=randn(N,1)
entrada=randn(N,1);
    %senal de entrada

if opcion_ruido==1
    entrada=entrada+ruido; % aqui agregamos el ruido adicional a la
entrada
end
n=sqrt(0.05)*randn(N,1); % ruido a la entrada del filtro, genera 2000
valores
% aleatorios de muestra
```

```
[b,a] = butter(2,0.25); % esta funcion calcula los valores de a y b
Gz = tf(b,a,-1); % la funcion de transferencia tf funciona en matlab 7
% esta funcion permite resolver la transformada z.
%Si no se tiene acceso a la caja de herramientas del matlab se puede
cargar valores de la central matlab usando el sample data
%load IIRsampledata;
%entrada= IIRsampledata(1:2000);
%d= IIRsampledata(1:2000);
% se puede usar ldiv para hallar las aproximaciones de pesos para un %
filtro IIR (filtro de respuesta infinita)
% tambien se puede evaluar  y=h*u
% ldiv es una funcion que nos permite hallar la inversa de la
% transformada Z (Matlab central file exchange)
% para hallar los valores de orden de los pesos se puede utilizar la
% siguiente formula
% use h=ldiv(b,a,sysorder)'; ==> here we use sysorder == 10

h=h(1:orden);
y = lsim(Gz,entrada);% simula respuestas arbitrarias de entrada,
relacion la señal Gz con la entrada
% para agregar el ruido, ruido adicional que genera el algoritmo rls
n = n * std(y)/(10*std(n));
senal_referencia = y + n; % en la mayoría de los programas lms, rls,
filtros se coloca el ruido del sistema en la señal de referncia

if opcion_ruido>1
    senal_referencia=senal_referencia+ruido; % agrega el ruido
adicional
    % a la salida del filtro
end
totallength=size(senal_referencia,1);% nos da el numero de valores
% de la función de entrada conociendo los voltajes de las antenas
% podemos determinar la potencia en ellas
% potencia de la señal a la salida (potencia del sistema
potencia=mean(senal_referencia.^2)
if x==1
% tambien calcularemos la potencia para cada una de las antenas
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
end
if x==2
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
end
```

```
if x==3
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
end
if x==4
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
end
if x==5
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
end
if x==6
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
```

```
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
end
if x==7
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
% antena7
d7=D(7)+n+E(7);
potencia7=mean(d7.^2)
end
if x==8
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
```

```
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
% antena7
d7=D(7)+n+E(7);
potencia7=mean(d7.^2)
% antena8
d8=D(8)+n+E(8);
potencia8=mean(d8.^2)
end
if x==9
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
% antena7
d7=D(7)+n+E(7);
potencia7=mean(d7.^2)
% antena8
d8=D(8)+n+E(8);
potencia8=mean(d8.^2)
% antena9
d9=D(9)+n+E(9);
potencia9=mean(d9.^2)
end
if x==10
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
```

```
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
% antena7
d7=D(7)+n+E(7);
potencia7=mean(d7.^2)
% antena8
d8=D(8)+n+E(8);
potencia8=mean(d8.^2)
% antena9
d9=D(9)+n+E(9);
potencia9=mean(d9.^2)
% antena10
d10=D(10)+n+E(10);
potencia9=mean(d10.^2)
end
if x==11
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
% antena7
d7=D(7)+n+E(7);
potencia7=mean(d7.^2)
% antena8
d8=D(8)+n+E(8);
```

```
potencia8=mean(d8.^2)
% antena9
d9=D(9)+n+E(9);
potencia9=mean(d9.^2)
% antena10
d10=D(10)+n+E(10);
potencia10=mean(d10.^2)
% antena11
d11=D(11)+n+E(11);
potencia11=mean(d11.^2)
end
if x==12
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
% antena7
d7=D(7)+n+E(7);
potencia7=mean(d7.^2)
% antena8
d8=D(8)+n+E(8);
potencia8=mean(d8.^2)
% antena9
d9=D(9)+n+E(9);
potencia9=mean(d9.^2)
% antena10
d10=D(10)+n+E(10);
potencia10=mean(d10.^2)
% antena11
d11=D(11)+n+E(11);
potencia11=mean(d11.^2)
% antena12
d12=D(12)+n+E(12);
potencia12=mean(d12.^2)
end
```

```
if x==13
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
% antena7
d7=D(7)+n+E(7);
potencia7=mean(d7.^2)
% antena8
d8=D(8)+n+E(8);
potencia8=mean(d8.^2)
% antena9
d9=D(9)+n+E(9);
potencia9=mean(d9.^2)
% antena10
d10=D(10)+n+E(10);
potencia10=mean(d10.^2)
% antena11
d11=D(11)+n+E(11);
potencia11=mean(d11.^2)
% antena12
d12=D(12)+n+E(12);
potencia12=mean(d12.^2)
% antena13
d13=D(13)+n+E(13);
potencia13=mean(d13.^2)
end
if x==14
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
```

```
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
% antena7
d7=D(7)+n+E(7);
potencia7=mean(d7.^2)
% antena8
d8=D(8)+n+E(8);
potencia8=mean(d8.^2)
% antena9
d9=D(9)+n+E(9);
potencia9=mean(d9.^2)
% antena10
d10=D(10)+n+E(10);
potencia10=mean(d10.^2)
% antena11
d11=D(11)+n+E(11);
potencia11=mean(d11.^2)
% antena12
d12=D(12)+n+E(12);
potencia12=mean(d12.^2)
% antena13
d13=D(13)+n+E(13);
potencia13=mean(d13.^2)
% antena14
d14=D(14)+n+E(14);
potencia14=mean(d14.^2)
end
if x==15
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
```

```
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
% antena7
d7=D(7)+n+E(7);
potencia7=mean(d7.^2)
% antena8
d8=D(8)+n+E(8);
potencia8=mean(d8.^2)
% antena9
d9=D(9)+n+E(9);
potencia9=mean(d9.^2)
% antena10
d10=D(10)+n+E(10);
potencia10=mean(d10.^2)
% antena11
d11=D(11)+n+E(11);
potencia11=mean(d11.^2)
% antena12
d12=D(12)+n+E(12);
potencia12=mean(d12.^2)
% antena13
d13=D(13)+n+E(13);
potencia13=mean(d13.^2)
% antena14
d14=D(14)+n+E(14);
potencia14=mean(d14.^2)
% antena15
d15=D(15)+n+E(15);
potencia15=mean(d15.^2)
end
% calculo de la cota de la contante de ajust
% cota=1/((orden+1)*potencia); el valor de cota se usa en el lms
%evalua 70 puntos por corrida ( N - orden 70 = 80 - 10 )
N=80;   % N=80
%begin of the algorithm
% el filtro del algoritmo rls se calcula con los coeficientes a,b
% con el valor de landa, con el valor de delta y con el numero N de
% seguridad (chequea los resultados hasta N en caso de que los
% valores de los pesos no convergan hay que cambiar estos valores,
% hasta encontrar valores que permitan la convergencia con poco
% error en caso de algunas funciones puede ser bastante pesado
% encontrar los valores correctos
%forgetting factor valores recomendados por la biblioteca matlab
lamda = 1 ;% lamda= 0.9995
%initial P matrix
```

```
delta=10000000000 % delta = 1e10 ;
P = delta * eye (orden);
w = zeros ( orden  , 1 ) ;
for n = orden : N
    u = entrada(n:-1:n-orden+1);
    phi = u' * P ;
    k = phi'/(lamda + phi * u );
    y(n)=w' * u;
    e(n) = senal_referencia(n) - y(n) ;
    j=j+1;
    if e(n)<error
        fprintf('el numero de iteraciones hasta que se cumpla el
error');
        j
    end
    w = (w) + k * e(n) ;
    % w=w/10;
    P = ( P - k * phi ) / lamda;
    % ajuste del ploteo, pesos
    Recordedw(1:orden,n)=w;
end
%check of results
for n =  N+1 : totallength
    u = entrada(n:-1:n-orden+1) ;
    y(n) = w' * u ;
    e(n) = senal_referencia(n) - y(n);
end
hold on
plot(senal_referencia)
plot(y,'r');
title('salida del sistema ') ;
xlabel('numero de muestras')
ylabel('valor estimado a la salida')
figure
semilogy((abs(e))) ;
title('curva de error') ;
xlabel('numero de muestras');
ylabel('valor del error');
figure
plot(Recordedw(1:orden,orden:N)');
title('convergencia de los pesos estimados') ;
xlabel('numero de muestras');
ylabel('valor de los pesos');
axis([1 N-orden min(min(Recordedw(1:orden,orden:N)'))
max(max(Recordedw(1:orden,orden:N)')) ]);
hold off
figure
plot(h, 'k+')
hold on
```

```
plot(w, 'r*')
legend('pesos actuales','pesos estimados');
title('Comparación de los pesos actuales con los pesos estimados') ;
```

```
function[caratula]=problema3_rls;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
% Algoritmo RLS
% Autores: Juan Alvarez , Maribel Chuez
%
%Escuela Superior Politecnica del Litoral
%Facultad de Electricidad y Computacion
%Guayaquil, Ecuador
% email :juan_alvarez8@latinmail.com
% email : mchuez@ceibo.espol.edu.ec
% Webpage : proximamente
% Date    : Viernes 30 de abril del 2009
% Version : 1.0.1
% Referencias : S. Haykin, Adaptive Filter Theory. 3rd edition, Upper
Saddle River, NJ: Prentice-Hall, 1996.
%
%clear all % esta funcion borra los valores anteriores que esten
memoria %
%close all % esta funcion borra cualquier ventana anterior activa %
%hold off   % esta funcion encera cualquier grafico que estuviera
activo %
 fprintf('*****************************************************.\n')
fprintf('*    ESCUELA SUPERIOR POLITECNICA DEL LITORAL        *.\n')
fprintf('*                                                    *.\n')
fprintf('* PROBLEMA3_rls.- Determinar los pesos, el numero de *.\n')
fprintf('* iteraciones y la potencia de un grupo de antenas   *.\n')
fprintf('* inteligentes utilizando el algoritmo RLS (algoritmo*.\n')
fprintf('* minimo cuadrado recursivo, las antenas estan distri*.\n')
fprintf('* buidas en forma de celdas                          *.\n')
fprintf('* para lo cual requeriremos de los siguientes datos: *.\n')
fprintf('*                                                    *.\n')
fprintf('*  • la señal aplicada a la entrada del filtro,      *.\n')
fprintf('*   ej  Xn=sin((2*pi*k)/N)                           *.\n')
fprintf('*  • el numero de antenas  x                         *.\n')
fprintf('*  • la señal de referencia, d(k)=cos((2*pi*k)/N);   *.\n')
fprintf('*  • el ruido de las diferentes antenas, (hay que    *.\n')
fprintf('*  tener presente que el algoritmo RLS en su         *.\n')
fprintf('*  estructura internamente produce un ruido que puede*.\n')
fprintf('*  ser de 2, 3 y hasta 10 %  aparte de este ruido    *.\n')
fprintf('*  generado por el mismo programa RLS se puede       *.\n')
fprintf('*  ingresar ruidos adicionales en el programa a la   *.\n')
fprintf('*  entrada o a la salida del programa.  n(f)         *.\n')
fprintf('*  • el orden del filtro FIR cuyos coeficientes se   *.\n')
fprintf('*  van a ajustar, orden                              *.\n')
fprintf('*  • el número de iteraciones en el aprendizaje, N   *.\n')
fprintf('*  (este número se hará  hasta que cumpla el error   *.\n')
fprintf('*  • la constante de ajuste   u                      *.\n')
```

```
fprintf('*  • el vector de pesos inicial. Los voltajes de las *.\n')
fprintf('*  antenas V(f)                                       *.\n')
fprintf('*  • el error deseado puede ser ingresado por teclado*.\n')
fprintf('*    error                                            *.\n')
fprintf('*                                                     *.\n')
fprintf('*  A su salida devolverá:                             *.\n')
fprintf('*  • la salida en cada iteración durante el proceso   *.\n')
fprintf('*  de ajuste, y(t)                                    *.\n')
fprintf('*  • la curva de aprendizaje (grafico de la curva)    *.\n')
fprintf('*  • los vectores de pesos obtenidos durante el pro-  *.\n')
fprintf('*  ceso. W(t)                                         *.\n')
fprintf('*  • la Potencia de la antena P(k)                    *.\n')
fprintf('*  • el numero de iteraciones en que converge la      *.\n')
fprintf('*    curva y se cumple el valor del error             *.\n')
fprintf('*                                                     *.\n')
fprintf('* alumno: Juan Francisco Alvarez Alvarado             *.\n')
fprintf('* alumno: Maribel del Rosario Chuez Gonzales          *.\n')
fprintf('*                                                     *.\n')
fprintf('* Profesor :Ing. Pedro Vargas                         *.\n')
fprintf('*                                                     *.\n')
fprintf('*******************************************************.\n')
t=input('teclee enter para proseguir');
 clc
% ingresamos el orden del filtro, cuyo coeficiente se va ha ajustar%
orden=input(' ingrese el orden del filtro, cuyo coeficiente se va ha
ajustar....')

% ingresamos el numero de antenas del sistema
disp(' ingrese el numero de antenas entre 1 y 15');
x=input(' ingrese el numero de antenas entre
1,2,3,4,5,6,7,8,9,10,11,12,13,14 o 15....');
if x==1
    D(1)=input('ingrese el voltaje1 (en V)..... voltaje1=');
end

if x==2
    D(1)=input('ingrese el voltaje1 (en V)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
end

if x==3
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
end

if x==4
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
```

```
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
end
if x==5
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
end
if x==6
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
end
if x==7
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
    D(7)=input(' ingrese la voltaje7 (en v)..... voltaje7=');
end
if x==8
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
    D(7)=input(' ingrese la voltaje7 (en v)..... voltaje7=');
    D(8)=input(' ingrese la voltaje8 (en v)..... voltaje8=');
end
if x==9
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
    D(7)=input(' ingrese la voltaje7 (en v)..... voltaje7=');
    D(8)=input(' ingrese la voltaje8 (en v)..... voltaje8=');
    D(9)=input(' ingrese la voltaje9 (en v)..... voltaje9=');
end
if x==10
```

```
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
    D(7)=input(' ingrese la voltaje7 (en v)..... voltaje7=');
    D(8)=input(' ingrese la voltaje8 (en v)..... voltaje8=');
    D(9)=input(' ingrese la voltaje9 (en v)..... voltaje9=');
    D(10)=input(' ingrese la voltaje10 (en v)..... voltaje10=');
end
if x==11
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
    D(7)=input(' ingrese la voltaje7 (en v)..... voltaje7=');
    D(8)=input(' ingrese la voltaje8 (en v)..... voltaje8=');
    D(9)=input(' ingrese la voltaje9 (en v)..... voltaje9=');
    D(10)=input(' ingrese la voltaje10 (en v)..... voltaje10=');
    D(11)=input(' ingrese la voltaje11 (en v)..... voltaje11=');
end
if x==12
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
    D(7)=input(' ingrese la voltaje7 (en v)..... voltaje7=');
    D(8)=input(' ingrese la voltaje8 (en v)..... voltaje8=');
    D(9)=input(' ingrese la voltaje9 (en v)..... voltaje9=');
    D(10)=input(' ingrese la voltaje10 (en v)..... voltaje10=');
    D(11)=input(' ingrese la voltaje11 (en v)..... voltaje11=');
    D(12)=input(' ingrese la voltaje12 (en v)..... voltaje12=');
end
if x==13
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
    D(7)=input(' ingrese la voltaje7 (en v)..... voltaje7=');
    D(8)=input(' ingrese la voltaje8 (en v)..... voltaje8=');
    D(9)=input(' ingrese la voltaje9 (en v)..... voltaje9=');
    D(10)=input(' ingrese la voltaje10 (en v)..... voltaje10=');
```

```
    D(11)=input(' ingrese la voltaje11 (en v)..... voltaje11=');
    D(12)=input(' ingrese la voltaje12 (en v)..... voltaje12=');
    D(13)=input(' ingrese la voltaje13 (en v)..... voltaje13=');
end
if x==14
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
    D(7)=input(' ingrese la voltaje7 (en v)..... voltaje7=');
    D(8)=input(' ingrese la voltaje8 (en v)..... voltaje8=');
    D(9)=input(' ingrese la voltaje9 (en v)..... voltaje9=');
    D(10)=input(' ingrese la voltaje10 (en v)..... voltaje10=');
    D(11)=input(' ingrese la voltaje11 (en v)..... voltaje11=');
    D(12)=input(' ingrese la voltaje12 (en v)..... voltaje12=');
    D(13)=input(' ingrese la voltaje13 (en v)..... voltaje13=');
    D(14)=input(' ingrese la voltaje14 (en v)..... voltaje14=');
end
if x==15
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
    D(7)=input(' ingrese la voltaje7 (en v)..... voltaje7=');
    D(8)=input(' ingrese la voltaje8 (en v)..... voltaje8=');
    D(9)=input(' ingrese la voltaje9 (en v)..... voltaje9=');
    D(10)=input(' ingrese la voltaje10 (en v)..... voltaje10=');
    D(11)=input(' ingrese la voltaje11 (en v)..... voltaje11=');
    D(12)=input(' ingrese la voltaje12 (en v)..... voltaje12=');
    D(13)=input(' ingrese la voltaje13 (en v)..... voltaje13=');
    D(14)=input(' ingrese la voltaje14 (en v)..... voltaje14=');
    D(15)=input(' ingrese la voltaje15 (en v)..... voltaje15=');
end

% ingresamos el ruido que presenta cada antena del sistema
% Si existe ruido adicional al ocasionado por el sistema
if x==1
    E(1)=input(' ingrese el ruido en la antena1.....ruido1=');
    ruido=E(1);
end

if x==2
    E(1)=input('ingrese el ruido en la antena1.....ruido1==');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    ruido=(E(1)+E(2))/2;
```

```
end

if x==3
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    ruido=(E(1)+E(2)+E(3))/3;
end

if x==4
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    ruido=(E(1)+E(2)+E(3)+E(4))/4;
end
if x==5
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    ruido=(E(1)+E(2)+E(3)+E(4)+E(5))/5;
end
if x==6
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6))/6;
end
if x==7
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    E(7)=input('ingrese el ruido en la antena7.....ruido7=');
    ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6)+E(7))/7;
end
if x==8
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
```

```
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    E(7)=input('ingrese el ruido en la antena7.....ruido7=');
    E(8)=input('ingrese el ruido en la antena8.....ruido8=');
    ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6)+E(7)+E(8))/8;
end
if x==9
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    E(7)=input('ingrese el ruido en la antena7.....ruido7=');
    E(8)=input('ingrese el ruido en la antena8.....ruido8=');
    E(9)=input('ingrese el ruido en la antena9.....ruido9=');
    ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6)+E(7)+E(8)+E(9))/9;
end
if x==10
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    E(7)=input('ingrese el ruido en la antena7.....ruido7=');
    E(8)=input('ingrese el ruido en la antena8.....ruido8=');
    E(9)=input('ingrese el ruido en la antena9.....ruido9=');
    E(10)=input('ingrese el ruido en la antena10.....ruido10=');
    ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6)+E(7)+E(8)+E(9)+E(10))/10;
end
if x==11
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    E(7)=input('ingrese el ruido en la antena7.....ruido7=');
    E(8)=input('ingrese el ruido en la antena8.....ruido8=');
    E(9)=input('ingrese el ruido en la antena9.....ruido9=');
    E(10)=input('ingrese el ruido en la antena10.....ruido10=');
    E(11)=input('ingrese el ruido en la antena11.....ruido11=');

ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6)+E(7)+E(8)+E(9)+E(10)+E(11))/11;
end
if x==12
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
```

```
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    E(7)=input('ingrese el ruido en la antena7.....ruido7=');
    E(8)=input('ingrese el ruido en la antena8.....ruido8=');
    E(9)=input('ingrese el ruido en la antena9.....ruido9=');
    E(10)=input('ingrese el ruido en la antena10.....ruido10=');
    E(11)=input('ingrese el ruido en la antena11.....ruido11=');
    E(12)=input('ingrese el ruido en la antena12.....ruido12=');

ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6)+E(7)+E(8)+E(9)+E(10)+E(11)+E(12))/
12;
end
if x==13
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    E(7)=input('ingrese el ruido en la antena7.....ruido7=');
    E(8)=input('ingrese el ruido en la antena8.....ruido8=');
    E(9)=input('ingrese el ruido en la antena9.....ruido9=');
    E(10)=input('ingrese el ruido en la antena10.....ruido10=');
    E(11)=input('ingrese el ruido en la antena11.....ruido11=');
    E(12)=input('ingrese el ruido en la antena12.....ruido12=');
    E(13)=input('ingrese el ruido en la antena13.....ruido13=');

ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6)+E(7)+E(8)+E(9)+E(10)+E(11)+E(12)+E
(13))/13;
end
if x==14
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    E(7)=input('ingrese el ruido en la antena7.....ruido7=');
    E(8)=input('ingrese el ruido en la antena8.....ruido8=');
    E(9)=input('ingrese el ruido en la antena9.....ruido9=');
    E(10)=input('ingrese el ruido en la antena10.....ruido10=');
    E(11)=input('ingrese el ruido en la antena11.....ruido11=');
    E(12)=input('ingrese el ruido en la antena12.....ruido12=');
    E(13)=input('ingrese el ruido en la antena13.....ruido13=');
    E(14)=input('ingrese el ruido en la antena14.....ruido14=');

ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6)+E(7)+E(8)+E(9)+E(10)+E(11)+E(12)+E
(13)+E(14))/14;
```

```
end
if x==15
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    E(7)=input('ingrese el ruido en la antena7.....ruido7=');
    E(8)=input('ingrese el ruido en la antena8.....ruido8=');
    E(9)=input('ingrese el ruido en la antena9.....ruido9=');
    E(10)=input('ingrese el ruido en la antena10.....ruido10=');
    E(11)=input('ingrese el ruido en la antena11.....ruido11=');
    E(12)=input('ingrese el ruido en la antena12.....ruido12=');
    E(13)=input('ingrese el ruido en la antena13.....ruido13=');
    E(14)=input('ingrese el ruido en la antena14.....ruido14=');
    E(15)=input('ingrese el ruido en la antena15.....ruido15=');

ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6)+E(7)+E(8)+E(9)+E(10)+E(11)+E(12)+E
(13)+E(14)+E(15))/15;
end

opcion_ruido=2;
disp('si desea que el ruido adicional vaya a la señal a la entrada
digite 1');
disp(' de lo contrario se considerara que el ruido adicional se
agregara a la señal de referencia');
opcion_ruido=input('opcion_ruido=');
error=input('ingrese el error del sistema=');
% numero de iteraciones del proceso hasta que se cumpla el error
opcion_senal=2;
disp('si desea que la señal de entrada se use como señal de referencia
digite 1');
disp('de lo contrario se considerara otra señal diferente como señal de
referencia');
opcion_senal=input('opcion_senal=');
N=2000;
j=0;
% valores recomendado por la central matlab para los pesos en una
funcion
%
% h=
% vamos a considerar que todas las antenas tienen la misma señal a la
% entrada del filtro por ejemplo  entrada=sin((2*pi*k)/N);
%
if x==1
    h=[D(1)]
    p=[D(1)]
end
```

```
if x==2
    h=[D(1);D(2)]
    p=[(D(1)+D(2))/2]+[0.8*(D(1)+D(2))/2];
end
if x==3
    h=[D(1);D(2);D(3)]
    p=[(D(1)+D(2)+D(3))/3]+[0.8*(D(1)+D(2)+D(3))/3];
end
if x==4
    h=[D(1);D(2);D(3);D(4)]
    p=[(D(1)+D(2)+D(3)+D(4))/4]+[0.8*(D(1)+D(2)+D(3)+D(4))/4];
end
if x==5
    h=[D(1);D(2);D(3);D(4);D(5)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5))/5]+[0.8*(D(1)+D(2)+D(3)+D(4)+D(5))/5];
end
if x==6
    h=[D(1);D(2);D(3);D(4);D(5);D(6)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6))/6]+[0.8*(D(1)+D(2)+D(3)+D(4)+D(5)+D(
6))/6];
end
if x==7
    h=[D(1);D(2);D(3);D(4);D(5);D(6);D(7)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7))/7]+[0.8*(D(1)+D(2)+D(3)+D(4)+D(
5)+D(6)+D(7))/7];
end
if x==8
    h=[D(1);D(2);D(3);D(4);D(5);D(6);D(7);D(8)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8))/8]+[0.8*(D(1)+D(2)+D(3)+D(
4)+D(5)+D(6)+D(7)+D(8))/8];
end
if x==9
    h=[D(1);D(2);D(3);D(4);D(5);D(6);D(7);D(8);D(9)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9))/9]+[0.8*(D(1)+D(2)+D(
3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9))/9];
end
if x==10
    h=[D(1);D(2);D(3);D(4);D(5);D(6);D(7);D(8);D(9);D(10)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10))/10]+[0.8*(D(1)+
D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10))/10];
end
if x==11
    h=[D(1);D(2);D(3);D(4);D(5);D(6);D(7);D(8);D(9);D(10);D(11)]
```

```
p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10)+D(11))/11]+[0.8*
(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10)+D(11))/11];
end
if x==12
    h=[D(1);D(2);D(3);D(4);D(5);D(6);D(7);D(8);D(9);D(10);D(11);D(12)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10)+D(11)+D(12))/12]
+[0.8*(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10)+D(11)+D(12))/
12];
end
if x==13

h=[D(1);D(2);D(3);D(4);D(5);D(6);D(7);D(8);D(9);D(10);D(11);D(12);D(13)
]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10)+D(11)+D(12)+D(13
))/13]+[0.8*(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10)+D(11)+D
(12)+D(13))/13];
end
if x==14

h=[D(1);D(2);D(3);D(4);D(5);D(6);D(7);D(8);D(9);D(10);D(11);D(12);D(13)
;D(14)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10)+D(11)+D(12)+D(13
)+D(14))/14]+[0.8*(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10)+D
(11)+D(12)+D(13)+D(14))/14];
end
if x==15

h=[D(1);D(2);D(3);D(4);D(5);D(6);D(7);D(8);D(9);D(10);D(11);D(12);D(13)
;D(14);D(15)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10)+D(11)+D(12)+D(13
)+D(14)+D(15))/15]+[0.8*(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D
(10)+D(11)+D(12)+D(13)+D(14)+D(15))/15];
end
M=30;
for k=1:400
    entrada(k)=sin((2*pi*k)/M);
    %senal de entrada
end
for k=1:400
    senal_referencia(k)=cos((2*pi*k)/M);
    % en algunos libros la señal de referencia se la llama señal
deseada
end
entrada';
```

```
entrada=entrada';
if opcion_ruido==1
    entrada=entrada+ruido; % aqui agregamos el ruido adicional a la
entrada
end
n=sqrt(0.05)*randn(400,1); % ruido a la entrada del filtro, genera 400
valores
% aleatorios de muestra
[b,a] = butter(2,0.25); % esta funcion calcula los valores de a y b
Gz = tf(b,a,-1); % la funcion de transferencia tf funciona en matlab 7
% esta funcion permite resolver la transformada z.

%Si no se tiene acceso a la caja de herramientas del matlab se puede
cargar valores de
%la central matlab usando el sample data
%load IIRsampledata;
%entrada= IIRsampledata(1:2000);
%d= IIRsampledata(1:2000);

% se puede usar ldiv para hallar las aproximaciones de pesos para un
filtro IIR (filtro de respuesta infinita)
% tambien se puede evaluar  y=h*u
%ldiv es una funcion que nos permite hallar la inversa de la
transformada Z (Matlab central file exchange)
%para hallar los valores de orden de los pesos se puede utilizar la
%siguiente formula
%use h=ldiv(b,a,sysorder)'; ==> here we use sysorder == 10

h=h(1:orden);
y = lsim(Gz,entrada);% simula respuestas arbitrarias de entrada,
relacion la señal Gz
% con la entrada
% para agregar el ruido, ruido adicional que genera el algoritmo rls
n = n * std(y)/(10*std(n));
if opcion_senal==1
senal_referencia = y + n;
end
if opcion_senal>1
    senal_referencia';
    senal_referencia=senal_referencia';
    senal_referencia=senal_referencia+n;
    %
end
if opcion_ruido>1
    senal_referencia=senal_referencia+ruido; % agrega el ruido
adicional
    % a la salida del filtro
end
```

```
totallength=size(senal_referencia,1);% nos da el numero de valores de
la funcion
% de entrada
% conociendo los voltajes de las antenas podemos determinar la potencia
% en ellas
% potencia de la señal a la salida (potencia del sistema
potencia=mean(senal_referencia.^2)
if x==1
% tambien calcularemos la potencia para cada una de las antenas
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
end
if x==2
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia1=mean(d2.^2)
end
if x==3
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia1=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
end
if x==4
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia1=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
end
if x==5
% antena1
d1=D(1)+n+E(1);
```

```
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
end
if x==6
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
end
if x==7
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
```

```
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
% antena7
d7=D(7)+n+E(7);
potencia7=mean(d7.^2)
end
if x==8
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
% antena7
d7=D(7)+n+E(7);
potencia7=mean(d7.^2)
% antena8
d8=D(8)+n+E(8);
potencia8=mean(d8.^2)
end
if x==9
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
```

```
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
% antena7
d7=D(7)+n+E(7);
potencia7=mean(d7.^2)
% antena8
d8=D(8)+n+E(8);
potencia8=mean(d8.^2)
% antena9
d9=D(9)+n+E(9);
potencia9=mean(d9.^2)
end
if x==10
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
% antena7
d7=D(7)+n+E(7);
potencia7=mean(d7.^2)
% antena8
d8=D(8)+n+E(8);
potencia8=mean(d8.^2)
% antena9
d9=D(9)+n+E(9);
potencia9=mean(d9.^2)
% antena10
d10=D(10)+n+E(10);
potencia9=mean(d10.^2)
end
if x==11
% antena1
d1=D(1)+n+E(1);
```

```
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
% antena7
d7=D(7)+n+E(7);
potencia7=mean(d7.^2)
% antena8
d8=D(8)+n+E(8);
potencia8=mean(d8.^2)
% antena9
d9=D(9)+n+E(9);
potencia9=mean(d9.^2)
% antena10
d10=D(10)+n+E(10);
potencia10=mean(d10.^2)
% antena11
d11=D(11)+n+E(11);
potencia11=mean(d11.^2)
end
if x==12
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
```

```
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
% antena7
d7=D(7)+n+E(7);
potencia7=mean(d7.^2)
% antena8
d8=D(8)+n+E(8);
potencia8=mean(d8.^2)
% antena9
d9=D(9)+n+E(9);
potencia9=mean(d9.^2)
% antena10
d10=D(10)+n+E(10);
potencia10=mean(d10.^2)
% antena11
d11=D(11)+n+E(11);
potencia11=mean(d11.^2)
% antena12
d12=D(12)+n+E(12);
potencia12=mean(d12.^2)
end
if x==13
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
% antena7
d7=D(7)+n+E(7);
potencia7=mean(d7.^2)
% antena8
d8=D(8)+n+E(8);
potencia8=mean(d8.^2)
% antena9
d9=D(9)+n+E(9);
potencia9=mean(d9.^2)
```

```
% antena10
d10=D(10)+n+E(10);
potencia10=mean(d10.^2)
% antena11
d11=D(11)+n+E(11);
potencia11=mean(d11.^2)
% antena12
d12=D(12)+n+E(12);
potencia12=mean(d12.^2)
% antena13
d13=D(13)+n+E(13);
potencia13=mean(d13.^2)
end
if x==14
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
% antena7
d7=D(7)+n+E(7);
potencia7=mean(d7.^2)
% antena8
d8=D(8)+n+E(8);
potencia8=mean(d8.^2)
% antena9
d9=D(9)+n+E(9);
potencia9=mean(d9.^2)
% antena10
d10=D(10)+n+E(10);
potencia10=mean(d10.^2)
% antena11
d11=D(11)+n+E(11);
potencia11=mean(d11.^2)
% antena12
d12=D(12)+n+E(12);
```

```
potencia12=mean(d12.^2)
% antena13
d13=D(13)+n+E(13);
potencia13=mean(d13.^2)
% antena14
d14=D(14)+n+E(14);
potencia14=mean(d14.^2)
end
if x==15
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
% antena7
d7=D(7)+n+E(7);
potencia7=mean(d7.^2)
% antena8
d8=D(8)+n+E(8);
potencia8=mean(d8.^2)
% antena9
d9=D(9)+n+E(9);
potencia9=mean(d9.^2)
% antena10
d10=D(10)+n+E(10);
potencia10=mean(d10.^2)
% antena11
d11=D(11)+n+E(11);
potencia11=mean(d11.^2)
% antena12
d12=D(12)+n+E(12);
potencia12=mean(d12.^2)
% antena13
d13=D(13)+n+E(13);
potencia13=mean(d13.^2)
% antena14
```

```
d14=D(14)+n+E(14);
potencia14=mean(d14.^2)
% antena15
d15=D(15)+n+E(15);
potencia15=mean(d15.^2)
end
% calculo de la cota de la contante de ajuste
% cota=1/((orden+1)*potencia);el valor de cota se usa en el lms
%evalua 70 puntos por corrida ( N - orden 70 = 80 - 10 )
N=80;   % N=80
%begin of the algorithm
%forgetting factor valores recomendados por la biblioteca matlab
lamda = 0.999994115 ;% lamda= 0.9995
%initial P matrix
delta=100000000000000 % delta = 1e10 ;
P = delta * eye (orden);
w = zeros ( orden  , 1 ) ;
for n = orden : N
    u = entrada(n:-1:n-orden+1);
    phi = u' * P ;
    k = phi'/(lamda + phi * u );
    y(n)=w' * u;
    e(n) = senal_referencia(n) - y(n) ;
    w = (w) + k * e(n) ;
    % w=w/10;
    P = ( P - k * phi ) / lamda;
    % ajuste del ploteo, pesos
    Recordedw(1:orden,n)=w;
end
%check of results
for n =  N+1 : totallength
    u = entrada(n:-1:n-orden+1) ;
    y(n) = w' * u ;
    e(n) = senal_referencia(n) - y(n);
end
hold on
plot(senal_referencia)
plot(y,'r');
title('salida del sistema ') ;
xlabel('numero de muestras')
ylabel('valor estimado a la salida')
figure
semilogy((abs(e))) ;
title('curva de error') ;
xlabel('numero de muestras');
ylabel('valor del error');
figure
plot(Recordedw(1:orden,orden:N)');
title('convergencia de los pesos estimados') ;
```

```
xlabel('numero de muestras');
ylabel('valor de los pesos');
axis([1 N-orden min(min(Recordedw(1:orden,orden:N)'))
max(max(Recordedw(1:orden,orden:N)')) ]);
hold off
figure
plot(h, 'k+')
hold on
plot(w/10, 'r*')
legend('pesos actuales','pesos estimados');
title('Comparacion de los pesos actuales con los pesos estimados') ;
```

Problema1_cma

```
function[caratula]=problema1_cma;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
% Algoritmo CMA
% Autores: Juan Alvarez y Maribel Chuez
%
%Escuela Superior Politecnica del Litoral
%Facultad de Electricidad y Computación
%Guayaquil, Ecuador
% email :juan_alvarez8@latinmail.com
% email : mchuez@ceibo.espol.edu.ec
% Webpage : proximamente
% Date    : Miercoles 22 de Julio del 2009
% Version : 1.0.1
% Referencias : S. Haykin, Adaptive Filter Theory. 3rd edition, Upper
Saddle River, NJ: Prentice-Hall, 1996.
%
%clear all % esta funcion borra los valores anteriores que esten
memoria %
%close all % esta funcion borra cualquier ventana anterior activa %
%hold off   % esta funcion encera cualquier grafico que estuviera
activo %
 fprintf('*****************************************************.\n')
fprintf('*    ESCUELA SUPERIOR POLITECNICA DEL LITORAL        *.\n')
fprintf('*                                                    *.\n')
fprintf('* PROBLEMA1_CMA.- Determinar los pesos, el numero de *.\n')
fprintf('* iteraciones y la potencia de un grupo de antenas   *.\n')
fprintf('* inteligentes utilizando el algoritmo CMA-RLS       *.\n')
fprintf('* las antenas estan distribuidas en forma de celdas  *.\n')
fprintf('* para lo cual requeriremos de los siguientes datos: *.\n')
fprintf('*                                                    *.\n')
fprintf('*  • la señal aplicada a la entrada del filtro,      *.\n')
fprintf('*   ej  Xn=sin((2*pi*k)/N)                           *.\n')
fprintf('*  • el numero de antenas  x                         *.\n')
fprintf('*  • la señal de referencia, d(k)=cos((2*pi*k)/N);   *.\n')
fprintf('*  • el ruido de las diferentes antenas, (hay que    *.\n')
fprintf('*  tener presente que el algoritmo CMA en su         *.\n')
fprintf('*  estructura internamente produce un ruido que puede*.\n')
fprintf('*  ser de 2, 3 y hasta 10 %  aparte de este ruido    *.\n')
fprintf('*  generado por el mismo programa CMA se puede       *.\n')
fprintf('*  ingresar ruidos adicionales en el programa a la   *.\n')
fprintf('*  entrada o a la salida del programa.  n(f)         *.\n')
fprintf('*  • el orden del filtro FIR cuyos coeficientes se   *.\n')
fprintf('*  van a ajustar, orden                              *.\n')
```

```
fprintf('*  • el número de iteraciones en el aprendizaje, N   *.\n')
fprintf('*  (este número se hará  hasta que cumpla el error   *.\n')
fprintf('*  • la constante de ajuste   u                      *.\n')
fprintf('*  • el vector de pesos inicial. Los voltajes de las *.\n')
fprintf('*  antenas V(f)                                      *.\n')
fprintf('*  • el error deseado puede ser ingresado por teclado*.\n')
fprintf('*    error                                           *.\n')
fprintf('*                                                    *.\n')
fprintf('*  A su salida devolverá:                            *.\n')
fprintf('*  • la salida en cada iteración durante el proceso  *.\n')
fprintf('*  de ajuste, y(t)                                   *.\n')
fprintf('*  • la curva de aprendizaje (grafico de la curva)   *.\n')
fprintf('*  • los vectores de pesos obtenidos durante el pro- *.\n')
fprintf('*  ceso. W(t)                                        *.\n')
fprintf('*  • la Potencia de la antena P(k)                   *.\n')
fprintf('*  • el numero de iteraciones en que converge la     *.\n')
fprintf('*    curva y se cumple el valor del error            *.\n')
fprintf('*                                                    *.\n')
fprintf('* alumno: Juan Francisco Alvarez Alvarado            *.\n')
fprintf('* alumno: Maribel del Rosario Chuez Gonzales         *.\n')
fprintf('*                                                    *.\n')
fprintf('* Profesor :Ing. Pedro Vargas                        *.\n')
fprintf('*                                                    *.\n')
fprintf('******************************************************.\n')
t=input('teclee enter para proseguir');
 clc
% ingresamos el orden del filtro, cuyo coeficiente se va ha ajustar
fprintf('consideraremos un orden = 2  para la segunda etapa de un ')
fprintf('conformador de haz, esto es para que la ecuación de      ')
fprintf('pesos que se utiliza en el programa sea la de un         ')
fprintf('algoritmo CMA, este valor será constante en la ecuación  ')
fprintf('el valor del orden del filtro para determinar los valo   ')
fprintf('res de cota y de la potencia si variaran normalmente como')
fprintf('si trabajaran en un algoritmo RLS                        ')
fprintf('                                                         ')
orden=input(' ingrese el orden del filtro, cuyo coeficiente se va ha
ajustar....')
% ingresamos el numero de antenas del sistema
disp(' restriccion: ingrese un numero de antenas igual al del orden del
filtro ingresado)');
x=input(' ingrese el numero de antenas entre
1,2,3,4,5,6,7,8,9,10,11,12,13,14 o 15....');
if x==1
    D(1)=input('ingrese el voltaje1 (en V)..... voltaje1=');
end

if x==2
    D(1)=input('ingrese el voltaje1 (en V)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
```

```
end

if x==3
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
end

if x==4
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
end
if x==5
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
end
if x==6
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
end
if x==7
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
    D(7)=input(' ingrese la voltaje7 (en v)..... voltaje7=');
end
if x==8
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
    D(7)=input(' ingrese la voltaje7 (en v)..... voltaje7=');
    D(8)=input(' ingrese la voltaje8 (en v)..... voltaje8=');
end
if x==9
```

```
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
    D(7)=input(' ingrese la voltaje7 (en v)..... voltaje7=');
    D(8)=input(' ingrese la voltaje8 (en v)..... voltaje8=');
    D(9)=input(' ingrese la voltaje9 (en v)..... voltaje9=');
end
if x==10
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
    D(7)=input(' ingrese la voltaje7 (en v)..... voltaje7=');
    D(8)=input(' ingrese la voltaje8 (en v)..... voltaje8=');
    D(9)=input(' ingrese la voltaje9 (en v)..... voltaje9=');
    D(10)=input(' ingrese la voltaje10 (en v)..... voltaje10=');
end
if x==11
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
    D(7)=input(' ingrese la voltaje7 (en v)..... voltaje7=');
    D(8)=input(' ingrese la voltaje8 (en v)..... voltaje8=');
    D(9)=input(' ingrese la voltaje9 (en v)..... voltaje9=');
    D(10)=input(' ingrese la voltaje10 (en v)..... voltaje10=');
    D(11)=input(' ingrese la voltaje11 (en v)..... voltaje11=');
end
if x==12
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
    D(7)=input(' ingrese la voltaje7 (en v)..... voltaje7=');
    D(8)=input(' ingrese la voltaje8 (en v)..... voltaje8=');
    D(9)=input(' ingrese la voltaje9 (en v)..... voltaje9=');
    D(10)=input(' ingrese la voltaje10 (en v)..... voltaje10=');
    D(11)=input(' ingrese la voltaje11 (en v)..... voltaje11=');
    D(12)=input(' ingrese la voltaje12 (en v)..... voltaje12=');
end
```

```
if x==13
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
    D(7)=input(' ingrese la voltaje7 (en v)..... voltaje7=');
    D(8)=input(' ingrese la voltaje8 (en v)..... voltaje8=');
    D(9)=input(' ingrese la voltaje9 (en v)..... voltaje9=');
    D(10)=input(' ingrese la voltaje10 (en v)..... voltaje10=');
    D(11)=input(' ingrese la voltaje11 (en v)..... voltaje11=');
    D(12)=input(' ingrese la voltaje12 (en v)..... voltaje12=');
    D(13)=input(' ingrese la voltaje13 (en v)..... voltaje13=');
end
if x==14
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
    D(7)=input(' ingrese la voltaje7 (en v)..... voltaje7=');
    D(8)=input(' ingrese la voltaje8 (en v)..... voltaje8=');
    D(9)=input(' ingrese la voltaje9 (en v)..... voltaje9=');
    D(10)=input(' ingrese la voltaje10 (en v)..... voltaje10=');
    D(11)=input(' ingrese la voltaje11 (en v)..... voltaje11=');
    D(12)=input(' ingrese la voltaje12 (en v)..... voltaje12=');
    D(13)=input(' ingrese la voltaje13 (en v)..... voltaje13=');
    D(14)=input(' ingrese la voltaje14 (en v)..... voltaje14=');
end
if x==15
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
    D(7)=input(' ingrese la voltaje7 (en v)..... voltaje7=');
    D(8)=input(' ingrese la voltaje8 (en v)..... voltaje8=');
    D(9)=input(' ingrese la voltaje9 (en v)..... voltaje9=');
    D(10)=input(' ingrese la voltaje10 (en v)..... voltaje10=');
    D(11)=input(' ingrese la voltaje11 (en v)..... voltaje11=');
    D(12)=input(' ingrese la voltaje12 (en v)..... voltaje12=');
    D(13)=input(' ingrese la voltaje13 (en v)..... voltaje13=');
    D(14)=input(' ingrese la voltaje14 (en v)..... voltaje14=');
    D(15)=input(' ingrese la voltaje15 (en v)..... voltaje15=');
end
```

```
% ingresamos el ruido que presenta cada antena del sistema
% Si existe ruido adicional al ocasionado por el sistema
if x==1
    E(1)=input(' ingrese el ruido en la antena1.....ruido1=');
    ruido=E(1);
end

if x==2
    E(1)=input('ingrese el ruido en la antena1.....ruido1==');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    ruido=(E(1)+E(2))/2;
end

if x==3
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    ruido=(E(1)+E(2)+E(3))/3;
end

if x==4
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    ruido=(E(1)+E(2)+E(3)+E(4))/4;
end
if x==5
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    ruido=(E(1)+E(2)+E(3)+E(4)+E(5))/5;
end
if x==6
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6))/6;
end
if x==7
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
```

```
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    E(7)=input('ingrese el ruido en la antena7.....ruido7=');
    ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6)+E(7))/7;
end
if x==8
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    E(7)=input('ingrese el ruido en la antena7.....ruido7=');
    E(8)=input('ingrese el ruido en la antena8.....ruido8=');
    ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6)+E(7)+E(8))/8;
end
if x==9
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    E(7)=input('ingrese el ruido en la antena7.....ruido7=');
    E(8)=input('ingrese el ruido en la antena8.....ruido8=');
    E(9)=input('ingrese el ruido en la antena9.....ruido9=');
    ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6)+E(7)+E(8)+E(9))/9;
end
if x==10
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    E(7)=input('ingrese el ruido en la antena7.....ruido7=');
    E(8)=input('ingrese el ruido en la antena8.....ruido8=');
    E(9)=input('ingrese el ruido en la antena9.....ruido9=');
    E(10)=input('ingrese el ruido en la antena10.....ruido10=');
    ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6)+E(7)+E(8)+E(9)+E(10))/10;
end
if x==11
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    E(7)=input('ingrese el ruido en la antena7.....ruido7=');
```

```
    E(8)=input('ingrese el ruido en la antena8.....ruido8=');
    E(9)=input('ingrese el ruido en la antena9.....ruido9=');
    E(10)=input('ingrese el ruido en la antena10.....ruido10=');
    E(11)=input('ingrese el ruido en la antena11.....ruido11=');

ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6)+E(7)+E(8)+E(9)+E(10)+E(11))/11;
end
if x==12
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    E(7)=input('ingrese el ruido en la antena7.....ruido7=');
    E(8)=input('ingrese el ruido en la antena8.....ruido8=');
    E(9)=input('ingrese el ruido en la antena9.....ruido9=');
    E(10)=input('ingrese el ruido en la antena10.....ruido10=');
    E(11)=input('ingrese el ruido en la antena11.....ruido11=');
    E(12)=input('ingrese el ruido en la antena12.....ruido12=');

ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6)+E(7)+E(8)+E(9)+E(10)+E(11)+E(12))/
12;
end
if x==13
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    E(7)=input('ingrese el ruido en la antena7.....ruido7=');
    E(8)=input('ingrese el ruido en la antena8.....ruido8=');
    E(9)=input('ingrese el ruido en la antena9.....ruido9=');
    E(10)=input('ingrese el ruido en la antena10.....ruido10=');
    E(11)=input('ingrese el ruido en la antena11.....ruido11=');
    E(12)=input('ingrese el ruido en la antena12.....ruido12=');
    E(13)=input('ingrese el ruido en la antena13.....ruido13=');

ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6)+E(7)+E(8)+E(9)+E(10)+E(11)+E(12)+E
(13))/13;
end
if x==14
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
```

```
    E(7)=input('ingrese el ruido en la antena7.....ruido7=');
    E(8)=input('ingrese el ruido en la antena8.....ruido8=');
    E(9)=input('ingrese el ruido en la antena9.....ruido9=');
    E(10)=input('ingrese el ruido en la antena10.....ruido10=');
    E(11)=input('ingrese el ruido en la antena11.....ruido11=');
    E(12)=input('ingrese el ruido en la antena12.....ruido12=');
    E(13)=input('ingrese el ruido en la antena13.....ruido13=');
    E(14)=input('ingrese el ruido en la antena14.....ruido14=');

ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6)+E(7)+E(8)+E(9)+E(10)+E(11)+E(12)+E
(13)+E(14))/14;
end
if x==15
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    E(7)=input('ingrese el ruido en la antena7.....ruido7=');
    E(8)=input('ingrese el ruido en la antena8.....ruido8=');
    E(9)=input('ingrese el ruido en la antena9.....ruido9=');
    E(10)=input('ingrese el ruido en la antena10.....ruido10=');
    E(11)=input('ingrese el ruido en la antena11.....ruido11=');
    E(12)=input('ingrese el ruido en la antena12.....ruido12=');
    E(13)=input('ingrese el ruido en la antena13.....ruido13=');
    E(14)=input('ingrese el ruido en la antena14.....ruido14=');
    E(15)=input('ingrese el ruido en la antena15.....ruido15=');

ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6)+E(7)+E(8)+E(9)+E(10)+E(11)+E(12)+E
(13)+E(14)+E(15))/15;
end

opcion_ruido=2;
disp('si desea que el ruido adicional vaya a la señal a la entrada
digite 1');
disp(' de lo contrario se considerara que el ruido adicional se
agregara a la señal de referencia');
opcion_ruido=input('opcion_ruido=');
error=input('ingrese el error del sistema=');
% numero de iteraciones del proceso hasta que se cumpla el error
j=0;
% valores recomendado por la central matlab para los pesos en una
funcion
% gaussiana
% h=[0.097631   0.287310   0.335965   0.220981 0.096354 0.017183  -
0.015917
% -0.020735  -0.014243  -0.006517 -0.001396   0.000856   0.001272
0.000914
```

```
% 0.000438 0.000108 -0.000044  -0.00008  -0.000058 -0.000029];
% vamos a considerar que todas las antenas tienen la misma señal a la
% entrada del filtro por ejemplo  entrada=sin((2*pi*k)/N);
% entrada=randn(N,1)
% en el presente programa no se usa una señal gaussiana, estos valores
solo son para referencia
if x==1
    h=[D(1)]
    p=[D(1)]
end
if x==2
    h=[D(1);D(2)]
    p=[(D(1)+D(2))/2]+[0.8*(D(1)+D(2))/2];
end
if x==3
    h=[D(1);D(2);D(3)]
    p=[(D(1)+D(2)+D(3))/3]+[0.8*(D(1)+D(2)+D(3))/3];
end
if x==4
    h=[D(1);D(2);D(3);D(4)]
    p=[(D(1)+D(2)+D(3)+D(4))/4]+[0.8*(D(1)+D(2)+D(3)+D(4))/4];
end
if x==5
    h=[D(1);D(2);D(3);D(4);D(5)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5))/5]+[0.8*(D(1)+D(2)+D(3)+D(4)+D(5))/5];
end
if x==6
    h=[D(1);D(2);D(3);D(4);D(5);D(6)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6))/6]+[0.8*(D(1)+D(2)+D(3)+D(4)+D(5)+D(
6))/6];
end
if x==7
    h=[D(1);D(2);D(3);D(4);D(5);D(6);D(7)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7))/7]+[0.8*(D(1)+D(2)+D(3)+D(4)+D(
5)+D(6)+D(7))/7];
end
if x==8
    h=[D(1);D(2);D(3);D(4);D(5);D(6);D(7);D(8)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8))/8]+[0.8*(D(1)+D(2)+D(3)+D(
4)+D(5)+D(6)+D(7)+D(8))/8];
end
if x==9
    h=[D(1);D(2);D(3);D(4);D(5);D(6);D(7);D(8);D(9)]
```

```
p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9))/9]+[0.8*(D(1)+D(2)+D(
3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9))/9];
end
if x==10
    h=[D(1);D(2);D(3);D(4);D(5);D(6);D(7);D(8);D(9);D(10)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10))/10]+[0.8*(D(1)+
D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10))/10];
end
if x==11
    h=[D(1);D(2);D(3);D(4);D(5);D(6);D(7);D(8);D(9);D(10);D(11)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10)+D(11))/11]+[0.8*
(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10)+D(11))/11];
end
if x==12
    h=[D(1);D(2);D(3);D(4);D(5);D(6);D(7);D(8);D(9);D(10);D(11);D(12)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10)+D(11)+D(12))/12]
+[0.8*(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10)+D(11)+D(12))/
12];
end
if x==13

h=[D(1);D(2);D(3);D(4);D(5);D(6);D(7);D(8);D(9);D(10);D(11);D(12);D(13)
]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10)+D(11)+D(12)+D(13
))/13]+[0.8*(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10)+D(11)+D
(12)+D(13))/13];
end
if x==14

h=[D(1);D(2);D(3);D(4);D(5);D(6);D(7);D(8);D(9);D(10);D(11);D(12);D(13)
;D(14)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10)+D(11)+D(12)+D(13
)+D(14))/14]+[0.8*(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10)+D
(11)+D(12)+D(13)+D(14))/14];
end
if x==15

h=[D(1);D(2);D(3);D(4);D(5);D(6);D(7);D(8);D(9);D(10);D(11);D(12);D(13)
;D(14);D(15)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10)+D(11)+D(12)+D(13
)+D(14)+D(15))/15]+[0.8*(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D
(10)+D(11)+D(12)+D(13)+D(14)+D(15))/15];
```

```
end
M=30;
for k=1:200
    entrada(k)=sin((2*pi*k)/M);
    %senal de entrada, genera k valores de entrada
end
entrada=entrada';
if opcion_ruido==1
    entrada=entrada+ruido; % aqui agregamos el ruido adicional a la
entrada
end
n=sqrt(0.05)*randn(200,1); % ruido a la entrada del filtro, genera 200
valores
% aleatorios de muestra ( si k=200 en n tambien deben agregarse 200 en
la
% formula) se k = 400 debe colocarse 400 en n (determinacion de ruido)
[b,a] = butter(2,0.25); % esta funcion calcula los valores de a y b
Gz = tf(b,a,-1); % la funcion de transferencia tf funciona en matlab 7
% esta funcion permite resolver la transformada z.


h=h(1:orden);
y = lsim(Gz,entrada);% simula respuestas arbitrarias de entrada,
relacion la señal Gz con la entrada
% para agregar el ruido, ruido adicional que genera el algoritmo rls
n = n * std(y)/(10*std(n));
senal_referencia = y + n; % en la mayoría de los programas lms, rls,
filtros se coloca el ruido del sistema en la señal de referencia.
if opcion_ruido>1
    senal_referencia=senal_referencia+ruido; % agrega el ruido
adicional
    % a la salida del filtro
end
totallength=size(senal_referencia,1);% nos da el numero de valores de
la funcion
% de entrada
% conociendo los voltajes de las antenas podemos determinar la potencia
% en ellas
% potencia de la señal a la salida (potencia del sistema
potencia=mean(senal_referencia.^2)
if x==1
% tambien calcularemos la potencia para cada una de las antenas
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
end
if x==2
% antena1
d1=D(1)+n+E(1);
```

```
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
end
if x==3
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
end
if x==4
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
end
if x==5
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
end
if x==6
% antena1
```

```
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
end
if x==7
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
% antena7
d7=D(7)+n+E(7);
potencia7=mean(d7.^2)
end
if x==8
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
```

```
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
% antena7
d7=D(7)+n+E(7);
potencia7=mean(d7.^2)
% antena8
d8=D(8)+n+E(8);
potencia8=mean(d8.^2)
end
if x==9
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
% antena7
d7=D(7)+n+E(7);
potencia7=mean(d7.^2)
% antena8
d8=D(8)+n+E(8);
potencia8=mean(d8.^2)
% antena9
d9=D(9)+n+E(9);
potencia9=mean(d9.^2)
end
if x==10
% antena1
```

```
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
% antena7
d7=D(7)+n+E(7);
potencia7=mean(d7.^2)
% antena8
d8=D(8)+n+E(8);
potencia8=mean(d8.^2)
% antena9
d9=D(9)+n+E(9);
potencia9=mean(d9.^2)
% antena10
d10=D(10)+n+E(10);
potencia9=mean(d10.^2)
end
if x==11
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
```

```
% antena7
d7=D(7)+n+E(7);
potencia7=mean(d7.^2)
% antena8
d8=D(8)+n+E(8);
potencia8=mean(d8.^2)
% antena9
d9=D(9)+n+E(9);
potencia9=mean(d9.^2)
% antena10
d10=D(10)+n+E(10);
potencia10=mean(d10.^2)
% antena11
d11=D(11)+n+E(11);
potencia11=mean(d11.^2)
end
if x==12
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
% antena7
d7=D(7)+n+E(7);
potencia7=mean(d7.^2)
% antena8
d8=D(8)+n+E(8);
potencia8=mean(d8.^2)
% antena9
d9=D(9)+n+E(9);
potencia9=mean(d9.^2)
% antena10
d10=D(10)+n+E(10);
potencia10=mean(d10.^2)
% antena11
d11=D(11)+n+E(11);
```

```
potencia11=mean(d11.^2)
% antena12
d12=D(12)+n+E(12);
potencia12=mean(d12.^2)
end
if x==13
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
% antena7
d7=D(7)+n+E(7);
potencia7=mean(d7.^2)
% antena8
d8=D(8)+n+E(8);
potencia8=mean(d8.^2)
% antena9
d9=D(9)+n+E(9);
potencia9=mean(d9.^2)
% antena10
d10=D(10)+n+E(10);
potencia10=mean(d10.^2)
% antena11
d11=D(11)+n+E(11);
potencia11=mean(d11.^2)
% antena12
d12=D(12)+n+E(12);
potencia12=mean(d12.^2)
% antena13
d13=D(13)+n+E(13);
potencia13=mean(d13.^2)
end
if x==14
% antena1
d1=D(1)+n+E(1);
```

```
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
% antena7
d7=D(7)+n+E(7);
potencia7=mean(d7.^2)
% antena8
d8=D(8)+n+E(8);
potencia8=mean(d8.^2)
% antena9
d9=D(9)+n+E(9);
potencia9=mean(d9.^2)
% antena10
d10=D(10)+n+E(10);
potencia10=mean(d10.^2)
% antena11
d11=D(11)+n+E(11);
potencia11=mean(d11.^2)
% antena12
d12=D(12)+n+E(12);
potencia12=mean(d12.^2)
% antena13
d13=D(13)+n+E(13);
potencia13=mean(d13.^2)
% antena14
d14=D(14)+n+E(14);
potencia14=mean(d14.^2)
end
if x==15
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
```

```
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
% antena7
d7=D(7)+n+E(7);
potencia7=mean(d7.^2)
% antena8
d8=D(8)+n+E(8);
potencia8=mean(d8.^2)
% antena9
d9=D(9)+n+E(9);
potencia9=mean(d9.^2)
% antena10
d10=D(10)+n+E(10);
potencia10=mean(d10.^2)
% antena11
d11=D(11)+n+E(11);
potencia11=mean(d11.^2)
% antena12
d12=D(12)+n+E(12);
potencia12=mean(d12.^2)
% antena13
d13=D(13)+n+E(13);
potencia13=mean(d13.^2)
% antena14
d14=D(14)+n+E(14);
potencia14=mean(d14.^2)
% antena15
d15=D(15)+n+E(15);
potencia15=mean(d15.^2)
end
% calculo de la cota de la contante de ajuste
cota=1/((orden+1)*potencia);
%evalua 70 puntos por corrida ( N - orden 70 = 80 - 10 )
N=100;  % N=80
%begin of the algorithm
%forgetting factor valores recomendados por la biblioteca matlab
lamda = 0.999994115 ;% lamda= 0.9995
%initial P matrix
delta=100000000000000 % delta = 1e10 ;
P = delta * eye (orden);
```

```
% w = zeros ( orden  , 1 ) ;
if x==1
    w=[0.15]
    w=w'
end
if x==2
    w=[ 0.15 0.10]
    w=w'
end
if x==3
    w=[0.15 0.10 0.20 ]
    w=w'
end
if x==4
    w=[0.15 0.10 0.20 0.30]
    w=w'
end
if x==5
    w=[0.15 0.10 0.20 0.30 -0.20]
    w=w'
end
if x==6
    w=[0.15 0.10 0.20 0.30 -0.20 0.14]
    w=w'
end
if x==7
    w=[0.15 0.10 0.20 0.30 -0.20 0.14 0.12]
    w=w'
end
if x==8
    w=[0.15 0.10 0.20 0.30 -0.20 0.14 0.12 0.11]
    w=w'
end
if x==9
    w=[0.15 0.10 0.20 0.30 -0.20 0.14 0.12 0.11 0.05]
    w=w'
end
if x==10
    w=[0.15 0.10 0.20 0.30 -0.20 0.14 0.12 0.11 0.05 -0.05]
    w=w'
end
if x==11
    w=[0.15 0.10 0.20 0.30 -0.20 0.14 0.12 0.11 0.05 -0.05 -0.09]
    w=w'
end
if x==12
    w=[0.15 0.10 0.20 0.30 -0.20 0.14 0.12 0.11 0.05 -0.05 -0.09 0.05]
    w=w'
end
```

```
if x==13
    w=[0.15 0.10 0.20 0.30 -0.20 0.14 0.12 0.11 0.05 -0.05 -0.09 0.05
0.08]
    w=w'
end
if x==14
    w=[0.15 0.10 0.20 0.30 -0.20 0.14 0.12 0.11 0.05 -0.05 -0.09 0.05
0.08 0.06]
    w=w'
end
if x==15
    w=[0.15 0.10 0.20 0.30 -0.20 0.14 0.12 0.11 0.05 -0.05 -0.09 0.05
0.08 0.06 0.02]
    w=w'
end
% T10=[D(1) D(2) D(3) D(4) D(5)]
Rp=1;
for n = orden : N
    u = entrada(n:-1:n-orden+1);
    phi = u' * P ;
    k = phi'/(lamda + phi * u );
    y(n)= w'*u ;
    T(n)=[abs(y(n))].^2;
    T2(n)=Rp-T(n); % Rp valor de la constante cma, si no converge,
    % pruebe con otros valores de Rp, valores de entrada,
    T3=T2(n)*u;
    e=T3*y(n); % e(n) = senal_referencia(n) - y(n); error para RLS
    % e(n)=senal_referencia(n)-y(n);
    T4=e';
    T5=T4*u;
    mu=5*10^-7;
    %if n < 20
    %    mu=cota/2
    %else
    %    mu=cota/4
    %end
    w=(w)-(mu*T5)  % w = (w) + k * e(n);  peso para RLS
    % w=w/10;
    P = ( P - k * phi ) / lamda;
    % ajuste del ploteo, pesos
    Recordedw(1:orden,n)=w;
end
%check of results
for n =  N+1 : totallength
    u = entrada(n:-1:n-orden+1) ;
    y(n) = w' * u ;
    T(n)=[abs(y(n))].^2;
    T2(n)=1-T(n);
    e=[T2(n)*y(n)]*u;    %e(n) = senal_referencia(n) - y(n);rls
```

```
    % e(n)=e(n)*u;
end

hold on
plot(senal_referencia)
plot(y,'r');
title('salida del sistema ') ;
xlabel('numero de muestras')
ylabel('valor estimado a la salida')
figure
semilogy((abs(e))) ;
title('curva de error') ;
xlabel('numero de muestras');
ylabel('valor del error');
figure
plot(Recordedw(1:orden,orden:N)');
title('convergencia de los pesos estimados') ;
xlabel('numero de muestras');
ylabel('valor de los pesos');
axis([1 N-orden min(min(Recordedw(1:orden,orden:N)'))
max(max(Recordedw(1:orden,orden:N)')) ]);
hold off
figure
plot(h, 'k+')
hold on
plot(w, 'r*')
legend('pesos actuales','pesos estimados');
title('Comparacion de los pesos actuales con los pesos estimados') ;
```

```
function[caratula]=problema2_cma;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
% Algoritmo CMA
% Autores: Juan Alvarez y Maribel Chuez
%
%Escuela Superior Politecnica del Litoral
%Facultad de Electricidad y Computación
%Guayaquil, Ecuador
% email :juan_alvarez8@latinmail.com
% email : mchuez@ceibo.espol.edu.ec
% Webpage : proximamente
% Date    : Miercoles 22 de Julio del 2009
% Version : 1.0.1
% Referencias : S. Haykin, Adaptive Filter Theory. 3rd edition, Upper
Saddle River, NJ: Prentice-Hall, 1996.
%
%clear all % esta funcion borra los valores anteriores que esten
memoria %
%close all % esta funcion borra cualquier ventana anterior activa %
%hold off   % esta funcion encera cualquier grafico que estuviera
activo %
 fprintf('*****************************************************.\n')
fprintf('*    ESCUELA SUPERIOR POLITECNICA DEL LITORAL        *.\n')
fprintf('*                                                    *.\n')
fprintf('* PROBLEMA2_CMA.- Determinar los pesos, el numero de *.\n')
fprintf('* iteraciones y la potencia de un grupo de antenas   *.\n')
fprintf('* inteligentes utilizando el algoritmo CMA-RLS       *.\n')
fprintf('* las antenas estan distribuidas en forma de celdas  *.\n')
fprintf('* para lo cual requeriremos de los siguientes datos: *.\n')
fprintf('*                                                    *.\n')
fprintf('*  • la señal aplicada a la entrada del filtro,      *.\n')
fprintf('*   ej  Xn=sin((2*pi*k)/N)                           *.\n')
fprintf('*  • el numero de antenas  x                         *.\n')
fprintf('*  • la señal de referencia, d(k)=cos((2*pi*k)/N);   *.\n')
fprintf('*  • el ruido de las diferentes antenas, (hay que    *.\n')
fprintf('*  tener presente que el algoritmo CMA en su         *.\n')
fprintf('*  estructura internamente produce un ruido que puede*.\n')
fprintf('*  ser de 2, 3 y hasta 10 %  aparte de este ruido    *.\n')
fprintf('*  generado por el mismo programa CMA se puede       *.\n')
fprintf('*  ingresar ruidos adicionales en el programa a la   *.\n')
fprintf('*  entrada o a la salida del programa.  n(f)         *.\n')
fprintf('*  • el orden del filtro FIR cuyos coeficientes se   *.\n')
fprintf('*  van a ajustar, orden                              *.\n')
fprintf('*  • el número de iteraciones en el aprendizaje, N   *.\n')
fprintf('*  (este número se hará  hasta que cumpla el error   *.\n')
fprintf('*  • la constante de ajuste   u                      *.\n')
fprintf('*  • el vector de pesos inicial. Los voltajes de las *.\n')
fprintf('*  antenas V(f)                                      *.\n')
```

```
fprintf('*  • el error deseado puede ser ingresado por teclado*.\n')
fprintf('*    error                                           *.\n')
fprintf('*                                                    *.\n')
fprintf('*  A su salida devolverá:                            *.\n')
fprintf('*  • la salida en cada iteración durante el proceso  *.\n')
fprintf('*  de ajuste, y(t)                                   *.\n')
fprintf('*  • la curva de aprendizaje (grafico de la curva)   *.\n')
fprintf('*  • los vectores de pesos obtenidos durante el pro- *.\n')
fprintf('*  ceso. W(t)                                        *.\n')
fprintf('*  • la Potencia de la antena P(k)                   *.\n')
fprintf('*  • el numero de iteraciones en que converge la     *.\n')
fprintf('*    curva y se cumple el valor del error            *.\n')
fprintf('*                                                    *.\n')
fprintf('* alumno: Juan Francisco Alvarez Alvarado            *.\n')
fprintf('* alumno: Maribel del Rosario Chuez Gonzales         *.\n')
fprintf('*                                                    *.\n')
fprintf('* Profesor :Ing. Pedro Vargas                        *.\n')
fprintf('*                                                    *.\n')
fprintf('******************************************************.\n')
t=input('teclee enter para proseguir');
 clc
% ingresamos el orden del filtro, cuyo coeficiente se va ha ajustar
fprintf('consideraremos un orden = 2  para la segunda etapa de un ')
fprintf('conformador de haz, esto es para que la ecuación de      ')
fprintf('pesos que se utiliza en el programa sea la de un         ')
fprintf('algoritmo CMA, este valor será constante en la ecuación  ')
fprintf('el valor del orden del filtro para determinar los valo   ')
fprintf('res de cota y de la potencia si variaran normalmente como')
fprintf('si trabajaran en un algoritmo RLS                        ')
fprintf('                                                         ')
orden=input(' ingrese el orden del filtro, cuyo coeficiente se va ha
ajustar....')
% ingresamos el numero de antenas del sistema
disp(' restriccion: ingrese un numero de antenas igual al del orden del
filtro ingresado)');
x=input(' ingrese el numero de antenas entre
1,2,3,4,5,6,7,8,9,10,11,12,13,14 o 15....');
if x==1
    D(1)=input('ingrese el voltaje1 (en V)..... voltaje1=');
end

if x==2
    D(1)=input('ingrese el voltaje1 (en V)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
end

if x==3
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
```

```
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
end

if x==4
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
end
if x==5
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
end
if x==6
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
end
if x==7
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
    D(7)=input(' ingrese la voltaje7 (en v)..... voltaje7=');
end
if x==8
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
    D(7)=input(' ingrese la voltaje7 (en v)..... voltaje7=');
    D(8)=input(' ingrese la voltaje8 (en v)..... voltaje8=');
end
if x==9
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
```

```
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
    D(7)=input(' ingrese la voltaje7 (en v)..... voltaje7=');
    D(8)=input(' ingrese la voltaje8 (en v)..... voltaje8=');
    D(9)=input(' ingrese la voltaje9 (en v)..... voltaje9=');
end
if x==10
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
    D(7)=input(' ingrese la voltaje7 (en v)..... voltaje7=');
    D(8)=input(' ingrese la voltaje8 (en v)..... voltaje8=');
    D(9)=input(' ingrese la voltaje9 (en v)..... voltaje9=');
    D(10)=input(' ingrese la voltaje10 (en v)..... voltaje10=');
end
if x==11
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
    D(7)=input(' ingrese la voltaje7 (en v)..... voltaje7=');
    D(8)=input(' ingrese la voltaje8 (en v)..... voltaje8=');
    D(9)=input(' ingrese la voltaje9 (en v)..... voltaje9=');
    D(10)=input(' ingrese la voltaje10 (en v)..... voltaje10=');
    D(11)=input(' ingrese la voltaje11 (en v)..... voltaje11=');
end
if x==12
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
    D(7)=input(' ingrese la voltaje7 (en v)..... voltaje7=');
    D(8)=input(' ingrese la voltaje8 (en v)..... voltaje8=');
    D(9)=input(' ingrese la voltaje9 (en v)..... voltaje9=');
    D(10)=input(' ingrese la voltaje10 (en v)..... voltaje10=');
    D(11)=input(' ingrese la voltaje11 (en v)..... voltaje11=');
    D(12)=input(' ingrese la voltaje12 (en v)..... voltaje12=');
end
if x==13
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
```

```
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
    D(7)=input(' ingrese la voltaje7 (en v)..... voltaje7=');
    D(8)=input(' ingrese la voltaje8 (en v)..... voltaje8=');
    D(9)=input(' ingrese la voltaje9 (en v)..... voltaje9=');
    D(10)=input(' ingrese la voltaje10 (en v)..... voltaje10=');
    D(11)=input(' ingrese la voltaje11 (en v)..... voltaje11=');
    D(12)=input(' ingrese la voltaje12 (en v)..... voltaje12=');
    D(13)=input(' ingrese la voltaje13 (en v)..... voltaje13=');
end
if x==14
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
    D(7)=input(' ingrese la voltaje7 (en v)..... voltaje7=');
    D(8)=input(' ingrese la voltaje8 (en v)..... voltaje8=');
    D(9)=input(' ingrese la voltaje9 (en v)..... voltaje9=');
    D(10)=input(' ingrese la voltaje10 (en v)..... voltaje10=');
    D(11)=input(' ingrese la voltaje11 (en v)..... voltaje11=');
    D(12)=input(' ingrese la voltaje12 (en v)..... voltaje12=');
    D(13)=input(' ingrese la voltaje13 (en v)..... voltaje13=');
    D(14)=input(' ingrese la voltaje14 (en v)..... voltaje14=');
end
if x==15
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
    D(7)=input(' ingrese la voltaje7 (en v)..... voltaje7=');
    D(8)=input(' ingrese la voltaje8 (en v)..... voltaje8=');
    D(9)=input(' ingrese la voltaje9 (en v)..... voltaje9=');
    D(10)=input(' ingrese la voltaje10 (en v)..... voltaje10=');
    D(11)=input(' ingrese la voltaje11 (en v)..... voltaje11=');
    D(12)=input(' ingrese la voltaje12 (en v)..... voltaje12=');
    D(13)=input(' ingrese la voltaje13 (en v)..... voltaje13=');
    D(14)=input(' ingrese la voltaje14 (en v)..... voltaje14=');
    D(15)=input(' ingrese la voltaje15 (en v)..... voltaje15=');
end

% ingresamos el ruido que presenta cada antena del sistema
% Si existe ruido adicional al ocasionado por el sistema
if x==1
    E(1)=input(' ingrese el ruido en la antena1.....ruido1=');
    ruido=E(1);
```

```
end

if x==2
    E(1)=input('ingrese el ruido en la antena1.....ruido1==');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    ruido=(E(1)+E(2))/2;
end

if x==3
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    ruido=(E(1)+E(2)+E(3))/3;
end

if x==4
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    ruido=(E(1)+E(2)+E(3)+E(4))/4;
end
if x==5
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    ruido=(E(1)+E(2)+E(3)+E(4)+E(5))/5;
end
if x==6
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6))/6;
end
if x==7
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    E(7)=input('ingrese el ruido en la antena7.....ruido7=');
    ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6)+E(7))/7;
end
```

```
if x==8
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    E(7)=input('ingrese el ruido en la antena7.....ruido7=');
    E(8)=input('ingrese el ruido en la antena8.....ruido8=');
    ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6)+E(7)+E(8))/8;
end
if x==9
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    E(7)=input('ingrese el ruido en la antena7.....ruido7=');
    E(8)=input('ingrese el ruido en la antena8.....ruido8=');
    E(9)=input('ingrese el ruido en la antena9.....ruido9=');
    ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6)+E(7)+E(8)+E(9))/9;
end
if x==10
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    E(7)=input('ingrese el ruido en la antena7.....ruido7=');
    E(8)=input('ingrese el ruido en la antena8.....ruido8=');
    E(9)=input('ingrese el ruido en la antena9.....ruido9=');
    E(10)=input('ingrese el ruido en la antena10.....ruido10=');
    ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6)+E(7)+E(8)+E(9)+E(10))/10;
end
if x==11
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    E(7)=input('ingrese el ruido en la antena7.....ruido7=');
    E(8)=input('ingrese el ruido en la antena8.....ruido8=');
    E(9)=input('ingrese el ruido en la antena9.....ruido9=');
    E(10)=input('ingrese el ruido en la antena10.....ruido10=');
    E(11)=input('ingrese el ruido en la antena11.....ruido11=');
```

```
ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6)+E(7)+E(8)+E(9)+E(10)+E(11))/11;
end
if x==12
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    E(7)=input('ingrese el ruido en la antena7.....ruido7=');
    E(8)=input('ingrese el ruido en la antena8.....ruido8=');
    E(9)=input('ingrese el ruido en la antena9.....ruido9=');
    E(10)=input('ingrese el ruido en la antena10.....ruido10=');
    E(11)=input('ingrese el ruido en la antena11.....ruido11=');
    E(12)=input('ingrese el ruido en la antena12.....ruido12=');

ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6)+E(7)+E(8)+E(9)+E(10)+E(11)+E(12))/
12;
end
if x==13
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    E(7)=input('ingrese el ruido en la antena7.....ruido7=');
    E(8)=input('ingrese el ruido en la antena8.....ruido8=');
    E(9)=input('ingrese el ruido en la antena9.....ruido9=');
    E(10)=input('ingrese el ruido en la antena10.....ruido10=');
    E(11)=input('ingrese el ruido en la antena11.....ruido11=');
    E(12)=input('ingrese el ruido en la antena12.....ruido12=');
    E(13)=input('ingrese el ruido en la antena13.....ruido13=');

ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6)+E(7)+E(8)+E(9)+E(10)+E(11)+E(12)+E
(13))/13;
end
if x==14
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    E(7)=input('ingrese el ruido en la antena7.....ruido7=');
    E(8)=input('ingrese el ruido en la antena8.....ruido8=');
    E(9)=input('ingrese el ruido en la antena9.....ruido9=');
    E(10)=input('ingrese el ruido en la antena10.....ruido10=');
```

```
    E(11)=input('ingrese el ruido en la antena11.....ruido11=');
    E(12)=input('ingrese el ruido en la antena12.....ruido12=');
    E(13)=input('ingrese el ruido en la antena13.....ruido13=');
    E(14)=input('ingrese el ruido en la antena14.....ruido14=');

ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6)+E(7)+E(8)+E(9)+E(10)+E(11)+E(12)+E
(13)+E(14))/14;
end
if x==15
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    E(7)=input('ingrese el ruido en la antena7.....ruido7=');
    E(8)=input('ingrese el ruido en la antena8.....ruido8=');
    E(9)=input('ingrese el ruido en la antena9.....ruido9=');
    E(10)=input('ingrese el ruido en la antena10.....ruido10=');
    E(11)=input('ingrese el ruido en la antena11.....ruido11=');
    E(12)=input('ingrese el ruido en la antena12.....ruido12=');
    E(13)=input('ingrese el ruido en la antena13.....ruido13=');
    E(14)=input('ingrese el ruido en la antena14.....ruido14=');
    E(15)=input('ingrese el ruido en la antena15.....ruido15=');

ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6)+E(7)+E(8)+E(9)+E(10)+E(11)+E(12)+E
(13)+E(14)+E(15))/15;
end

opcion_ruido=2;
disp('si desea que el ruido adicional vaya a la señal a la entrada
digite 1');
disp(' de lo contrario se considerara que el ruido adicional se
agregara a la señal de referencia');
opcion_ruido=input('opcion_ruido=');
error=input('ingrese el error del sistema=');
% numero de iteraciones del proceso hasta que se cumpla el error
N=400;
j=0;

if x==1
    h=[D(1)]
    p=[D(1)]
end
if x==2
    h=[D(1);D(2)]
    p=[(D(1)+D(2))/2]+[0.8*(D(1)+D(2))/2];
end
if x==3
```

```
    h=[D(1);D(2);D(3)]
    p=[(D(1)+D(2)+D(3))/3]+[0.8*(D(1)+D(2)+D(3))/3];
end
if x==4
    h=[D(1);D(2);D(3);D(4)]
    p=[(D(1)+D(2)+D(3)+D(4))/4]+[0.8*(D(1)+D(2)+D(3)+D(4))/4];
end
if x==5
    h=[D(1);D(2);D(3);D(4);D(5)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5))/5]+[0.8*(D(1)+D(2)+D(3)+D(4)+D(5))/5];
end
if x==6
    h=[D(1);D(2);D(3);D(4);D(5);D(6)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6))/6]+[0.8*(D(1)+D(2)+D(3)+D(4)+D(5)+D(
6))/6];
end
if x==7
    h=[D(1);D(2);D(3);D(4);D(5);D(6);D(7)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7))/7]+[0.8*(D(1)+D(2)+D(3)+D(4)+D(
5)+D(6)+D(7))/7];
end
if x==8
    h=[D(1);D(2);D(3);D(4);D(5);D(6);D(7);D(8)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8))/8]+[0.8*(D(1)+D(2)+D(3)+D(
4)+D(5)+D(6)+D(7)+D(8))/8];
end
if x==9
    h=[D(1);D(2);D(3);D(4);D(5);D(6);D(7);D(8);D(9)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9))/9]+[0.8*(D(1)+D(2)+D(
3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9))/9];
end
if x==10
    h=[D(1);D(2);D(3);D(4);D(5);D(6);D(7);D(8);D(9);D(10)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10))/10]+[0.8*(D(1)+
D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10))/10];
end
if x==11
    h=[D(1);D(2);D(3);D(4);D(5);D(6);D(7);D(8);D(9);D(10);D(11)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10)+D(11))/11]+[0.8*
(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10)+D(11))/11];
end
if x==12
```

```
    h=[D(1);D(2);D(3);D(4);D(5);D(6);D(7);D(8);D(9);D(10);D(11);D(12)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10)+D(11)+D(12))/12]
+[0.8*(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10)+D(11)+D(12))/
12];
end
if x==13

h=[D(1);D(2);D(3);D(4);D(5);D(6);D(7);D(8);D(9);D(10);D(11);D(12);D(13)
]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10)+D(11)+D(12)+D(13
))/13]+[0.8*(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10)+D(11)+D
(12)+D(13))/13];
end
if x==14

h=[D(1);D(2);D(3);D(4);D(5);D(6);D(7);D(8);D(9);D(10);D(11);D(12);D(13)
;D(14)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10)+D(11)+D(12)+D(13
)+D(14))/14]+[0.8*(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10)+D
(11)+D(12)+D(13)+D(14))/14];
end
if x==15

h=[D(1);D(2);D(3);D(4);D(5);D(6);D(7);D(8);D(9);D(10);D(11);D(12);D(13)
;D(14);D(15)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10)+D(11)+D(12)+D(13
)+D(14)+D(15))/15]+[0.8*(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D
(10)+D(11)+D(12)+D(13)+D(14)+D(15))/15];
end
j=0;
N=400; % numero de muestras
% vamos a considerar que todas las antenas tienen la misma señal a la
% entrada del filtro por ejemplo  entrada=randn(N,1)
entrada=randn(N,1);
    %senal de entrada

if opcion_ruido==1
    entrada=entrada+ruido; % aqui agregamos el ruido adicional a la
entrada
end
n=sqrt(0.05)*randn(N,1); % ruido a la entrada del filtro, genera N
valores aleatorios de muestra
[b,a] = butter(2,0.25); % esta funcion calcula los valores de a y b
Gz = tf(b,a,-1); % la funcion de transferencia tf funciona en matlab 7
% esta funcion permite resolver la transformada z.
```

```
h=h(1:orden);
y = lsim(Gz,entrada);% simula respuestas arbitrarias de entrada,
relacion la señal Gz
% con la entrada
% para agregar el ruido, ruido adicional que genera el algoritmo rls
n = n * std(y)/(10*std(n));
senal_referencia = y + n; % En la mayoría de los programas lms, rls,
filtros, se coloca el ruido del sistema en la señal de referencia

if opcion_ruido>1
    senal_referencia=senal_referencia+ruido; % agrega el ruido
adicional
    % a la salida del filtro
end
totallength=size(senal_referencia,1);% nos da el numero de valores de
la funcion
% de entrada
% conociendo los voltajes de las antenas podemos determinar la potencia
% en ellas
% potencia de la señal a la salida (potencia del sistema
potencia=mean(senal_referencia.^2)
if x==1
% tambien calcularemos la potencia para cada una de las antenas
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
end
if x==2
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
end
if x==3
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
end
if x==4
```

```
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
end
if x==5
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
end
if x==6
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
```

```
end
if x==7
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
% antena7
d7=D(7)+n+E(7);
potencia7=mean(d7.^2)
end
if x==8
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
% antena7
d7=D(7)+n+E(7);
potencia7=mean(d7.^2)
% antena8
d8=D(8)+n+E(8);
potencia8=mean(d8.^2)
```

```
end
if x==9
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
% antena7
d7=D(7)+n+E(7);
potencia7=mean(d7.^2)
% antena8
d8=D(8)+n+E(8);
potencia8=mean(d8.^2)
% antena9
d9=D(9)+n+E(9);
potencia9=mean(d9.^2)
end
if x==10
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
```

```
% antena7
d7=D(7)+n+E(7);
potencia7=mean(d7.^2)
% antena8
d8=D(8)+n+E(8);
potencia8=mean(d8.^2)
% antena9
d9=D(9)+n+E(9);
potencia9=mean(d9.^2)
% antena10
d10=D(10)+n+E(10);
potencia10=mean(d10.^2)
end
if x==11
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
% antena7
d7=D(7)+n+E(7);
potencia7=mean(d7.^2)
% antena8
d8=D(8)+n+E(8);
potencia8=mean(d8.^2)
% antena9
d9=D(9)+n+E(9);
potencia9=mean(d9.^2)
% antena10
d10=D(10)+n+E(10);
potencia10=mean(d10.^2)
% antena11
d11=D(11)+n+E(11);
potencia11=mean(d11.^2)
end
if x==12
```

```
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
% antena7
d7=D(7)+n+E(7);
potencia7=mean(d7.^2)
% antena8
d8=D(8)+n+E(8);
potencia8=mean(d8.^2)
% antena9
d9=D(9)+n+E(9);
potencia9=mean(d9.^2)
% antena10
d10=D(10)+n+E(10);
potencia10=mean(d10.^2)
% antena11
d11=D(11)+n+E(11);
potencia11=mean(d11.^2)
% antena12
d12=D(12)+n+E(12);
potencia12=mean(d12.^2)
end
if x==13
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
```

```
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
% antena7
d7=D(7)+n+E(7);
potencia7=mean(d7.^2)
% antena8
d8=D(8)+n+E(8);
potencia8=mean(d8.^2)
% antena9
d9=D(9)+n+E(9);
potencia9=mean(d9.^2)
% antena10
d10=D(10)+n+E(10);
potencia10=mean(d10.^2)
% antena11
d11=D(11)+n+E(11);
potencia11=mean(d11.^2)
% antena12
d12=D(12)+n+E(12);
potencia12=mean(d12.^2)
% antena13
d13=D(13)+n+E(13);
potencia13=mean(d13.^2)
end
if x==14
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
% antena7
```

```
d7=D(7)+n+E(7);
potencia7=mean(d7.^2)
% antena8
d8=D(8)+n+E(8);
potencia8=mean(d8.^2)
% antena9
d9=D(9)+n+E(9);
potencia9=mean(d9.^2)
% antena10
d10=d(10)+n+E(10);
potencia10=mean(d10.^2)
% antena11
d11=D(11)+n+E(11);
potencia11=mean(d11.^2)
% antena12
d12=D(12)+n+E(12);
potencia12=mean(d12.^2)
% antena13
d13=D(13)+n+E(13);
potencia13=mean(d13.^2)
% antena14
d14=D(14)+n+E(14);
potencia14=mean(d14.^2)
end
if x==15
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
% antena7
d7=D(7)+n+E(7);
potencia7=mean(d7.^2)
% antena8
d8=D(8)+n+E(8);
potencia8=mean(d8.^2)
```

```
% antena9
d9=D(9)+n+E(9);
potencia9=mean(d9.^2)
% antena10
d10=D(10)+n+E(10);
potencia10=mean(d10.^2)
% antena11
d11=D(11)+n+E(11);
potencia11=mean(d11.^2)
% antena12
d12=D(12)+n+E(12);
potencia12=mean(d12.^2)
% antena13
d13=D(13)+n+E(13);
potencia13=mean(d13.^2)
% antena14
d14=D(14)+n+E(14);
potencia14=mean(d14.^2)
% antena15
d15=D(15)+n+E(15);
potencia15=mean(d15.^2)
end
% calculo de la cota de la contante de ajuste
cota=1/((orden+1)*potencia);
%evalua 70 puntos por corrida ( N - orden 70 = 80 - 10 )
N=80;   % N=80
%begin of the algorithm
% el filtro del algoritmo rls se calcula con los coeficientes a,b
% con el valor de landa, con el valor de delta y con el numero N de
% seguridad (chequea los resultados hasta N
% en caso de que los valores de los pesos no convergan
% hay que cambiar estos valores, hasta encontrar valores que
% permitan la convergencia con poco error
% en caso de algunas funciones puede ser bastante pesado encontrar los
% valores correctos
%forgetting factor valores recomendados por la biblioteca matlab
lamda = 1 ;% lamda= 0.9995
%initial P matrix
delta=10000000000 % delta = 1e10 ;
P = delta * eye (orden);
% w = zeros ( orden  , 1 )
if x==1
    w=[0.015]
    w=w'
end
if x==2
    w=[ 0.015 0.010]
    w=w'
end
```

```
if x==3
    w=[0.015 0.010 0.020 ]
    w=w'
end
if x==4
    w=[0.015 0.010 0.020 0.030]
    w=w'
end
if x==5
    w=[0.015 0.010 0.020 0.030 -0.020]
    w=w'
end
if x==6
    w=[0.015 0.010 0.020 0.030 -0.020 0.014]
    w=w'
end
if x==7
    w=[0.015 0.010 0.020 0.030 -0.020 0.014 0.012]
    w=w'
end
if x==8
    w=[0.015 0.010 0.020 0.030 -0.020 0.014 0.012 0.011]
    w=w'
end
if x==9
    w=[0.015 0.010 0.020 0.030 -0.020 0.014 0.012 0.011 0.005]
    w=w'
end
if x==10
    w=[0.015 0.010 0.020 0.030 -0.020 0.014 0.012 0.011 0.005 -0.005]
    w=w'
end
if x==11
    w=[0.015 0.010 0.020 0.30 -0.20 0.14 0.12 0.11 0.05 -0.05 -0.09]
    w=w'
end
if x==12
    w=[0.15 0.10 0.20 0.30 -0.20 0.14 0.12 0.11 0.05 -0.05 -0.09 0.05]
    w=w'
end
if x==13
    w=[0.15 0.10 0.20 0.30 -0.20 0.14 0.12 0.11 0.05 -0.05 -0.09 0.05
0.08]
    w=w'
end
if x==14
    w=[0.15 0.10 0.20 0.30 -0.20 0.14 0.12 0.11 0.05 -0.05 -0.09 0.05
0.08 0.06]
    w=w'
```

```
end
if x==15
    w=[0.15 0.10 0.20 0.30 -0.20 0.14 0.12 0.11 0.05 -0.05 -0.09 0.05
0.08 0.06 0.02]
    w=w'
end
% T10=[D(1) D(2) D(3) D(4) D(5)]
Rp=1;
for n = orden : N
    u = entrada(n:-1:n-orden+1);
    phi = u' * P ;
    k = phi'/(lamda + phi * u );
    y(n)= w'*u ;
    y(n);
    T(n)=[abs(y(n))].^2;
    T2(n)=Rp-T(n); %Rp valor de la constante cma, si no converge,
    %pruebe con otros valores de Rp, valores de entrada, señal de
    % entrada
    T3=T2(n)*u;
    e=T3*y(n); % e(n) = senal_referencia(n) - y(n); error para RLS
    % e(n)=senal_referencia(n)-y(n);
    T4=e';
    T5=T4*u;
    mu=5*10^-7;
    %if n < 20
    %    mu=cota/2
    %else
    %    mu=cota/4
    %end
    w=(w)-(mu*T5)  % w = (w) + k * e(n);  peso para RLS
    % w=w/10;
    P = ( P - k * phi ) / lamda;
    % ajuste del ploteo, pesos
    Recordedw(1:orden,n)=w;
end
%check of results
for n =  N+1 : totallength
    u = entrada(n:-1:n-orden+1) ;
    y(n) = w' * u ;
    T(n)=[abs(y(n))].^2;
    T2(n)=1-T(n);
    e=[T2(n)*y(n)]*u;    %e(n) = senal_referencia(n) - y(n);rls
    % e(n)=e(n)*u;
end
hold on
plot(senal_referencia)
plot(y,'r');
title('salida del sistema ') ;
xlabel('numero de muestras')
```

```
ylabel('valor estimado a la salida')
figure
semilogy((abs(e))) ;
title('curva de error') ;
xlabel('numero de muestras');
ylabel('valor del error');
figure
plot(Recordedw(1:orden,orden:N)');
title('convergencia de los pesos estimados') ;
xlabel('numero de muestras');
ylabel('valor de los pesos');
axis([1 N-orden min(min(Recordedw(1:orden,orden:N)'))
max(max(Recordedw(1:orden,orden:N)')) ]);
hold off
figure
plot(h, 'k+')
hold on
plot(w, 'r*')
legend('pesos actuales','pesos estimados');
title('Comparacion de los pesos actuales con los pesos estimados') ;
```

```
function[caratula]=problema3_cma;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
% Algoritmo CMA
% Autores: Juan Alvarez y Maribel Chuez
%
%Escuela Superior Politecnica del Litoral
%Facultad de Electricidad y Computación
%Guayaquil, Ecuador
% email :juan_alvarez8@latinmail.com
% email : mchuez@ceibo.espol.edu.ec
% Webpage : proximamente
% Date    : Miercoles 22 de Julio del 2009
% Version : 1.0.1
% Referencias : S. Haykin, Adaptive Filter Theory. 3rd edition, Upper
Saddle River, NJ: Prentice-Hall, 1996.
%
%clear all % esta funcion borra los valores anteriores que esten
memoria %
%close all % esta funcion borra cualquier ventana anterior activa %
%hold off   % esta funcion encera cualquier grafico que estuviera
activo %
 fprintf('*****************************************************.\n')
fprintf('*    ESCUELA SUPERIOR POLITECNICA DEL LITORAL        *.\n')
fprintf('*                                                    *.\n')
fprintf('* PROBLEMA3_CMA.- Determinar los pesos, el numero de *.\n')
fprintf('* iteraciones y la potencia de un grupo de antenas   *.\n')
fprintf('* inteligentes utilizando el algoritmo CMA-RLS       *.\n')
fprintf('* las antenas estan distribuidas en forma de celdas  *.\n')
fprintf('* para lo cual requeriremos de los siguientes datos: *.\n')
fprintf('*                                                    *.\n')
fprintf('*  • la señal aplicada a la entrada del filtro,      *.\n')
fprintf('*   ej  Xn=sin((2*pi*k)/N)                           *.\n')
fprintf('*  • el numero de antenas  x                         *.\n')
fprintf('*  • la señal de referencia, d(k)=cos((2*pi*k)/N);   *.\n')
fprintf('*  • el ruido de las diferentes antenas, (hay que    *.\n')
fprintf('*  tener presente que el algoritmo CMA en su         *.\n')
fprintf('*  estructura internamente produce un ruido que puede*.\n')
fprintf('*  ser de 2, 3 y hasta 10 %  aparte de este ruido    *.\n')
fprintf('*  generado por el mismo programa CMA se puede       *.\n')
fprintf('*  ingresar ruidos adicionales en el programa a la   *.\n')
fprintf('*  entrada o a la salida del programa.  n(f)         *.\n')
fprintf('*  • el orden del filtro FIR cuyos coeficientes se   *.\n')
fprintf('*  van a ajustar, orden                              *.\n')
fprintf('*  • el número de iteraciones en el aprendizaje, N   *.\n')
fprintf('*  (este número se hará  hasta que cumpla el error   *.\n')
fprintf('*  • la constante de ajuste   u                      *.\n')
fprintf('*  • el vector de pesos inicial. Los voltajes de las *.\n')
fprintf('*  antenas V(f)                                      *.\n')
```

```
fprintf('*  • el error deseado puede ser ingresado por teclado*.\n')
fprintf('*    error                                           *.\n')
fprintf('*                                                    *.\n')
fprintf('*  A su salida devolverá:                            *.\n')
fprintf('*  • la salida en cada iteración durante el proceso  *.\n')
fprintf('*  de ajuste, y(t)                                   *.\n')
fprintf('*  • la curva de aprendizaje (grafico de la curva)   *.\n')
fprintf('*  • los vectores de pesos obtenidos durante el pro- *.\n')
fprintf('*  ceso. W(t)                                        *.\n')
fprintf('*  • la Potencia de la antena P(k)                   *.\n')
fprintf('*  • el numero de iteraciones en que converge la     *.\n')
fprintf('*    curva y se cumple el valor del error            *.\n')
fprintf('*                                                    *.\n')
fprintf('* alumno: Juan Francisco Alvarez Alvarado            *.\n')
fprintf('* alumno: Maribel del Rosario Chuez Gonzales         *.\n')
fprintf('*                                                    *.\n')
fprintf('* Profesor :Ing. Pedro Vargas                        *.\n')
fprintf('*                                                    *.\n')
fprintf('******************************************************.\n')
t=input('teclee enter para proseguir');
 clc
% ingresamos el orden del filtro, cuyo coeficiente se va ha ajustar
fprintf('consideraremos un orden = 2  para la segunda etapa de un ')
fprintf('conformador de haz, esto es para que la ecuación de      ')
fprintf('pesos que se utiliza en el programa sea la de un         ')
fprintf('algoritmo CMA, este valor será constante en la ecuación  ')
fprintf('el valor del orden del filtro para determinar los valo   ')
fprintf('res de cota y de la potencia si variaran normalmente como')
fprintf('si trabajaran en un algoritmo RLS                        ')
fprintf('                                                         ')
orden=input(' ingrese el orden del filtro, cuyo coeficiente se va ha
ajustar....')
% ingresamos el numero de antenas del sistema
disp(' restriccion: ingrese un numero de antenas igual al del orden del
filtro ingresado)');
x=input(' ingrese el numero de antenas entre
1,2,3,4,5,6,7,8,9,10,11,12,13,14 o 15....');
if x==1
    D(1)=input('ingrese el voltaje1 (en V)..... voltaje1=');
end

if x==2
    D(1)=input('ingrese el voltaje1 (en V)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
end

if x==3
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
```

```
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
end

if x==4
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
end
if x==5
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
end
if x==6
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
end
if x==7
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
    D(7)=input(' ingrese la voltaje7 (en v)..... voltaje7=');
end
if x==8
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
    D(7)=input(' ingrese la voltaje7 (en v)..... voltaje7=');
    D(8)=input(' ingrese la voltaje8 (en v)..... voltaje8=');
end
if x==9
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
```

```
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
    D(7)=input(' ingrese la voltaje7 (en v)..... voltaje7=');
    D(8)=input(' ingrese la voltaje8 (en v)..... voltaje8=');
    D(9)=input(' ingrese la voltaje9 (en v)..... voltaje9=');
end
if x==10
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
    D(7)=input(' ingrese la voltaje7 (en v)..... voltaje7=');
    D(8)=input(' ingrese la voltaje8 (en v)..... voltaje8=');
    D(9)=input(' ingrese la voltaje9 (en v)..... voltaje9=');
    D(10)=input(' ingrese la voltaje10 (en v)..... voltaje10=');
end
if x==11
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
    D(7)=input(' ingrese la voltaje7 (en v)..... voltaje7=');
    D(8)=input(' ingrese la voltaje8 (en v)..... voltaje8=');
    D(9)=input(' ingrese la voltaje9 (en v)..... voltaje9=');
    D(10)=input(' ingrese la voltaje10 (en v)..... voltaje10=');
    D(11)=input(' ingrese la voltaje11 (en v)..... voltaje11=');
end
if x==12
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
    D(7)=input(' ingrese la voltaje7 (en v)..... voltaje7=');
    D(8)=input(' ingrese la voltaje8 (en v)..... voltaje8=');
    D(9)=input(' ingrese la voltaje9 (en v)..... voltaje9=');
    D(10)=input(' ingrese la voltaje10 (en v)..... voltaje10=');
    D(11)=input(' ingrese la voltaje11 (en v)..... voltaje11=');
    D(12)=input(' ingrese la voltaje12 (en v)..... voltaje12=');
end
if x==13
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
```

```
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
    D(7)=input(' ingrese la voltaje7 (en v)..... voltaje7=');
    D(8)=input(' ingrese la voltaje8 (en v)..... voltaje8=');
    D(9)=input(' ingrese la voltaje9 (en v)..... voltaje9=');
    D(10)=input(' ingrese la voltaje10 (en v)..... voltaje10=');
    D(11)=input(' ingrese la voltaje11 (en v)..... voltaje11=');
    D(12)=input(' ingrese la voltaje12 (en v)..... voltaje12=');
    D(13)=input(' ingrese la voltaje13 (en v)..... voltaje13=');
end
if x==14
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
    D(7)=input(' ingrese la voltaje7 (en v)..... voltaje7=');
    D(8)=input(' ingrese la voltaje8 (en v)..... voltaje8=');
    D(9)=input(' ingrese la voltaje9 (en v)..... voltaje9=');
    D(10)=input(' ingrese la voltaje10 (en v)..... voltaje10=');
    D(11)=input(' ingrese la voltaje11 (en v)..... voltaje11=');
    D(12)=input(' ingrese la voltaje12 (en v)..... voltaje12=');
    D(13)=input(' ingrese la voltaje13 (en v)..... voltaje13=');
    D(14)=input(' ingrese la voltaje14 (en v)..... voltaje14=');
end
if x==15
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
    D(6)=input(' ingrese la voltaje6 (en v)..... voltaje6=');
    D(7)=input(' ingrese la voltaje7 (en v)..... voltaje7=');
    D(8)=input(' ingrese la voltaje8 (en v)..... voltaje8=');
    D(9)=input(' ingrese la voltaje9 (en v)..... voltaje9=');
    D(10)=input(' ingrese la voltaje10 (en v)..... voltaje10=');
    D(11)=input(' ingrese la voltaje11 (en v)..... voltaje11=');
    D(12)=input(' ingrese la voltaje12 (en v)..... voltaje12=');
    D(13)=input(' ingrese la voltaje13 (en v)..... voltaje13=');
    D(14)=input(' ingrese la voltaje14 (en v)..... voltaje14=');
    D(15)=input(' ingrese la voltaje15 (en v)..... voltaje15=');
end

% ingresamos el ruido que presenta cada antena del sistema
% Si existe ruido adicional al ocasionado por el sistema
if x==1
    E(1)=input(' ingrese el ruido en la antena1.....ruido1=');
    ruido=E(1);
```

```
end

if x==2
    E(1)=input('ingrese el ruido en la antena1.....ruido1==');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    ruido=(E(1)+E(2))/2;
end

if x==3
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    ruido=(E(1)+E(2)+E(3))/3;
end

if x==4
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    ruido=(E(1)+E(2)+E(3)+E(4))/4;
end
if x==5
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    ruido=(E(1)+E(2)+E(3)+E(4)+E(5))/5;
end
if x==6
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6))/6;
end
if x==7
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    E(7)=input('ingrese el ruido en la antena7.....ruido7=');
    ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6)+E(7))/7;
end
```

```
if x==8
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    E(7)=input('ingrese el ruido en la antena7.....ruido7=');
    E(8)=input('ingrese el ruido en la antena8.....ruido8=');
    ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6)+E(7)+E(8))/8;
end
if x==9
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    E(7)=input('ingrese el ruido en la antena7.....ruido7=');
    E(8)=input('ingrese el ruido en la antena8.....ruido8=');
    E(9)=input('ingrese el ruido en la antena9.....ruido9=');
    ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6)+E(7)+E(8)+E(9))/9;
end
if x==10
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    E(7)=input('ingrese el ruido en la antena7.....ruido7=');
    E(8)=input('ingrese el ruido en la antena8.....ruido8=');
    E(9)=input('ingrese el ruido en la antena9.....ruido9=');
    E(10)=input('ingrese el ruido en la antena10.....ruido10=');
    ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6)+E(7)+E(8)+E(9)+E(10))/10;
end
if x==11
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    E(7)=input('ingrese el ruido en la antena7.....ruido7=');
    E(8)=input('ingrese el ruido en la antena8.....ruido8=');
    E(9)=input('ingrese el ruido en la antena9.....ruido9=');
    E(10)=input('ingrese el ruido en la antena10.....ruido10=');
    E(11)=input('ingrese el ruido en la antena11.....ruido11=');
```

```
ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6)+E(7)+E(8)+E(9)+E(10)+E(11))/11;
end
if x==12
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    E(7)=input('ingrese el ruido en la antena7.....ruido7=');
    E(8)=input('ingrese el ruido en la antena8.....ruido8=');
    E(9)=input('ingrese el ruido en la antena9.....ruido9=');
    E(10)=input('ingrese el ruido en la antena10.....ruido10=');
    E(11)=input('ingrese el ruido en la antena11.....ruido11=');
    E(12)=input('ingrese el ruido en la antena12.....ruido12=');

ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6)+E(7)+E(8)+E(9)+E(10)+E(11)+E(12))/
12;
end
if x==13
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    E(7)=input('ingrese el ruido en la antena7.....ruido7=');
    E(8)=input('ingrese el ruido en la antena8.....ruido8=');
    E(9)=input('ingrese el ruido en la antena9.....ruido9=');
    E(10)=input('ingrese el ruido en la antena10.....ruido10=');
    E(11)=input('ingrese el ruido en la antena11.....ruido11=');
    E(12)=input('ingrese el ruido en la antena12.....ruido12=');
    E(13)=input('ingrese el ruido en la antena13.....ruido13=');

ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6)+E(7)+E(8)+E(9)+E(10)+E(11)+E(12)+E
(13))/13;
end
if x==14
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    E(7)=input('ingrese el ruido en la antena7.....ruido7=');
    E(8)=input('ingrese el ruido en la antena8.....ruido8=');
    E(9)=input('ingrese el ruido en la antena9.....ruido9=');
    E(10)=input('ingrese el ruido en la antena10.....ruido10=');
```

```
    E(11)=input('ingrese el ruido en la antena11.....ruido11=');
    E(12)=input('ingrese el ruido en la antena12.....ruido12=');
    E(13)=input('ingrese el ruido en la antena13.....ruido13=');
    E(14)=input('ingrese el ruido en la antena14.....ruido14=');

ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6)+E(7)+E(8)+E(9)+E(10)+E(11)+E(12)+E
(13)+E(14))/14;
end
if x==15
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    E(6)=input('ingrese el ruido en la antena6.....ruido6=');
    E(7)=input('ingrese el ruido en la antena7.....ruido7=');
    E(8)=input('ingrese el ruido en la antena8.....ruido8=');
    E(9)=input('ingrese el ruido en la antena9.....ruido9=');
    E(10)=input('ingrese el ruido en la antena10.....ruido10=');
    E(11)=input('ingrese el ruido en la antena11.....ruido11=');
    E(12)=input('ingrese el ruido en la antena12.....ruido12=');
    E(13)=input('ingrese el ruido en la antena13.....ruido13=');
    E(14)=input('ingrese el ruido en la antena14.....ruido14=');
    E(15)=input('ingrese el ruido en la antena15.....ruido15=');

ruido=(E(1)+E(2)+E(3)+E(4)+E(5)+E(6)+E(7)+E(8)+E(9)+E(10)+E(11)+E(12)+E
(13)+E(14)+E(15))/15;
end

opcion_ruido=2;
disp('si desea que el ruido adicional vaya a la señal a la entrada
digite 1');
disp(' de lo contrario se considerara que el ruido adicional se
agregara a la señal de referencia');
opcion_ruido=input('opcion_ruido=');
error=input('ingrese el error del sistema=');
% numero de iteraciones del proceso hasta que se cumpla el error
opcion_senal=2;
disp('si desea que la señal de entrada se use como señal de referencia
digite 1');
disp('de lo contrario se considerara otra señal diferente como señal de
referencia');
opcion_senal=input('opcion_señal=');
N=400;
j=0;
% valores recomendado por la central matlab para los pesos en una
funcion
%
% h=
```

```
% vamos a considerar que todas las antenas tienen la misma señal a la
% entrada del filtro por ejemplo  entrada=sin((2*pi*k)/N);
%
if x==1
    h=[D(1)]
    p=[D(1)]
end
if x==2
    h=[D(1);D(2)]
    p=[(D(1)+D(2))/2]+[0.8*(D(1)+D(2))/2];
end
if x==3
    h=[D(1);D(2);D(3)]
    p=[(D(1)+D(2)+D(3))/3]+[0.8*(D(1)+D(2)+D(3))/3];
end
if x==4
    h=[D(1);D(2);D(3);D(4)]
    p=[(D(1)+D(2)+D(3)+D(4))/4]+[0.8*(D(1)+D(2)+D(3)+D(4))/4];
end
if x==5
    h=[D(1);D(2);D(3);D(4);D(5)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5))/5]+[0.8*(D(1)+D(2)+D(3)+D(4)+D(5))/5];
end
if x==6
    h=[D(1);D(2);D(3);D(4);D(5);D(6)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6))/6]+[0.8*(D(1)+D(2)+D(3)+D(4)+D(5)+D(
6))/6];
end
if x==7
    h=[D(1);D(2);D(3);D(4);D(5);D(6);D(7)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7))/7]+[0.8*(D(1)+D(2)+D(3)+D(4)+D(
5)+D(6)+D(7))/7];
end
if x==8
    h=[D(1);D(2);D(3);D(4);D(5);D(6);D(7);D(8)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8))/8]+[0.8*(D(1)+D(2)+D(3)+D(
4)+D(5)+D(6)+D(7)+D(8))/8];
end
if x==9
    h=[D(1);D(2);D(3);D(4);D(5);D(6);D(7);D(8);D(9)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9))/9]+[0.8*(D(1)+D(2)+D(
3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9))/9];
end
if x==10
```

```
    h=[D(1);D(2);D(3);D(4);D(5);D(6);D(7);D(8);D(9);D(10)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10))/10]+[0.8*(D(1)+
D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10))/10];
end
if x==11
    h=[D(1);D(2);D(3);D(4);D(5);D(6);D(7);D(8);D(9);D(10);D(11)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10)+D(11))/11]+[0.8*
(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10)+D(11))/11];
end
if x==12
    h=[D(1);D(2);D(3);D(4);D(5);D(6);D(7);D(8);D(9);D(10);D(11);D(12)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10)+D(11)+D(12))/12]
+[0.8*(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10)+D(11)+D(12))/
12];
end
if x==13

h=[D(1);D(2);D(3);D(4);D(5);D(6);D(7);D(8);D(9);D(10);D(11);D(12);D(13)
]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10)+D(11)+D(12)+D(13
))/13]+[0.8*(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10)+D(11)+D
(12)+D(13))/13];
end
if x==14

h=[D(1);D(2);D(3);D(4);D(5);D(6);D(7);D(8);D(9);D(10);D(11);D(12);D(13)
;D(14)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10)+D(11)+D(12)+D(13
)+D(14))/14]+[0.8*(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10)+D
(11)+D(12)+D(13)+D(14))/14];
end
if x==15

h=[D(1);D(2);D(3);D(4);D(5);D(6);D(7);D(8);D(9);D(10);D(11);D(12);D(13)
;D(14);D(15)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D(10)+D(11)+D(12)+D(13
)+D(14)+D(15))/15]+[0.8*(D(1)+D(2)+D(3)+D(4)+D(5)+D(6)+D(7)+D(8)+D(9)+D
(10)+D(11)+D(12)+D(13)+D(14)+D(15))/15];
end
M=30;
for k=1:400
    entrada(k)=sin((2*pi*k)/M);
    %senal de entrada
```

```
end
for k=1:400
    senal_referencia(k)=cos((2*pi*k)/M);
    % en algunos libros la señal de referencia se la llama señal
deseada
end
entrada';
entrada=entrada';
if opcion_ruido==1
    entrada=entrada+ruido; % aqui agregamos el ruido adicional a la
entrada
end
n=sqrt(0.05)*randn(400,1); % ruido a la entrada del filtro, genera 400
valores
% aleatorios de muestra
[b,a] = butter(2,0.25); % esta funcion calcula los valores de a y b
Gz = tf(b,a,-1); % la funcion de transferencia tf funciona en matlab 7
% esta funcion permite resolver la transformada z.

%Si no se tiene acceso a la caja de herramientas del matlab se puede
cargar valores de
%la central matlab usando el sample data
%load IIRsampledata;
%entrada= IIRsampledata(1:2000);
%d= IIRsampledata(1:2000);

% se puede usar ldiv para hallar las aproximaciones de pesos para un
filtro IIR (filtro de respuesta infinita)
% tambien se puede evaluar  y=h*u
%ldiv es una funcion que nos permite hallar la inversa de la
transformada Z (Matlab central file exchange)
%para hallar los valores de orden de los pesos se puede utilizar la
%siguiente formula
%use h=ldiv(b,a,sysorder)'; ==> here we use sysorder == 10

h=h(1:orden);
y = lsim(Gz,entrada);% simula respuestas arbitrarias de entrada,
relacion la señal Gz
% con la entrada
% para agregar el ruido, ruido adicional que genera el algoritmo rls
n = n * std(y)/(10*std(n));
if opcion_senal==1
senal_referencia = y + n;
end
if opcion_senal>1
    senal_referencia';
    senal_referencia=senal_referencia';
    senal_referencia=senal_referencia+n;
    %
```

```
end
if opcion_ruido>1
    senal_referencia=senal_referencia+ruido; % agrega el ruido
adicional
    % a la salida del filtro
end
totallength=size(senal_referencia,1);% nos da el numero de valores de
la funcion
% de entrada
% conociendo los voltajes de las antenas podemos determinar la potencia
% en ellas
% potencia de la señal a la salida (potencia del sistema
potencia=mean(senal_referencia.^2)
if x==1
% tambien calcularemos la potencia para cada una de las antenas
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
end
if x==2
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
end
if x==3
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2);
% antena2
d2=D(2)+n+E(2);
potencia1=mean(d2.^2);
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2);
end
if x==4
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2);
% antena2
d2=D(2)+n+E(2);
potencia1=mean(d2.^2);
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2);
% antena4
```

```
d4=D(4)+n+E(4);
potencia4=mean(d4.^2);
end
if x==5
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
end
if x==6
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
end
if x==7
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
```

```
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
% antena7
d7=D(7)+n+E(7);
potencia7=mean(d7.^2)
end
if x==8
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
% antena7
d7=D(7)+n+E(7);
potencia7=mean(d7.^2)
% antena8
d8=D(8)+n+E(8);
potencia8=mean(d8.^2)
end
if x==9
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
```

```
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
% antena7
d7=D(7)+n+E(7);
potencia7=mean(d7.^2)
% antena8
d8=D(8)+n+E(8);
potencia8=mean(d8.^2)
% antena9
d9=D(9)+n+E(9);
potencia9=mean(d9.^2)
end
if x==10
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
% antena7
d7=D(7)+n+E(7);
potencia7=mean(d7.^2)
% antena8
d8=D(8)+n+E(8);
potencia8=mean(d8.^2)
% antena9
d9=D(9)+n+E(9);
potencia9=mean(d9.^2)
% antena10
```

```
d10=D(10)+n+E(10);
potencia10=mean(d10.^2)
end
if x==11
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=d(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
% antena7
d7=D(7)+n+E(7);
potencia7=mean(d7.^2)
% antena8
d8=D(8)+n+E(8);
potencia8=mean(d8.^2)
% antena9
d9=D(9)+n+E(9);
potencia9=mean(d9.^2)
% antena10
d10=D(10)+n+E(10);
potencia10=mean(d10.^2)
% antena11
d11=D(11)+n+E(11);
potencia11=mean(d11.^2)
end
if x==12
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
```

```
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
% antena7
d7=D(7)+n+E(7);
potencia7=mean(d7.^2)
% antena8
d8=D(8)+n+E(8);
potencia8=mean(d8.^2)
% antena9
d9=D(9)+n+E(9);
potencia9=mean(d9.^2)
% antena10
d10=D(10)+n+E(10);
potencia10=mean(d10.^2)
% antena11
d11=d(11)+n+E(11);
potencia11=mean(d11.^2)
% antena12
d12=D(12)+n+E(12);
potencia12=mean(d12.^2)
end
if x==13
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
% antena7
d7=D(7)+n+E(7);
potencia7=mean(d7.^2)
```

```
% antena8
d8=D(8)+n+E(8);
potencia8=mean(d8.^2)
% antena9
d9=D(9)+n+E(9);
potencia9=mean(d9.^2)
% antena10
d10=D(10)+n+E(10);
potencia10=mean(d10.^2)
% antena11
d11=D(11)+n+E(11);
potencia11=mean(d11.^2)
% antena12
d12=D(12)+n+E(12);
potencia12=mean(d12.^2)
% antena13
d13=D(13)+n+E(13);
potencia13=mean(d13.^2)
end
if x==14
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
% antena7
d7=D(7)+n+E(7);
potencia7=mean(d7.^2)
% antena8
d8=D(8)+n+E(8);
potencia8=mean(d8.^2)
% antena9
d9=D(9)+n+E(9);
potencia9=mean(d9.^2)
% antena10
d10=D(10)+n+E(10);
```

```
potencia10=mean(d10.^2)
% antena11
d11=D(11)+n+E(11);
potencia11=mean(d11.^2)
% antena12
d12=D(12)+n+E(12);
potencia12=mean(d12.^2)
% antena13
d13=D(13)+n+E(13);
potencia13=mean(d13.^2)
% antena14
d14=D(14)+n+E(14);
potencia14=mean(d14.^2)
end
if x==15
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
% antena6
d6=D(6)+n+E(6);
potencia6=mean(d6.^2)
% antena7
d7=D(7)+n+E(7);
potencia7=mean(d7.^2)
% antena8
d8=D(8)+n+E(8);
potencia8=mean(d8.^2)
% antena9
d9=D(9)+n+E(9);
potencia9=mean(d9.^2)
% antena10
d10=D(10)+n+E(10);
potencia10=mean(d10.^2)
% antena11
d11=D(11)+n+E(11);
potencia11=mean(d11.^2)
% antena12
```

```
d12=D(12)+n+E(12);
potencia12=mean(d12.^2)
% antena13
d13=D(13)+n+E(13);
potencia13=mean(d13.^2)
% antena14
d14=D(14)+n+E(14);
potencia14=mean(d14.^2)
% antena15
d15=D(15)+n+E(15);
potencia15=mean(d15.^2)
end
% calculo de la cota de la contante de ajuste
cota=1/((orden+1)*potencia);
%evalua 70 puntos por corrida ( N - orden 70 = 80 - 10 )
N=80;   % N=80
%begin of the algorithm
%forgetting factor valores recomendados por la biblioteca matlab
lamda = 0.999994115 ;% lamda= 0.9995
%initial P matrix
delta=1000000000000000 % delta = 1e10 ;
P = delta * eye (orden);
% w = zeros ( orden  , 1 )
if x==1
    w=[0.15]
    w=w'
end
if x==2
    w=[ 0.15 0.10]
    w=w'
end
if x==3
    w=[0.15 0.10 0.20 ]
    w=w'
end
if x==4
    w=[0.15 0.10 0.20 0.30]
    w=w'
end
if x==5
    w=[0.15 0.10 0.20 0.30 -0.20]
    w=w'
end
if x==6
    w=[0.15 0.10 0.20 0.30 -0.20 0.14]
    w=w'
end
if x==7
    w=[0.15 0.10 0.20 0.30 -0.20 0.14 0.12]
```

```
    w=w'
end
if x==8
    w=[0.15 0.10 0.20 0.30 -0.20 0.14 0.12 0.11]
    w=w'
end
if x==9
    w=[0.15 0.10 0.20 0.30 -0.20 0.14 0.12 0.11 0.05]
    w=w'
end
if x==10
    w=[0.15 0.10 0.20 0.30 -0.20 0.14 0.12 0.11 0.05 -0.05]
    w=w'
end
if x==11
    w=[0.15 0.10 0.20 0.30 -0.20 0.14 0.12 0.11 0.05 -0.05 -0.09]
    w=w'
end
if x==12
    w=[0.15 0.10 0.20 0.30 -0.20 0.14 0.12 0.11 0.05 -0.05 -0.09 0.05]
    w=w'
end
if x==13
    w=[0.15 0.10 0.20 0.30 -0.20 0.14 0.12 0.11 0.05 -0.05 -0.09 0.05
0.08]
    w=w'
end
if x==14
    w=[0.15 0.10 0.20 0.30 -0.20 0.14 0.12 0.11 0.05 -0.05 -0.09 0.05
0.08 0.06]
    w=w'
end
if x==15
    w=[0.15 0.10 0.20 0.30 -0.20 0.14 0.12 0.11 0.05 -0.05 -0.09 0.05
0.08 0.06 0.02]
    w=w'
end
% T10=[D(1) D(2) D(3) D(4) D(5)]
Rp =1;
for n = orden : N
    u = entrada(n:-1:n-orden+1)
    phi = u' * P ;
    k = phi'/(lamda + phi * u );
    y(n)= w'*u ;
    y(n)
    T(n)=[abs(y(n))].^2
    T2(n)=Rp-T(n); % Rp valor de la constante cma, si no converge,
                   % pruebe con otros valores de Rp
    T3=T2(n)*u
```

```
    e=T3*y(n) % e(n) = senal_referencia(n) - y(n); error para RLS
    % e(n)=senal_referencia(n)-y(n);
    T4=e'
    T5=T4*u
    mu=5*10^-7
    w=(w)-(mu*T5)  % w = (w) + k * e(n);  peso para RLS
    % si no converg utilice e* conjugada;
    P = ( P - k * phi ) / lamda;
    % ajuste del ploteo, pesos
    Recordedw(1:orden,n)=w
end
%check of results
for n =  N+1 : totallength
    u = entrada(n:-1:n-orden+1) ;
    y(n) = w' * u ;
    T(n)=[abs(y(n))].^2;
    T2(n)=1-T(n);
    e(n)=T2(n)*y(n);    %e(n) = senal_referencia(n) - y(n);rls
    % e(n)=e(n)*u;
end

hold on
plot(senal_referencia)
plot(y,'r');
title('salida del sistema ') ;
xlabel('numero de muestras')
ylabel('valor estimado a la salida')
figure
semilogy((abs(e))) ;
title('curva de error') ;
xlabel('numero de muestras');
ylabel('valor del error');
figure
plot(Recordedw(1:orden,orden:N)');
title('convergencia de los pesos estimados') ;
xlabel('numero de muestras');
ylabel('valor de los pesos');
axis([1 N-orden min(min(Recordedw(1:orden,orden:N)'))
max(max(Recordedw(1:orden,orden:N)')) ]);
hold off
figure
plot(h, 'k+')
hold on
plot(w, 'r*')
legend('pesos actuales','pesos estimados');
title('Comparacion de los pesos actuales con los pesos estimados') ;
```

Problema1_dmi

```
function[caratula]=problema1_dmi;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
% Algoritmo DMI
% Autores: Juan Alvarez , Maribel Chuez
%
%Escuela Superior Politecnica del Litoral
%Facultad de Electricidad y Computación
%Guayaquil, Ecuador
% email :juan_alvarez8@latinmail.com
% email : mchuez@ceibo.espol.edu.ec
% Webpage : proximamente
% Date    : Domingo 26 de Julio del 2009
% Version : 1.0.1
% Referencias : S. Haykin, Adaptive Filter Theory. 3rd edition, Upper
Saddle River, NJ: Prentice-Hall, 1996.
%
%clear all % esta funcion borra los valores anteriores que esten
memoria %
%close all % esta funcion borra cualquier ventana anterior activa %
%hold off   % esta funcion encera cualquier grafico que estuviera
activo %
 fprintf('*****************************************************.\n')
fprintf('*    ESCUELA SUPERIOR POLITECNICA DEL LITORAL        *.\n')
fprintf('*                                                    *.\n')
fprintf('* PROBLEMA1_dmi.- Determinar los pesos, el numero de *.\n')
fprintf('* iteraciones y la potencia de un grupo de antenas   *.\n')
fprintf('* inteligentes utilizando el algoritmo DMI (algoritmo*.\n')
fprintf('* adaptativo de inversión de matriz directa, las     *.\n')
fprintf('* antenas están distribuidas en forma de celdas      *.\n')
fprintf('* para lo cual requeriremos de los siguientes datos: *.\n')
fprintf('*                                                    *.\n')
fprintf('*  • la señal aplicada a la entrada del filtro,      *.\n')
fprintf('*   ej  Xn=sin((2*pi*k)/N)                           *.\n')
fprintf('*  • el numero de antenas  x                         *.\n')
fprintf('*  • la señal de referencia, d(k)=cos((2*pi*k)/N);   *.\n')
fprintf('*  • el ruido de las diferentes antenas, (hay que    *.\n')
fprintf('*  tener presente que el algoritmo DMI en su         *.\n')
fprintf('*  estructura internamente produce un ruido que puede*.\n')
fprintf('*  ser de 2, 3 y hasta 10 %  aparte de este ruido    *.\n')
fprintf('*  generado por el mismo programa DMI se puede       *.\n')
```

```
fprintf('*  ingresar ruidos adicionales en el programa a la   *.\n')
fprintf('*  entrada o a la salida del programa.  n(f)         *.\n')
fprintf('*  • el orden del filtro FIR cuyos coeficientes se   *.\n')
fprintf('*  van a ajustar, orden                              *.\n')
fprintf('*  • el número de iteraciones en el aprendizaje, N   *.\n')
fprintf('*  (este número se hará  hasta que cumpla el error   *.\n')
fprintf('*  • la constante de ajuste   u                      *.\n')
fprintf('*  • el vector de pesos inicial. Los voltajes de las *.\n')
fprintf('*  antenas V(f)                                      *.\n')
fprintf('*  • el error deseado puede ser ingresado por teclado*.\n')
fprintf('*    error                                           *.\n')
fprintf('*                                                    *.\n')
fprintf('*  A su salida devolverá:                            *.\n')
fprintf('*  • la salida en cada iteración durante el proceso  *.\n')
fprintf('*  de ajuste, y(t)                                   *.\n')
fprintf('*  • la curva de aprendizaje (grafico de la curva)   *.\n')
fprintf('*  • los vectores de pesos obtenidos durante el pro- *.\n')
fprintf('*  ceso. W(t)                                        *.\n')
fprintf('*  • la Potencia de la antena P(k)                   *.\n')
fprintf('*  • el numero de iteraciones en que converge la     *.\n')
fprintf('*    curva y se cumple el valor del error            *.\n')
fprintf('*                                                    *.\n')
fprintf('* alumno: Juan Francisco Alvarez Alvarado            *.\n')
fprintf('* alumno: Maribel del Rosario Chuez Gonzales         *.\n')
fprintf('*                                                    *.\n')
fprintf('* Profesor :Ing. Pedro Vargas                        *.\n')
fprintf('*                                                    *.\n')
fprintf('******************************************************.\n')
t=input('teclee enter para proseguir');
 clc
% ingresamos el orden del filtro, cuyo coeficiente se va ha ajustar
orden=input(' ingrese el orden del filtro, cuyo coeficiente se va ha
ajustar....')
% ingresamos el numero de antenas del sistema
% los valores de h (pesos iniciales que recomendamos usar para el
ejemplo
% son h= 0.26; 0.25; 0.21 ;0.18; 0.15
% valores recomendados por Juan Alvarez y Maribel Chuez
% para este ejemplo
% se pueden ingresar valores cercanos a estos, de lo contrario aumenta
el
% error
% nota :ingresar valores que esten dentro del maximo de la curva, por
% logica si ingresamos como peso por ejemplo 2 y la curva llega hasta 1
% nunca lo encontrara, nunca habra convergencia.

% ingresamos el numero de antenas del sistema
disp(' ingrese el numero de antenas entre 1 y 5');
x=input(' ingrese el numero de antenas entre 1,2,3,4 o 5....');
```

```
if x==1
    D(1)=input('ingrese el voltaje1 (en V)..... voltaje1=');
end

if x==2
    D(1)=input('ingrese el voltaje1 (en V)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
end

if x==3
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
end

if x==4
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
end
if x==5
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
end
% ingresamos el ruido que presenta cada antena del sistema
% Si existe ruido adicional al ocasionado por el sistema
if x==1
    E(1)=input(' ingrese el ruido en la antena1.....ruido1=');
    ruido=E(1);
end

if x==2
    E(1)=input('ingrese el ruido en la antena1.....ruido1==');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    ruido=(E(1)+E(2))/2;
end

if x==3
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    ruido=(E(1)+E(2)+E(3))/3;
end
```

```
if x==4
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    ruido=(E(1)+E(2)+E(3)+E(4))/4;
end
if x==5
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    ruido=(E(1)+E(2)+E(3)+E(4)+E(5))/5;
end
opcion_ruido=2;
disp('si desea que el ruido adicional vaya a la señal a la entrada
digite 1');
disp(' de lo contrario se considerara que el ruido adicional se
agregara a la señal de referencia');
opcion_ruido=input('opcion_ruido=');
error=input('ingrese el error del sistema=');
N=2000;
j=0;
% vamos a considerar que todas las antenas tienen la misma señal a la
% entrada del filtro por ejemplo  entrada=sin((2*pi*k)/N)
if x==1
    h=[D(1)]
    p=[D(1)]
end
if x==2
    h=[D(1);D(2)]
    p=[(D(1)+D(2))/2]+[0.8*(D(1)+D(2))/2];
end
if x==3
    h=[D(1);D(2);D(3)]
    p=[(D(1)+D(2)+D(3))/3]+[0.8*(D(1)+D(2)+D(3))/3];
end
if x==4
    h=[D(1);D(2);D(3);D(4)]
    p=[(D(1)+D(2)+D(3)+D(4))/4]+[0.8*(D(1)+D(2)+D(3)+D(4))/4];
end
if x==5
    h=[D(1);D(2);D(3);D(4);D(5)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5))/5]+[0.8*(D(1)+D(2)+D(3)+D(4)+D(5))/5];
end
% numero de iteraciones del proceso hasta que se cumpla el error k, 400
% k es el numero de muestras que se evaluan en la funcion seno
```

```
M=30;
for k=1:400
entrada(k)=sin((2*pi*k)/M);
% senal_referencia(k)=cos((2*pi*k)/M); % señal de referencia en algunos
libros se la llama señal deseada
end
% señal a la entrada del filtro, genera K valores aleatorios de muestra
entrada';
entrada=entrada';
if opcion_ruido==1
    entrada=entrada+ruido;
end
n = sqrt(0.05)*randn(400,1); % ruido a la entrada del filtro, genera K
valores aleatorios de muestra
[b,a] = butter(2,0.25) % usaremos otra forma para calcular a y b
Gz = tf(b,a,-1) % la funcion de transferencia tf funciona en matlab 7
% esta funcion permite resolver la transformada z,
% el primer orden evalua valores de pesos con la funcion ldiv
%h=ldiv(b,a,orden)';
% if you use ldiv this will give h :filter weights to be
% ingreso de los voltajes en la señal de entrada (estos voltajes seran
% los valores de pesos iniciales que serviran para hallar los pesos W
y = lsim(Gz,entrada); % simula respuestas arbitrarias de entrada,
relaciona la señal Gz con la entrada para agregar el ruido
% este es el ruido que se genera internamente por el metodo LMS
% por lo general el valor de ruido que genera el algoritmo LMS puede
llegar al 10 % de la señal
n = n * std(y)/(10*std(n)); % aqui se genera el ruido del sistema
% el valor a la entrada del filtro es la suma de la señal entrante mas
el ruido
senal_referencia=y+n; % en la mayoría de los programas lms, rls,
filtros, se coloca el ruido del sistema en la señal de referencia
if opcion_ruido>1
    senal_referencia=senal_referencia+ruido;
end
totallength=size(senal_referencia,1) % da el largo de la funcion d
entrada

% conociendo los voltajes de las antenas podemos determinar la potencia
% en ellas
% potencia de la señal a la salida (potencia del sistema
potencia=mean(senal_referencia.^2)
if x==1
% tambien calcularemos la potencia para cada una de las antenas
% antena1
entrada1=y+n+E(1);
potencia1=mean(entrada1.^2)
end
if x==2
```

```
% antena1
entrada1=D(1)+n+E(1);
potencia1=mean(entrada1.^2)
% antena2
entrada2=D(2)+n+E(2);
potencia2=mean(entrada2.^2)
end
if x==3
% antena1
entrada1=D(1)+n+E(1);
potencia1=mean(entrada1.^2)
% antena2
entrada2=D(2)+n+E(2);
potencia2=mean(entrada2.^2)
% antena3
entrada3=D(3)+n+E(3);
potencia3=mean(entrada3.^2)
end
if x==4
% antena1
entrada1=D(1)+n+E(1);
potencia1=mean(entrada1.^2)
% antena2
entrada2=D(2)+n+E(2);
potencia2=mean(entrada2.^2)
% antena3
entrada3=D(3)+n+E(3);
potencia3=mean(entrada3.^2)
% antena4
entrada4=D(4)+n+E(4);
potencia4=mean(entrada4.^2)
end
if x==5
% antena1
entrada1=D(1)+n+E(1);
potencia1=mean(entrada1.^2)
% antena2
entrada2=D(2)+n+E(2);
potencia2=mean(entrada2.^2)
% antena3
entrada3=D(3)+n+E(3);
potencia3=mean(entrada3.^2)
% antena4
entrada4=D(4)+n+E(4);
potencia4=mean(entrada4.^2)
% antena5
entrada5=D(5)+n+E(5);
potencia5=mean(entrada5.^2)
end
```

```
% calculo de la cota de la contante de ajuste
cota=1/((orden+1)*potencia);
%begin of algorithm
Rxx=0;
rxx=0;
% w = zeros ( orden  , 1 )
 w = zeros ( orden  , 1 )
%para 80 puntos  por corrida determinamos los pesos w
N=80
for n = orden : N
    u = entrada(n:-1:n-orden+1)
u(find(u>1))=(u(1)+u(2)+u(3)+u(4)+u(5))/5;% para que los valores de
u(find(u<-1))=(u(1)+u(2)+u(3)+u(4)+u(5))/5;% entrada estén dentro de
%la curva en esta función seno
%u(find(u>1))=0.886;
%u(find(u<-1))=-0.886;
    u
    n
    y(n)= w' * u;  % y(n) señal a la salida
    % e(n) = senal_referencia(n) - y(n);
    e(n)=senal_referencia(n)-y(n);
    j=j+1 % nos da el numero de iteraciones hasta que se cumpla el
error
     if e(n)<error
        fprintf('el numero de iteraciones hasta que se completo el
error ');
        j
     end

    if n < 20
        mu=cota/2; % los valores de mu no son usados por el
                    algoritmo dmi en la determinación de los pesos.
    else
        mu=cota/4;
    end
    T=u*u';
    if j==1
        Rxx=T*0;
    end
    Rxx=Rxx+T;
    T2=senal_referencia(n)*u;
    if j==1
        rxx=T2*0;
    end
    rxx=rxx+T2;
    T3=inv(Rxx);
    w = (inv(Rxx))*rxx;  % ecuacion de pesos para el algoritmo dmi
```

```
w(find(w>1))=(w(1)+w(2)+w(3)+w(4)+w(5))/5; % valores de peso maximo
para que la
    w(find(w<-1))=(w(1)+w(2)+w(3)+w(4)+w(5))/5; % la funcion seno no se
pierda
w

% w=w+mu*u*e(n)   ecuacion de pesos en el algoritmo lms
    % algunos autores usan el valor de w = w + 2*mu * u * e(n); para
    % evaluar los pesos
end
% chequeo de resultados de error
N=200
for n =  N+1 : totallength
    u = entrada(n:-1:n-orden+1) ;
    y(n) = w' * u ;   % y(n) señal a la salida
    % e(n) = senal_referencia(n) - y(n) ;
    e(n)=senal_referencia(n)-y(n);
end

hold on
plot(senal_referencia)
plot(y,'r');
title('salida del sistema') ;
xlabel('numero de muestras ')
ylabel('valor estimado a la salida ')
figure
semilogy((abs(e))) ;
title('curva de error') ;
xlabel('numero de muestras ')
ylabel('valor de error')
figure
plot(h, 'k+')
hold on
plot(w, 'r*')
legend('pesos actuales','pesos estimados')
title('comparacion de los pesos actuales con los pesos estimados') ;
axis([0 6 0.05 0.35])
```

Problema2_dmi

```
function[caratula]=problema2_dmi;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
% Algoritmo DMI
% Autores: Juan Alvarez , Maribel Chuez
%
%Escuela Superior Politecnica del Litoral
%Facultad de Electricidad y Computación
%Guayaquil, Ecuador
% email :juan_alvarez8@latinmail.com
% email : mchuez@ceibo.espol.edu.ec
% Webpage : proximamente
% Date    : Domingo 26 de Julio del 2009
% Version : 1.0.1
% Referencias : S. Haykin, Adaptive Filter Theory. 3rd edition, Upper
Saddle River, NJ: Prentice-Hall, 1996.
%
%clear all % esta funcion borra los valores anteriores que esten
memoria %
%close all % esta funcion borra cualquier ventana anterior activa %
%hold off   % esta funcion encera cualquier grafico que estuviera
activo %
 fprintf('*****************************************************.\n')
fprintf('*    ESCUELA SUPERIOR POLITECNICA DEL LITORAL        *.\n')
fprintf('*                                                    *.\n')
fprintf('* PROBLEMA2_dmi.- Determinar los pesos, el numero de *.\n')
fprintf('* iteraciones y la potencia de un grupo de antenas   *.\n')
fprintf('* inteligentes utilizando el algoritmo DMI (algoritmo*.\n')
fprintf('* adaptativo de inversión de matriz directa, las     *.\n')
fprintf('* antenas están distribuidas en forma de celdas      *.\n')
fprintf('* para lo cual requeriremos de los siguientes datos: *.\n')
fprintf('*                                                    *.\n')
fprintf('*  • la señal aplicada a la entrada del filtro,      *.\n')
fprintf('*   ej  Xn=sin((2*pi*k)/N)                           *.\n')
fprintf('*  • el numero de antenas  x                         *.\n')
fprintf('*  • la señal de referencia, d(k)=cos((2*pi*k)/N);   *.\n')
fprintf('*  • el ruido de las diferentes antenas, (hay que    *.\n')
fprintf('*  tener presente que el algoritmo CMA en su         *.\n')
fprintf('*  estructura internamente produce un ruido que puede*.\n')
fprintf('*  ser de 2, 3 y hasta 10 %  aparte de este ruido    *.\n')
fprintf('*  generado por el mismo programa CMA se puede       *.\n')
```

```
fprintf('*  ingresar ruidos adicionales en el programa a la   *.\n')
fprintf('*  entrada o a la salida del programa.  n(f)         *.\n')
fprintf('*  • el orden del filtro FIR cuyos coeficientes se   *.\n')
fprintf('*  van a ajustar, orden                              *.\n')
fprintf('*  • el número de iteraciones en el aprendizaje, N   *.\n')
fprintf('*  (este número se hará  hasta que cumpla el error   *.\n')
fprintf('*  • la constante de ajuste   u                      *.\n')
fprintf('*  • el vector de pesos inicial. Los voltajes de las *.\n')
fprintf('*  antenas V(f)                                      *.\n')
fprintf('*  • el error deseado puede ser ingresado por teclado*.\n')
fprintf('*    error                                           *.\n')
fprintf('*                                                    *.\n')
fprintf('*  A su salida devolverá:                            *.\n')
fprintf('*  • la salida en cada iteración durante el proceso  *.\n')
fprintf('*  de ajuste, y(t)                                   *.\n')
fprintf('*  • la curva de aprendizaje (grafico de la curva)   *.\n')
fprintf('*  • los vectores de pesos obtenidos durante el pro- *.\n')
fprintf('*  ceso. W(t)                                        *.\n')
fprintf('*  • la Potencia de la antena P(k)                   *.\n')
fprintf('*  • el numero de iteraciones en que converge la     *.\n')
fprintf('*    curva y se cumple el valor del error            *.\n')
fprintf('*                                                    *.\n')
fprintf('* alumno: Juan Francisco Alvarez Alvarado            *.\n')
fprintf('* alumno: Maribel del Rosario Chuez Gonzales         *.\n')
fprintf('*                                                    *.\n')
fprintf('* Profesor :Ing. Pedro Vargas                        *.\n')
fprintf('*                                                    *.\n')
fprintf('******************************************************.\n')
t=input('teclee enter para proseguir');
 clc
% ingresamos el orden del filtro, cuyo coeficiente se va ha ajustar
orden=input(' ingrese el orden del filtro, cuyo coeficiente se va ha
ajustar....')
% ingresamos el numero de antenas del sistema
disp(' ingrese el numero de antenas entre 1 y 5');
x=input(' ingrese el numero de antenas entre 1,2,3,4 o 5....');
% los valores recomendado por la central matlab como valores de los
pesos, son
% h=0.0976;0.2873;0.3360;0.2210;0.0964
if x==1
    D(1)=input('ingrese el voltaje1 (en V)..... voltaje1=');
end

if x==2
    D(1)=input('ingrese el voltaje1 (en V)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
end

if x==3
```

```
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
end

if x==4
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
end
if x==5
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
end
% ingresamos el ruido que presenta cada antena del sistema
% Si existe ruido adicional al ocasionado por el sistema
if x==1
    E(1)=input(' ingrese el ruido en la antena1.....ruido1=');
    ruido=E(1);
end

if x==2
    E(1)=input('ingrese el ruido en la antena1.....ruido1==');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    ruido=(E(1)+E(2))/2;
end

if x==3
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    ruido=(E(1)+E(2)+E(3))/3;
end

if x==4
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    ruido=(E(1)+E(2)+E(3)+E(4))/4;
end
if x==5
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
```

```
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    ruido=(E(1)+E(2)+E(3)+E(4)+E(5))/5;
end
opcion_ruido=2;
disp('si desea que el ruido adicional vaya a la señal a la entrada
digite 1');
disp('de lo contrario se considerara que el ruido adicional se agregara
a la señal de referencia');
opcion_ruido=input('opcion_ruido=');
error=input('ingrese el error del sistema=');
% numero de iteraciones del proceso hasta que se cumpla el error
N=2000;
j=0;
% vamos a considerar que todas las antenas tienen la misma señal a la
% entrada del filtro por ejemplo  entrada=sin((2*pi*k)/N),entrada =
randn(N,1)
if x==1
    h=[D(1)]
    p=[D(1)]
end
if x==2
    h=[D(1);D(2)]
    p=[(D(1)+D(2))/2]+[0.8*(D(1)+D(2))/2];
end
if x==3
    h=[D(1);D(2);D(3)]
    p=[(D(1)+D(2)+D(3))/3]+[0.8*(D(1)+D(2)+D(3))/3];
end
if x==4
    h=[D(1);D(2);D(3);D(4)]
    p=[(D(1)+D(2)+D(3)+D(4))/4]+[0.8*(D(1)+D(2)+D(3)+D(4))/4];
end
if x==5
    h=[D(1);D(2);D(3);D(4);D(5)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5))/5]+[0.8*(D(1)+D(2)+D(3)+D(4)+D(5))/5];
end
entrada = randn(N,1); % señal a la entrada del filtro, genera 2000
valores aleatorios de muestra
if opcion_ruido==1
    entrada=entrada+ruido
end
n = randn(N,1); % ruido a la entrada del filtro, genera 2000 valores
aleatorios de muestra
[b,a] = butter(2,0.25);
Gz = tf(b,a,-1) % la funcion de transferencia tf funciona en matlab 7
% esta funcion permite resolver la transformada z,
% el primer orden evalua valores de pesos con la funcion ldiv
```

```
%h=ldiv(b,a,orden)';
% if you use ldiv this will give h :filter weights to be
% ingreso de los voltajes en la señal de entrada (estos voltajes seran
% los valores de pesos iniciales que serviran para hallar los pesos W
y = lsim(Gz,entrada); % simula respuestas arbitrarias de entrada,
relaciona la señal Gz con inp
% para agregar el ruido
% este es el ruido que se genera internamente por el metodo LMS
% por lo general el valor de ruido que genera el algoritmo LMS puede
llegar al 10 % de la señal
n = n * std(y)/(10*std(n)); % aqui se genera el ruido del sistema
% el valor a la entrada del filtro es la suma de la señal entrante mas
el ruido
senal_referencia = y + n;
if opcion_ruido>1
    senal_referencia=senal_referencia+ruido
end
totallength=size(senal_referencia,1); % da el largo de la funcion d
% conociendo los voltajes de las antenas podemos determinar la potencia
% en ellas
% potencia de la señal a la salida (potencia del sistema
potencia=mean(senal_referencia.^2)
if x==1
% tambien calcularemos la potencia para cada una de las antenas
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
end
if x==2
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
end
if x==3
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
end
if x==4
% antena1
```

```
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
end
if x==5
% antena1
d1=D(1)+n+E(1);
potencia1=mean(d1.^2)
% antena2
d2=D(2)+n+E(2);
potencia2=mean(d2.^2)
% antena3
d3=D(3)+n+E(3);
potencia3=mean(d3.^2)
% antena4
d4=D(4)+n+E(4);
potencia4=mean(d4.^2)
% antena5
d5=D(5)+n+E(5);
potencia5=mean(d5.^2)
end
% calculo de la cota de la contante de ajuste
cota=1/((orden+1)*potencia);

%begin of algorithm
Rxx=0;
rxx=0;
% w = zeros ( orden  , 1 )
 w = zeros ( orden  , 1 )
%para 80 puntos  por corrida determinamos los pesos w
N=80
for n = orden : N
    u = entrada(n:-1:n-orden+1);
u(find(u>1))=(u(1)+u(2)+u(3)+u(4)+u(5))/5;% para que los valores de
u(find(u<-1))=(u(1)+u(2)+u(3)+u(4)+u(5))/5;% entrada estén dentro de
%la curva en esta función seno, para orden=5
    u
    n
    y(n)= w' * u;  % y(n) señal a la salida
    % e(n) = senal_referencia(n) - y(n);
    e(n)=senal_referencia(n)-y(n);
```

```
    j=j+1 % nos da el numero de iteraciones hasta que se cumpla el
error
     if e(n)<error
        fprintf('el numero de iteraciones hasta que se completo el
error ');
        j
     end

    if n < 20
        mu=cota/2;
    else
        mu=cota/4;
    end
% los valores de mu no son usados por el algoritmo dmi en la
determinación de los pesos.

    T=u*u';
    if j==1
        Rxx=T*0;
    end
    Rxx=Rxx+T;
    T2=senal_referencia(n)*u;
    if j==1
        rxx=T2*0;
    end
    rxx=rxx+T2;
    T3=inv(Rxx);
    w = (inv(Rxx))*rxx;  % ecuacion de pesos para el algoritmo dmi
w(find(w>1))=(w(1)+w(2)+w(3)+w(4)+w(5))/5; % valores de peso maximo
para que la función seno no se pierda
    w(find(w<-1))=(w(1)+w(2)+w(3)+w(4)+w(5))/5;
w

% w=w+mu*u*e(n)   ecuacion de pesos en el algoritmo lms
    % algunos autores usan el valor de w = w + 2*mu * u * e(n); para
    % evaluar los pesos
end
% chequeo de resultados de error
N=200
for n =  N+1 : totallength
    u = entrada(n:-1:n-orden+1) ;
    y(n) = w' * u ;   % y(n) señal a la salida
    % e(n) = senal_referencia(n) - y(n) ;
    e(n)=senal_referencia(n)-y(n);
end

hold on
plot(senal_referencia)
plot(y,'r');
```

```
title('salida del sistema') ;
xlabel('numero de muestras ')
ylabel('valor estimado a la salida ')
figure
semilogy((abs(e))) ;
title('curva de error') ;
xlabel('numero de muestras ')
ylabel('valor de error')
figure
plot(h, 'k+')
hold on
plot(w, 'r*')
legend('pesos actuales','pesos estimados')
title('comparacion de los pesos actuales con los pesos estimados') ;
axis([0 6 0.05 p])
```

```
function[caratula]=problema3_dmi;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
% Algoritmo DMI
% Autores: Juan Alvarez , Maribel Chuez
%
%Escuela Superior Politecnica del Litoral
%Facultad de Electricidad y Computación
%Guayaquil, Ecuador
% email :juan_alvarez8@latinmail.com
% email : mchuez@ceibo.espol.edu.ec
% Webpage : proximamente
% Date    : Domingo 26 de Julio del 2009
% Version : 1.0.1
% Referencias : S. Haykin, Adaptive Filter Theory. 3rd edition, Upper
Saddle River, NJ: Prentice-Hall, 1996.
%
%clear all % esta funcion borra los valores anteriores que esten
memoria %
%close all % esta funcion borra cualquier ventana anterior activa %
%hold off   % esta funcion encera cualquier grafico que estuviera
activo %
 fprintf('*****************************************************.\n')
fprintf('*    ESCUELA SUPERIOR POLITECNICA DEL LITORAL        *.\n')
fprintf('*                                                    *.\n')
fprintf('* PROBLEMA3_dmi.- Determinar los pesos, el numero de *.\n')
fprintf('* iteraciones y la potencia de un grupo de antenas   *.\n')
fprintf('* inteligentes utilizando el algoritmo DMI (algoritmo*.\n')
fprintf('* adaptativo de inversión de matriz directa, las     *.\n')
fprintf('* antenas están distribuidas en forma de celdas      *.\n')
fprintf('* para lo cual requeriremos de los siguientes datos: *.\n')
fprintf('*                                                    *.\n')
fprintf('*  • la señal aplicada a la entrada del filtro,      *.\n')
fprintf('*   ej  Xn=sin((2*pi*k)/N)                           *.\n')
fprintf('*  • el numero de antenas  x                         *.\n')
fprintf('*  • la señal de referencia, d(k)=cos((2*pi*k)/N);   *.\n')
fprintf('*  • el ruido de las diferentes antenas, (hay que    *.\n')
fprintf('*  tener presente que el algoritmo CMA en su         *.\n')
fprintf('*  estructura internamente produce un ruido que puede*.\n')
fprintf('*  ser de 2, 3 y hasta 10 %  aparte de este ruido    *.\n')
fprintf('*  generado por el mismo programa CMA se puede       *.\n')
fprintf('*  ingresar ruidos adicionales en el programa a la   *.\n')
fprintf('*  entrada o a la salida del programa.  n(f)         *.\n')
fprintf('*  • el orden del filtro FIR cuyos coeficientes se   *.\n')
fprintf('*  van a ajustar, orden                              *.\n')
fprintf('*  • el número de iteraciones en el aprendizaje, N   *.\n')
fprintf('*  (este número se hará  hasta que cumpla el error   *.\n')
fprintf('*  • la constante de ajuste   u                      *.\n')
```

```
fprintf('*  • el vector de pesos inicial. Los voltajes de las *.\n')
fprintf('*  antenas V(f)                                       *.\n')
fprintf('*  • el error deseado puede ser ingresado por teclado*.\n')
fprintf('*    error                                            *.\n')
fprintf('*                                                     *.\n')
fprintf('*  A su salida devolverá:                             *.\n')
fprintf('*  • la salida en cada iteración durante el proceso   *.\n')
fprintf('*  de ajuste, y(t)                                    *.\n')
fprintf('*  • la curva de aprendizaje (grafico de la curva)    *.\n')
fprintf('*  • los vectores de pesos obtenidos durante el pro-  *.\n')
fprintf('*  ceso. W(t)                                         *.\n')
fprintf('*  • la Potencia de la antena P(k)                    *.\n')
fprintf('*  • el numero de iteraciones en que converge la      *.\n')
fprintf('*    curva y se cumple el valor del error             *.\n')
fprintf('*                                                     *.\n')
fprintf('* alumno: Juan Francisco Alvarez Alvarado             *.\n')
fprintf('* alumno: Maribel del Rosario Chuez Gonzales          *.\n')
fprintf('*                                                     *.\n')
fprintf('* Profesor :Ing. Pedro Vargas                         *.\n')
fprintf('*                                                     *.\n')
fprintf('*******************************************************.\n')
t=input('teclee enter para proseguir');
 clc
% ingresamos el orden del filtro, cuyo coeficiente se va ha ajustar
orden=input(' ingrese el orden del filtro, cuyo coeficiente se va ha
ajustar....')
% ingresamos el numero de antenas del sistema
disp(' ingrese el numero de antenas entre 1 y 5');
x=input(' ingrese el numero de antenas entre 1,2,3,4 o 5....');
if x==1
    D(1)=input('ingrese el voltaje1 (en V)..... voltaje1=');
end

if x==2
    D(1)=input('ingrese el voltaje1 (en V)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
end

if x==3
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
end

if x==4
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
```

```
end
if x==5
    D(1)=input('ingrese el voltaje1 (en v)..... voltaje1=');
    D(2)=input(' ingrese la voltaje2 (en V).... voltaje2=');
    D(3)=input(' ingrese la voltaje3 (en V).....voltaje3=');
    D(4)=input(' ingrese la voltaje4 (en v)..... voltaje4=');
    D(5)=input(' ingrese la voltaje5 (en v)..... voltaje5=');
end
% ingresamos el ruido que presenta cada antena del sistema
% Si existe ruido adicional al ocasionado por el sistema
if x==1
    E(1)=input(' ingrese el ruido en la antena1.....ruido1=');
    ruido=E(1);
end

if x==2
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    ruido=(E(1)+E(2))/2;
end

if x==3
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    ruido=(E(1)+E(2)+E(3))/3;
end

if x==4
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    ruido=(E(1)+E(2)+E(3)+E(4))/4;
end
if x==5
    E(1)=input('ingrese el ruido en la antena1.....ruido1=');
    E(2)=input('ingrese el ruido en la antena2.....ruido2=');
    E(3)=input('ingrese el ruido en la antena3.....ruido3=');
    E(4)=input('ingrese el ruido en la antena4.....ruido4=');
    E(5)=input('ingrese el ruido en la antena5.....ruido5=');
    ruido=(E(1)+E(2)+E(3)+E(4)+E(5))/5;
end
opcion_ruido=2;
disp('si desea que el ruido adicional vaya a la señal a la entrada
digite 1');
disp(' de lo contrario se considerara que el ruido adicional se
agregara a la señal de referencia');
opcion_ruido=input('opcion_ruido=');
```

```
error=input('ingrese el error del sistema=');
% numero de iteraciones del proceso hasta que se cumpla el error
opcion_senal=2;
disp('si desea que la señal de entrada se use como señal de referencia
digite 1');
disp(' de lo contrario se considerara otra señal diferente como señal
de referencia ');
opcion_senal=input('opcion_señal=');
% numero de iteraciones del proceso hasta que se cumpla el error
N=2000;
j=0;
% vamos a considerar que todas las antenas tienen la misma señal a la
% entrada del filtro por ejemplo  entrada=sin((2*pi*k)/N)
if x==1
    h=[D(1)]
    p=[D(1)]
end
if x==2
    h=[D(1);D(2)]
    p=[(D(1)+D(2))/2]+[0.8*(D(1)+D(2))/2];
end
if x==3
    h=[D(1);D(2);D(3)]
    p=[(D(1)+D(2)+D(3))/3]+[0.8*(D(1)+D(2)+D(3))/3];
end
if x==4
    h=[D(1);D(2);D(3);D(4)]
    p=[(D(1)+D(2)+D(3)+D(4))/4]+[0.8*(D(1)+D(2)+D(3)+D(4))/4];
end
if x==5
    h=[D(1);D(2);D(3);D(4);D(5)]

p=[(D(1)+D(2)+D(3)+D(4)+D(5))/5]+[0.8*(D(1)+D(2)+D(3)+D(4)+D(5))/5];
end
M=30;
for k=1:400
entrada(k)=sin((2*pi*k)/M);
% senal_referencia(k)=cos((2*pi*k)/M); % señal de referencia en algunos
libros se la llama señal deseada
end
for k=1:400
%entrada(k)=sin((2*pi*k)/M);
senal_referencia(k)=cos((2*pi*k)/M); % señal de referencia en algunos
libros se la llama señal deseada
end
% señal a la entrada del filtro, genera 400 valores aleatorios de
muestra
entrada';
entrada=entrada';
```

```
if opcion_ruido==1
    entrada=entrada+ruido;
end
n = sqrt(0.05)*randn(400,1); % ruido a la entrada del filtro, genera
400 valores aleatorios de muestra
[b,a] = butter(2,0.25) % usaremos otra forma para calcular a y b
Gz = tf(b,a,-1) % la funcion de transferencia tf funciona en matlab 7
% esta funcion permite resolver la transformada z,
% el primer orden evalua valores de pesos con la funcion ldiv
%h=ldiv(b,a,orden)';
% if you use ldiv this will give h :filter weights to be
% ingreso de los voltajes en la señal de entrada (estos voltajes seran
% los valores de pesos iniciales que serviran para hallar los pesos W
y = lsim(Gz,entrada); % simula respuestas arbitrarias de entrada,
relaciona la señal Gz con la entrada
% para agregar el ruido
% este es el ruido que se genera internamente por el metodo LMS
% por lo general el valor de ruido que genera el algoritmo LMS puede
llegar al 10 % de la señal
n = n * std(y)/(10*std(n)); % aqui se genera el ruido del sistema
% el valor a la entrada del filtro es la suma de la señal entrante mas
el ruido
if opcion_senal==1
senal_referencia = y + n;
end
if opcion_senal>1
    senal_referencia';
    senal_referencia=senal_referencia';
    senal_referencia=senal_referencia+n;
end
if opcion_ruido>1
    senal_referencia=senal_referencia+ruido;
end
totallength=size(senal_referencia,1) % da el largo de la funcion d
entrada

% conociendo los voltajes de las antenas podemos determinar la potencia
% en ellas
% potencia de la señal a la salida (potencia del sistema
potencia=mean(senal_referencia.^2)
if x==1
% tambien calcularemos la potencia para cada una de las antenas
% antena1
entrada1=D(1)+n+E(1);
potencia1=mean(entrada1.^2)
end
if x==2
% antena1
entrada1=D(1)+n+E(1);
```

```
potencia1=mean(entrada1.^2)
% antena2
entrada2=D(2)+n+E(2);
potencia2=mean(entrada2.^2)
end
if x==3
% antena1
entrada1=D(1)+n+E(1);
potencia1=mean(entrada1.^2)
% antena2
entrada2=D(2)+n+E(2);
potencia2=mean(entrada2.^2)
% antena3
entrada3=D(3)+n+E(3);
potencia3=mean(entrada3.^2)
end
if x==4
% antena1
entrada1=D(1)+n+E(1);
potencia1=mean(entrada1.^2)
% antena2
entrada2=D(2)+n+E(2);
potencia2=mean(entrada2.^2)
% antena3
entrada3=D(3)+n+E(3);
potencia3=mean(entrada3.^2)
% antena4
entrada4=D(4)+n+E(4);
potencia4=mean(entrada4.^2)
end
if x==5
% antena1
entrada1=D(1)+n+E(1);
potencia1=mean(entrada1.^2)
% antena2
entrada2=D(2)+n+E(2);
potencia2=mean(entrada2.^2)
% antena3
entrada3=D(3)+n+E(3);
potencia3=mean(entrada3.^2)
% antena4
entrada4=D(4)+n+E(4);
potencia4=mean(entrada4.^2)
% antena5
entrada5=D(5)+n+E(5);
potencia5=mean(entrada5.^2)
end

% calculo de la cota de la contante de ajuste
```

```
cota=1/((orden+1)*potencia);
%para 60 puntos  por corrida determinamos los pesos w
%begin of algorithm
Rxx=0;
rxx=0;
% w = zeros ( orden  , 1 )
 w = zeros ( orden  , 1 )
N=80
for n = orden : N
    u = entrada(n:-1:n-orden+1);
u(find(u>1))=(u(1)+u(2)+u(3)+u(4)+u(5))/5;% para que los valores de
u(find(u<-1))=(u(1)+u(2)+u(3)+u(4)+u(5))/5;% entrada estén dentro de
%la curva en esta función seno
%u(find(u>1))=0.886;
%u(find(u<-1))=-0.886;
    u
    n
    y(n)= w' * u;  % y(n) señal a la salida
    % e(n) = senal_referencia(n) - y(n);
    e(n)=senal_referencia(n)-y(n);
    j=j+1 % nos da el numero de iteraciones hasta que se cumpla el
error
     if e(n)<error
        fprintf('el numero de iteraciones hasta que se completo el
error ');
        j
     end

    if n < 20
        mu=cota/2; % los valores de mu no son usados por el algoritmo
dmi en la determinación de los pesos.
    else
        mu=cota/4;
    end
    T=u*u';
    if j==1
        Rxx=T*0;
    end
    Rxx=Rxx+T;
    T2=senal_referencia(n)*u;
    if j==1
        rxx=T2*0;
    end
    rxx=rxx+T2;
    T3=inv(Rxx);
    w = (inv(Rxx))*rxx;  % ecuacion de pesos para el algoritmo dmi
w(find(w>1))=(w(1)+w(2)+w(3)+w(4)+w(5))/5; % valores de peso maximo
para que la función seno no se pierda
    w(find(w<-1))=(w(1)+w(2)+w(3)+w(4)+w(5))/5;
```

```
w

% w=w+mu*u*e(n)   ecuacion de pesos en el algoritmo lms
    % algunos autores usan el valor de w = w + 2*mu * u * e(n); para
    % evaluar los pesos
end
% chequeo de resultados de error
N=200
for n =  N+1 : totallength
    u = entrada(n:-1:n-orden+1) ;
    y(n) = w' * u ;   % y(n) señal a la salida
    % e(n) = senal_referencia(n) - y(n) ;
    e(n)=senal_referencia(n)-y(n);
end
hold on
plot(senal_referencia)
plot(y,'r');
title('salida del sistema') ;
xlabel('numero de muestras ')
ylabel('valor estimado a la salida ')
figure
semilogy((abs(e))) ;
title('curva de error') ;
xlabel('numero de muestras ')
ylabel('valor de error')
figure
plot(h, 'k+')
hold on
plot(w, 'r*')
legend('pesos actuales','pesos estimados')
title('comparacion de los pesos actuales con los pesos estimados') ;
% axis([coordenada eje x- coordenada eje x+  coordenada y- coordenada
y+])
axis([0 6 -1 1])
```

ANEXO G

CONCEPTOS IMPORTANTES SOBRE FILTROS ADAPTATIVOS.

En este anexo hacemos un resumen sobre filtros adaptativos, filtro de wiener, filtro de kalman y las principales formulas utilizadas por los algoritmos LMS y RLS.

Filtro adaptativo

La definición de **filtro adaptativo** es un dispositivo que intenta modelizar la relación entre señales en tiempo real de forma iterativa.

Se diferencia de los filtros digitales en que éstos últimos tienen coeficientes invariantes en el tiempo, mientras que un adaptativo puede cambiar su forma de comportarse, es decir pueden cambiar sus coeficientes de acuerdo con un *algoritmo adaptativo*. De hecho no se saben los coeficientes del filtro cuando se diseña, éstos coeficientes son calculados cuando el filtro se implementa y se reajustan automáticamente en cada iteración mientras dura su fase de aprendizaje.

El hecho de que estos filtros no sean invariantes temporales y que tampoco sean lineales hace que su estudio sea más complejo que el de un filtro digital, ya que no se pueden aplicar, salvo en un par de excepciones, las transformaciones en frecuencia, dominio Z, etc.

Implementación

Los filtros adaptativos normalmente se implementan en forma de algoritmos sobre microprocesadores, DSP o FPGA.

La estructura de un filtro adaptativo es un sistema al que le llegan dos señales: x(n) y e(n), ésta última se llama **señal de error** y viene de la resta de una señal a la que se llama **señal deseada**, d(n), y otra que es la **salida del filtro** y(n). A los **coeficientes del filtro** se les llama w(n), que son los que multiplican a la **entrada** x(n) para obtener la salida.

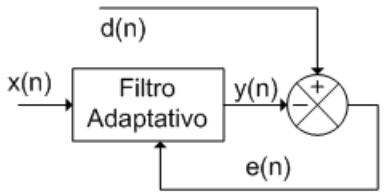


fig g.1 Filtro Adaptativo

Estructura directa de un filtro adaptativo

$$y(n) = w(n)\, x(n) \qquad \textbf{(g.1)}$$

$$e(n) = d(n) - y(n) \qquad \text{(g.2)}$$

En principio el objetivo es hacer que la señal de error sea cero, para ello el sistema debe configurarse para que, a partir de la señal de entrada x(n) se genera la salida y(n) de forma que sea igual a la señal deseada d(n). Cada una de las formas de minimizar ese error es un método de implementar los filtros adaptativos. Por ejemplo se podría proponer minimizar la función de coste $J = e^2(n)$, aplicando la regla delta se obtendrían los nuevos coeficientes como:

$$w(n+1) = w(n) - \alpha \cdot \nabla J \qquad \text{(g.3)}$$

Donde la constante α se usa para ajustar la velocidad convergencia y evitar posibles inestabilidades. Operando se llega a esta otra ecuación:

$$w(n+1) = w(n) - 2 \cdot \alpha \cdot e(n) \cdot x(n) \qquad \text{(g.4)}$$

Algoritmo

Un algoritmo de aprendizaje de un sistema adaptativo podría ser:

1. Inicializar de forma aleatoria los pesos

2. Elegir un valor α

3. Calcular la salida y(n)

4. Calcular el error e(n)

5. Actualizar los pesos con la función de coste elegida

6. Repetir un determinado número de veces desde el punto 3.

Aplicaciones

Según lo explicado antes se puede plantear una pregunta: si ya se tiene la señal deseada ¿Para qué crear un sistema que la genere?.

Una respuesta es para modelizar un sistema. Un ejemplo práctico es una transmisión de fax. En ella, las señales eléctricas pueden verse perturbadas durante el envío, y esta

perturbación puede ser diferente de una transmisión a otra. Una solución para esto es, antes de mandar los datos mandar una señal patrón, que tanto el emisor como el receptor conozcan de antemano, así cuando llegue el receptor tendrá la señal perturbada y sabe la que debía ser. Estas dos señales se aplican a un filtro adaptativo que se configura para compensar las perturbaciones que pudieran haber durante esa transmisión. Como resultado se han modelizado todas las causas que modifican la transmisión y se ha creado un sistema que las compensa.

Otras aplicaciones de los filtros adaptativos son:

- Sistema de identificación
- Cancelación de eco
- Eliminación de ruido blanco

Predictores

Filtro de Wiener

En procesamiento de señales, el **filtro de Wiener** es un filtro propuesto por Norbert Wiener en la década de 1940 y publicado en 1949. Su propósito es, utilizando métodos estadísticos, reducir el ruido presente en la señal de entrada de tal modo que la señal de salida del filtro se aproxime lo más posible (en el sentido cuadrático medio) a una señal deseada (sin ruido). El equivalente en tiempo discreto del trabajo de Wiener fue derivado independientemente por Kolmogorov y publicado en 1941. Por esto, la teoría es a veces referida como *teoría de filtrado de* ***Wiener-Kolmogorov***.

Filtro de Kalman

El **filtro de Kalman** es un algoritmo desarrollado por Rudolf E. Kalman en 1960 que sirve para poder identificar el estado oculto (no medible) de un sistema dinámico lineal, al igual que el observador de Luenberger, pero sirve además cuando el sistema está sometido a ruido blanco aditivo. La diferencia entre ambos es que en el observador de Luenberger, la ganancia K de realimentación del error debe ser elegida "a mano", mientras que el filtro de Kalman es capaz de escogerla de forma óptima cuando se conocen las varianzas de los ruidos que afectan al sistema.

Caso de tiempo discreto:

Se tiene un sistema dado por:

$$x_k = A_{k-1}x_{k-1} + B_{k-1}u_{k-1} + w_{k-1} \qquad \textbf{(g.5)}$$

$$z_k = C_k x_k + v_k \qquad \textbf{(g.6)}$$

donde:

$w_k$es ruido blanco de valor promedio igual a cero y con varianza $Q_k$en el instante k.

$v_k$es ruido blanco de valor promedio igual a cero y con varianza $R_k$en el instante k.

El filtro de Kalman permite identificar el estado x_ka partir de las mediciones anteriores de u_k, z_k, Q_k, R_ky las identificaciones anteriores de $x(t)$.

Caso de tiempo continuo:

Se tiene un sistema dado por:

$$\frac{d}{dt}x(t) = A(t)x(t) + B(t)u(t) + w(t) \qquad \textbf{(g.7)}$$

$$z(t) = C(t)x(t) + v(t) \qquad \textbf{(g.8)}$$

donde:

$w(t)$es ruido blanco de valor promedio igual a cero y con varianza $Q(t)$en el intervalo de tiempo descrito como t.

$v(t)$es ruido blanco de valor promedio igual a cero y con varianza $R(t)$en el intervalo de tiempo descrito como t.

El filtro de Kalman permite identificar el estado $x(t+dt)$a partir de las mediciones anteriores de $u(t), z(t), Q(t), R(t)$y las identificaciones anteriores de $x(t)$.

En el caso de que el sistema dinámico sea no-lineal, es posible usar una modificación del algoritmo llamada "filtro de Kalman extendido", el cual linealiza el sistema en torno al $\hat{x}(t)$ identificado realmente, para calcular la ganancia y la dirección de corrección adecuada. En este caso, en vez de haber matrices A, B y C, hay dos funciones $f(x,u,w)$ y $h(x,v)$ que entregan la transición de estado y la observación (la salida contaminada) respectivamente.

Algoritmo LMS

El **algoritmo LMS** (del inglés, *Least-Mean-Square algorithm*) se usa en filtros adaptativos para encontrar los coeficientes del filtro que permiten obtener el valor esperado mínimo del cuadrado de la señal de error, definida como la diferencia entre la señal deseada y la señal producida a la salida del filtro.

Pertenece a la familia de los algoritmos de gradiente estocástico, es decir, el filtro se adapta en base al error en el instante actual únicamente. Fue inventado en 1960 por el profesor de la Universidad de Stanford Bernard Widrow y su primer estudiante de doctorado, Ted Hoff.

Su importancia radica en que es un algoritmo muy simple. No requiere medidas de las funciones de correlación, ni tampoco inversión de la matriz de correlación.

Un filtro es un proceso mediante el cual a una señal cualquiera se le modifica su contenido espectral. El algoritmo LMS es un algoritmo de filtrado lineal adaptativo que, en general, consiste de dos procesos básicos:

- Un *proceso de filtrado*, que involucra:
  - el cómputo de la salida de un filtro lineal en respuesta a una señal de entrada, y
  - la generación de una estimación del error mediante la comparación de esta salida con la señal deseada.
- Un *proceso adaptativo*, que involucra el ajuste automático de los parámetros del filtro de acuerdo al error estimado.

Cuando se habla de filtros adaptativos, está implícito que los parámetros que caracterizan al filtro, tales como el ancho de banda y frecuencias de los ceros, entre otros, cambian

con el tiempo, esto es, los coeficientes de los filtros adaptativos cambian con el tiempo, en contraposición a los coeficientes de los filtros fijos que son, teóricamente, invariantes con el tiempo.

El algoritmo LMS, para un filtro de orden *M*, puede resumirse de la siguiente manera:

Parámetros: *M* = orden del filtro

μ = tamaño del paso

Inicialización: Si se dispone de información acerca del vector de coeficientes del filtro $\hat{\mathbf{w}}(n)$, usarla para elegir un valor apropiado para $\hat{\mathbf{w}}(0)$. En caso contrario, usar $\hat{\mathbf{w}}(0) = \mathbf{0}$

Datos:

Dados: $\mathbf{u}(n) = [u(n), u(n-1), \ldots, u(n-M+1)]^T$:

señal de entrada en el instante *n* **(g.9)**

d(*n*): señal deseada a la salida del filtro

$\hat{\mathbf{w}}(n+1) = [\hat{w}_0(n+1), \hat{w}_1(n+1), \ldots, \hat{w}_{M-1}(n+1)]^T$: estimación del vector de coeficientes del filtro **(g.10)**

A calcular: en el instante *n* + 1

Cómputo: Para $n = 0, 1, 2, \ldots$, calcular:

$e(n) = d(n) - \hat{\mathbf{w}}^H(n) \cdot \mathbf{u}(n)$: señal de error **(g.11)**

$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \mu\, e^*(n)\mathbf{u}(n)$: **(g.12)**

adaptación de los coeficientes del filtro

El superíndice *T* denota trasposición, el superíndice *H* denota traspuesta conjugada, el asterisco denota conjugación y $\hat{\mathbf{w}}^H(n) \cdot \mathbf{u}(n)$ es la salida del filtro, que se calcula como el producto interno entre el vector de coeficientes del

filtro $\hat{\mathbf{w}}(n)$, cuyos componentes suelen llamarse *pesos* o *weighs*, y el vector de datos de entrada al filtro $\mathbf{u}(n)$

El **algoritmo RLS** (del inglés, *Recursive-Least-Squares algorithm*) se usa en filtros adaptativos para encontrar los coeficientes del filtro que permiten obtener el mínimo cuadrado de la señal de error (definida como la diferencia entre la señal deseada y la señal producida a la salida del filtro) en forma recursiva.

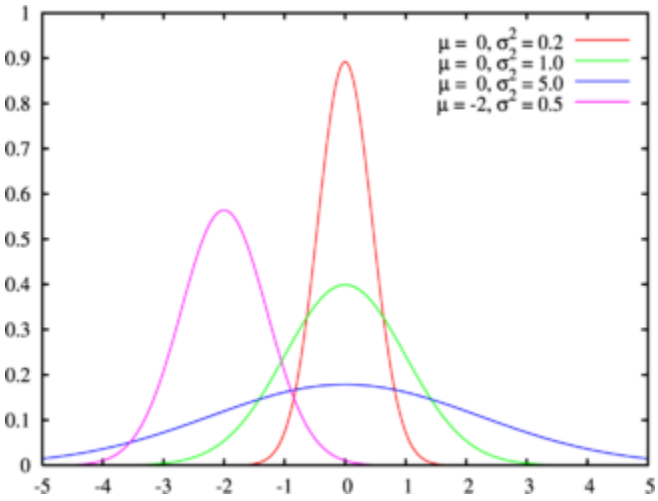
Considérese el modelo de series temporales lineal

$$y(n+1) = wx(n) + e(n) \qquad \textbf{(g.13)}$$

donde $e(n)\tilde{}N(0,1)$ es ruido blanco. Deseamos estimar el parámetro *w* mediante cuadrados mínimos. A cada instante *N* nos referimos al nuevo estimador de cuadrados mínimos por $\hat{w}_N$. A medida que pasa el tiempo, desearíamos evitar repetir el algoritmo para encontrar el nuevo estimador $\hat{w}_{N+1}$en términos de $\hat{w}_N$, sino actualizarlos usando distintas técnicas.

La ventaja del uso del algoritmo RLS es que no hay necesidad de invertir matrices extremadamente grandes, ahorrando así poder de cómputo.

Figura G.2 Función gaussiana



Curvas gaussianas con distintos parámetros

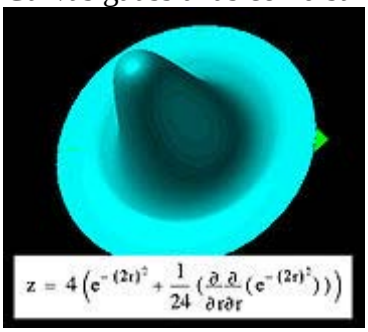


Figura G.3. Forma tridimensional de una función gaussiana.

En matemáticas la **función gaussiana** (en honor a Carl Friedrich Gauss), es una función definida por la expresión:

$$f(x) = ae^{-\frac{(x-b)^2}{2c^2}} \qquad \textbf{(g.14)}$$

donde *a*, *b* y *c* son constantes reales ($a > 0$).

La grafica de la función es simétrica con forma de campana, conocida como **campana de Gauss**. El parámetro *a* es la altura de la campana centrada en el punto *b*, determinando *c* el ancho de la misma.

Las funciones gaussianas se utilizan frecuentemente en estadística correspondiendo, en el caso de que *a* sea igual a $\frac{1}{c\sqrt{2\pi}}$, a la Función de densidad de una variable aleatoria con distribución normal de media $\mu=b$ y varianza $\sigma^2=c^2$.

Las funciones gaussianas con $c^2 = 2$ son las autofunciones de la Transformada de Fourier.. Esto significa que la transformada de Fourier de una función gaussiana no es sólo otra gaussiana, sino además un múltiplo escalar de la función original.

Propiedades

Las gaussianas se encuentran entre las funciones elementales, aunque no poseen primitivas elementales. Sin embargo, el valor exacto de la integral impropia sobre todo el rango real puede derivarse a partir del valor de la integral de Gauss obteniéndose que:

$$\int_{-\infty}^{\infty} ae^{-\frac{(x-b)^2}{2c^2}}\,dx = a|c|\sqrt{2\pi}. \qquad \textbf{(g.15)}$$

El valor de la integral es 1 si y solo si $a = \frac{1}{c\sqrt{2\pi}}$, en cuyo caso la función

gaussiana es la funcion de densidad de una variable aleatoria con distribución normal de media $\mu=b$ y varianza $\sigma^2=c^2$. Se muestran varias gráficas de funciones gaussianas en la imagen adjunta.

Aplicaciones

La primitiva de una función gaussiana es lafuncion error.

Estas funciones aparecen en numerosos contextos de las ciencias naturales, ciencias sociales, matematicas e ingeniería. Algunos ejemplos:

En estadística y teoría de probabilidades, las funciones gaussianas aparecen como la función de densidad de la distribución normal, la cual es una distribución de probabilidad límite de sumas complicadas, según el teorema del limite central.

Una función gaussiana es la función de onda del estado fundamental del oscilador armonico cuantico.

Los orbitales moleculares usados en química computacional son combinaciones lineales de funciones gaussianas llamados orbitales gaussianos.

Matemáticamente, la función gaussiana juega un papel importante en la definición de los polinomios de Hermite.

Consecuentemente, están también asociadas con el estado de vacio en la teoría cuántica de campos.

Los rayos gaussianos se usan en sistemas ópticos y de microondas.

Las funciones gaussianas se utilizan como filtro de suavizado en el procesamiento digital de imágenes.

BIBLIOGRAFIA

Ahmed El Zooghby, “Smart Antenna Engineering”, Artech House, London 2005.

Barry D. Van Veen and Kevin M. Buckley. Beamforming: A Versatile Approach to Spatial Filtering, IEEE ASSP Magazine, April 1998.

IEEE. V encuentro nacional de ramas , universidad de Carabobo. noviembre del 2004. **http://www.lant.ing.uc.edu.ve/**.

Tapan K Sarkas Michaele Wicks, Magdalena Salazar, Palma and Robert J. “ SMART ANTENNAS“, 2007

Alexander D. Poularikas and Zayed M. Ramadan “Adaptative Filtering Primer With Matlab“, Taylor y Francis Group, A CRC PRESS BOOK, 2006

Simon Haykin, Adaptative Filter Theory, Prentice Hall, Upper Saddler River, four edition, 2002.

SMART ANTENNAS FOR WIRELESS COMUNICATIONS, Liberty, Joseph C,and Rappaport, Theodore S. Prentice Hall PTR, Estados Unidos de America 1999.,

Andrea Goldsmith, WIRELESS COMMUNICATION, Sanford University. 2005

AGILENT TECHNOLOGIES, Optimizing Your CDMA Wíreless Network Today and Tomorrow Using Drive-Test Solutions Aplication Note – 1345 2000, **http://cp.literature.agilent..com/litweb/pdf/5968-9916E.pff**

w3.iec.csic.es/ursi/articulos_villaviciosaodon_2001/articulos/122.pdf Sitio Web de tutoriales Estudio Preliminar de Prestaciones de Antenas Inteligentes para UMTS

http://es.wikipedia.org, Enciclopedia en español Wikipedia 2010

Sitio Web de tutoriales La investigacion de Intel sobre antennas inteligentes aumenta el alcance y ancho de banda de las WLAN **www.intel.com/espanol/technolpgy/magazine/archive/2004/sep/wi09041.pdf**

w3.iec.csic.es/ursi/articulos_villaviciosaodon_2001/articulos/278.pdf Sitio Web de tutoriales Aplicabilidad y Prestaciones de Antenas Adaptativas en UMTS

Sitio Web de tutoriales Antenas inteligentes con aplicaciones en SDMA cita2003.fing.edu.uy/articulosyf/17.pdf

a.cicese.mx/posgrado/3trim/telecom/antenas.htm, - 12k, Sitio Web de tutoriales Antenas Inteligentes para comunicaciones moviles y electronica

Theodore, S Rappaport. “Smart Antenas for Gíreles Communications: IS-95 and third Generation CDMA Applications”, Prentice Hall, New Jersey, 1999.

A.N. Mucciardi. ´A new class of search algorithms for adaptive computation´ Proc. 1973 IEEE conference on Decision and Control, Dec. San Diego, Paper No. WA 5-3,pp 94-100

ZENG,H.H; TONG, L. Relationships Between the Constant Modulus and Wiener receivers. IEEE Transactions on Information Theory, (IT-44), New Jersey, p. 1523-1537, 1998.

DIEGO SEPULVEDA J. Introduccion a Matlab y Simulink . Chile p 23-30 agosto de 2002.

The Mathworks, \Getting Started with Matlab"

Jarabo Amores Maria Pilar, Técnicas de optimización de ingeniería, el algoritmo LMS, p 2-8 Mayo de 2007

Jeronimo Arenas Garcia, Manel Martinez Ramon y Anibal R. Figuiras Vidal, Adaptacion de adaptaciones: una nueva aproximación para el tratamiento digital en comunicaciones, p 6-8. 2004

Julio C Mansilla Hernández, Julio N. García Silverio, Francisco J. Arteaga Universidad de Carabobo, Escuela de Ingeniería Eléctrica, Dpto. de Sistemas y Automática. Control de Velocidad de un motor DC, usando filtros de kalman en tiempo continuo.p 1-5 . 2002

Miguel Angel Lagunas. Estimaciones MSE (Minimo error cuadratico) y filtro de wiener p 6-8, p 10-14 Agosto 2003

Tamer abdelazim Mellik, Ph.D. demo Algorithm lms, demo Algorithm rls. Central matlab octubre 2003

Conformador de haz de referencia por código de dos etapas, cap 3, p 71. 2003

Dr. Luis M. San José Revuelta, Procesado Adaptativo de señales Universidad de Carabobo p 4-9, Mayo 2008

Departamento de Ingeniería de comunicaciones DICOM, Universidad

de Cantabria, Igualación, e Identificación ciega, Tratamiento avanzado de señal en telecomunicaciones. Grupo de tratamiento avanzado de señal GTAS, , p 4-11, p13-16. 2007

Blight, J.D., Dailey, R.L. and Gangass, D., "Practical control law design for aircraft using multivariable techniques," International Journal of Control, Vol. 59, No. 1, 93-137. (1994)

Kalman, R.E. and Bucy, R, S, *New results in linear filtering and prediction theory,* ASME – J. Basic Eng. Ser. D, 83, 95-108, 1961

Athans, Michael, *Kalman Filtering,* The Control Handbook, 1996

Kailath, Thomas, *A view of three decades of linear filtering theory,* IEEE Transactions on Information Theory, March 1974

http://www.avaxhome.ru, 2009