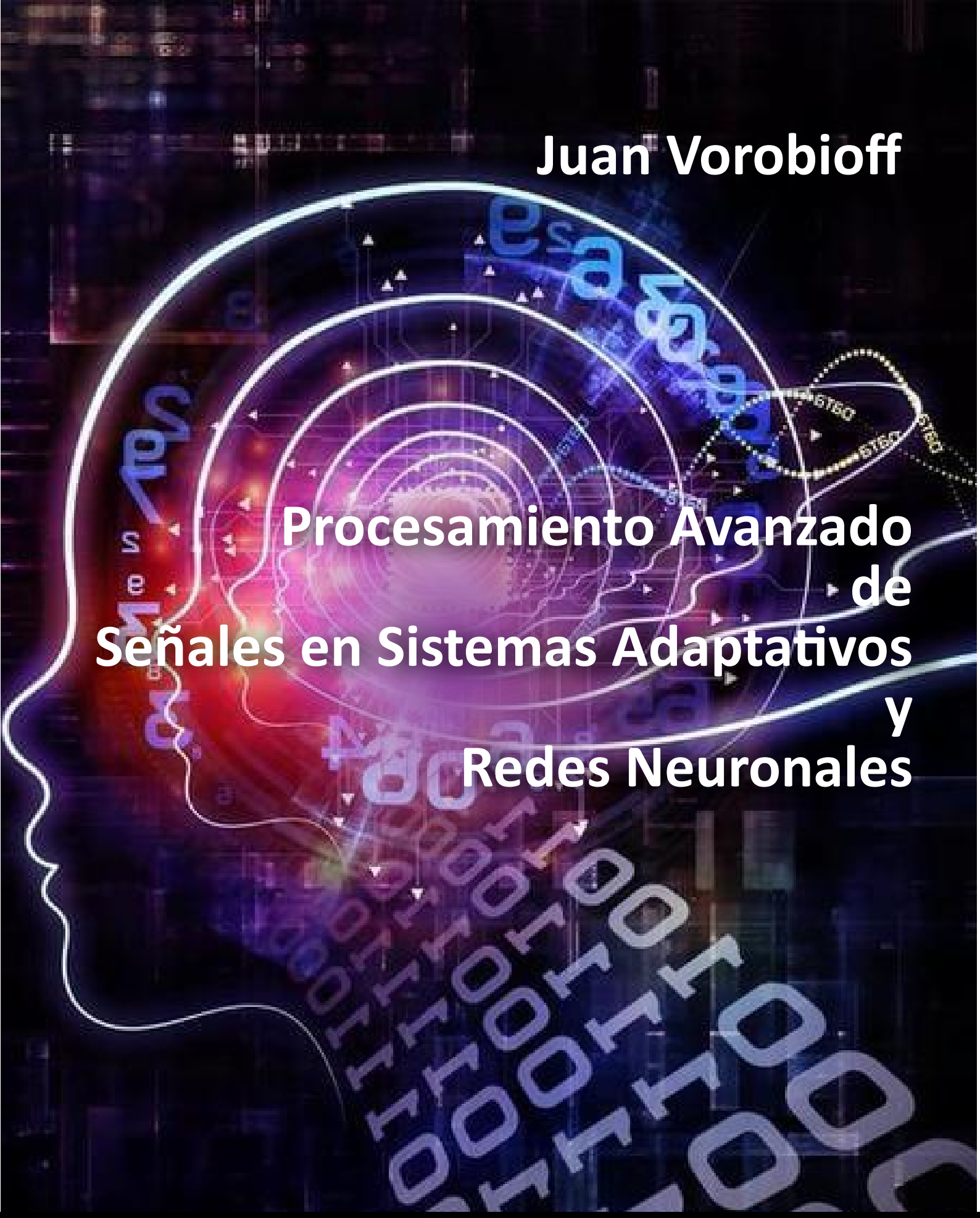


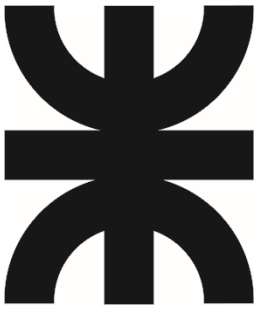
Juan Vorobioff

Procesamiento Avanzado de Señales en Sistemas Adaptativos y Redes Neuronales

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

CiN REUN
Red de Editoriales
de Universidades Nacionales
de la Argentina

edUTecNe



Procesamiento Avanzado de Señales en Sistemas Adaptativos y Redes Neuronales

Juan Vorobioff

Universidad Tecnológica Nacional

Facultad Regional Buenos Aires

2021

Vorobioff, Juan

Procesamiento avanzado de señales en sistemas adaptativos y redes neuronales / Juan Vorobioff ; editado por Fernando Cejas. - 1a ed. - Ciudad Autónoma de Buenos Aires: edUTecNe, 2021.

Libro digital, PDF

Archivo Digital: descarga y online

ISBN 978-987-4998-66-8

1. Ingeniería Electrónica. 2. Matemática para Ingenieros. I. Cejas, Fernando, ed. II. Título.

CDD 621.38

Edición y Diseño: Fredy

Universidad Tecnológica Nacional – República Argentina
Rector: Ing. Héctor Eduardo **Aiassa**
Vicerrector: Ing. Haroldo **Avetta**
Secretaria Académica: Ing. Liliana Raquel **Cuenca Pletsch**

Universidad Tecnológica Nacional – Facultad Regional Buenos Aires
Decano: Ing. Guillermo **Oliveto**
Vicedecano: Ing. Andrés **Bursztyn**

edUTecNe – Editorial de la Universidad Tecnológica Nacional
Coordinador general a cargo: Fernando H. **Cejas**
Director Colección Energías Renovables, Uso Racional de Energía, Ambiente: Dr. Jaime **Moragues.**

Queda hecho el depósito que marca la Ley Nº 11.723

Sarmiento 440, Piso 6 (C1041AAJ)
Buenos Aires, República Argentina
Publicado Argentina – Published in Argentina

ISBN 978-987-4998-66-8

Dr. Ing. J. Vorobioff

Dedicatoria

A mi esposa, mis hermanos, mis ahijados y mis sobrinos

Agradecimientos

Facultad Regional Buenos Aires, Universidad Tecnológica Nacional (FRBA – UTN)

Comisión Nacional de Energía Atómica (CNEA)

Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET)

Contenidos

Prólogo

Estimado lector:

Este libro es un resumen de mi experiencia como profesor y Doctor en Ingeniería en área del procesamiento de señales e imágenes, y tiene como objetivo introducir temas relacionados con el procesamiento avanzado de señales. En mis trabajos de Investigación utilicé el procesamiento avanzado de señales y reconocimiento de patrones en Olfatometría Electrónica, estudios de Espectrometría por Ablación Láser (LIBS), Espectrometría de Movilidad Iónica y en diferentes proyectos de Ingeniería.

El procesamiento de señales e imágenes involucra diferentes áreas de las ciencias y de la ingeniería. Se puede utilizar en química, física, economía, automatización, ciencias de la tierra, ciencias sociales, biología, medicina, etc. En los últimos años, aumento notablemente el procesamiento avanzado de datos en muchas áreas de la vida cotidiana. Esto se observa en aplicaciones de teléfonos celulares, en buscadores de internet utilizando inteligencia artificial, en reconocimiento de rostros mediante el método de aprendizaje profundo y en reconocimiento de voz. En este sentido, las redes sociales como Twitter y Facebook emplean en forma intensiva este tipo de procesamiento para estudiar el comportamiento de los usuarios. Para un mismo problema se pueden plantear diferentes métodos de análisis de datos, por lo que resulta indispensable reconocer métodos óptimos y descartar aquellos que no sirvan. Es necesario un enfoque moderno y multidisciplinar para procesar datos en forma correcta y obtener resultados satisfactorios.

Es importante destacar que el procesamiento avanzado de señales resuelve diferentes problemas de ingeniería y estudios científicos. Esto puede observarse en innumerables publicaciones de artículos de varias disciplinas que proponen diferentes tipos de aplicaciones. Estas aplicaciones se pueden clasificar en identificación de sistemas, eliminación de ruidos e interferencias y predicción.

Así también, si el entorno cambia los sistemas de procesamiento de datos deben ser dinámicos y modificar su funcionamiento rápidamente para adaptarse al entorno. Se utilizan Sistemas Adaptativos que presentan muchas ventajas respecto de un sistema de respuesta fija con parámetros constantes. Muchas veces resulta indispensable utilizar un sistema adaptativo, ya que si se plantea un sistema de filtrado digital con coeficientes fijos puede fracasar el análisis. Un sistema adaptativo puede modificar su comportamiento, es decir, puede ajustar sus parámetros internos mediante un algoritmo adaptativo. Estos parámetros se ajustan mediante una regla de aprendizaje en cada iteración del algoritmo.

Los sistemas adaptativos se encuentran en todos los ámbitos de la ingeniería en la actualidad. Desde el control de procesos industriales, pasando por diversos ámbitos de las comunicaciones y todo tipo de artefactos vinculados a la industria de componentes (para la industria del entretenimiento, automotriz, aeroespacial, mecánica, naval, artefactos médicos, etc.). En consecuencia, se hace necesario disponer de herramientas adecuadas para su análisis y desarrollo.

El presente libro corresponde al análisis de Sistemas Adaptativos. En el primer capítulo de repaso se estudian conceptos de procesamiento digital de señales. En el capítulo I se realiza una introducción a sistemas adaptativos con filtros lineales y no lineales. También se analizan las 4 aplicaciones básicas de sistemas adaptativos. En el capítulo II se estudian procesos estacionarios y modelos estocásticos. El capítulo III corresponde al análisis espectral donde se realiza estimación del espectro y se presentan distintas aplicaciones. En el capítulo IV se estudian algoritmos de filtros lineales adaptativos, presentando ventajas y desventajas de distintos métodos. Se muestran distintas aplicaciones con ejemplos de programas. Por último, en el capítulo V se realiza una breve introducción a redes neuronales

Se analizan diferentes sistemas en el dominio temporal y en dominio de la frecuencia. Se abarcan los temas mediante un marco teórico, con desarrollos matemáticos, ejercicios analíticos, ejemplos de aplicaciones variadas y ejercicios en Matlab® y Python.

Se espera que el contenido del libro sea de utilidad al lector en el desarrollo de sus aplicaciones de ingeniería científico tecnológicas.

Para citar el presente libro por favor incluir datos completos con número de ISBN.

Dr. Ing. Juan Vorobioff

Repaso de Procesamiento de Señales

Señales

La señal es un conjunto de datos que se puede utilizar para transmitir información o mensajes. Por ejemplo, voz, tensión en un circuito, presión, flujo de bits, imágenes, etc.

Clasificación de Señales

Par o Impar

Tiempo continuo o Tiempo discreto

Analógicas o Digitales

Determinísticas o Estocásticas

Unidimensionales o Multidimensionales

Duración Finita o Infinita

Periódicas, Aperiódicas, Periódica en un tramo

Acotadas o No Acotadas

Señal de Tiempo Continuo y Señal de Tiempo Discreto (ver Fig. 1)

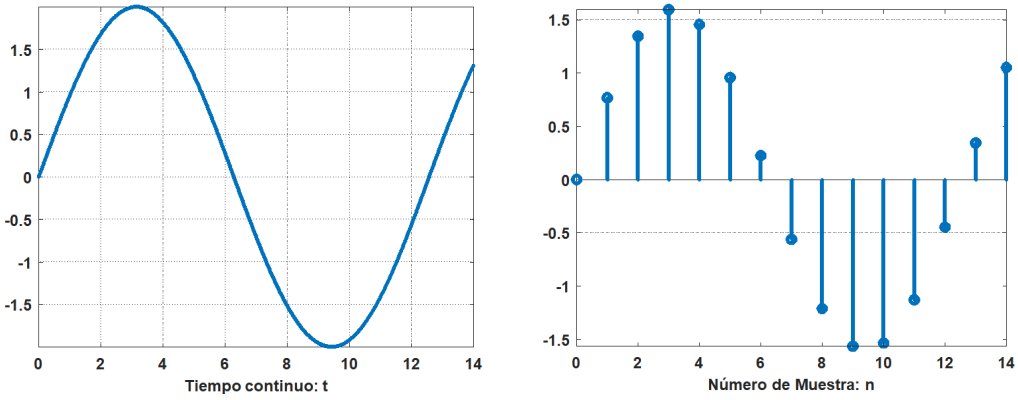


Fig. 1: Señal de tiempo continuo (izq.) y señal de tiempo discreto (derecha)

Tiempo continuo: $t \rightarrow f(t)$

Tiempo discreto: $n \rightarrow f[n]$

En la señal de tiempo Discreto: n, la señal no está definida entre una muestra y otra.

Señal Analógica: su magnitud presenta valores continuos

Señal Digital: es una señal de tiempo discreto, su magnitud esta codificada en dígitos binarios u otra forma discreta. Los valores de sus magnitudes siempre son discretos.

Señal determinística: contiene valores fijos que se pueden expresar mediante una fórmula matemática, una tabla o un método.

Señales Estocásticas: es una señal que toma valores aleatorios y sólo se pueden caracterizar estadísticamente

Señal Par: $f(t) = f(-t)$

Señal Impar: $f(t) = -f(-t)$

Señal Periódica Continua	**Señal Periódica Discreta**
$x(t) = x(t \pm k.T)$, $\forall t$, T: Período	$x[n] = x[n \pm N_0]$, $\forall n$
Caso contrario: Señal Aperiódica	Caso contrario: Señal Aperiódica
$T \in \mathcal{R}eales$	$\boldsymbol{N_0} \in Enteros$
$f_0 = \frac{1}{T_0}$; $w_0 = \frac{2.\pi}{T_0} = 2.\pi.f_0$	$F_0 = \frac{k}{N_0} = \frac{f_0}{f_s}$; $\Omega_0 = \frac{2.\pi.k}{N_0}$
	$\boxed{N_0 = \frac{2.\pi.k}{\Omega_0}}$

Señales de Energía y de Potencia

Las señales de Energía (E) poseen amplitud finita, además convergen a cero para $t \rightarrow \infty$ o tienen duración finita.

$\boxed{E = \int_{-\infty}^{\infty} |f(t)|^2 . dt}$ Tiempo Continuo

$\boxed{E = \sum_{n=-\infty}^{\infty} |f[n]|^2}$ Tiempo Discreto

$\boxed{0 < E < \infty \iff Señal\ de\ Energía}$ $\Rightarrow$ $P = 0$

Las señales de potencia (P) tienen energía infinita. El caso más común es el de las señales periódicas. Se evalúa la energía promedio por período (E/T).

$\boxed{P = \frac{1}{T} . \int_{\langle T \rangle} |f(t)|^2 . dt}$ Tiempo Continuo

$\boxed{P = \frac{1}{N_0} \sum_{\langle N_0 \rangle} |f[n]|^2}$ Tiempo Discreto

$\boxed{0 < P < \infty \iff Señal\ de\ Potencia}$ $\Rightarrow$ $E = \infty$

Valor Eficaz:

$$\boxed{V_{ef} = V_{RMS} = \sqrt{P}}$$

Si la señal no resulta periódica pero es de energía infinita, la potencia puede calcularse considerando con $T \to \infty$:

$$\boxed{P = \lim_{T\to\infty} \frac{1}{T} . \int_{\langle T \rangle} |f(t)|^2 . dt}$$

Si bien las señales de potencia y energía son excluyentes, es posible que una señal no sea ni de potencia ni de energía, dado que ambos valores resultan infinitos

Funciones Básicas de Tiempo Continuo

En la Fig. 2 se muestran funciones básicas para el armado de señales.

Función Delta de Dirac (Impulso Unitario Continuo): $\delta(t)$

Función Escalón: $u(t)$

Función Rampa: $\rho(t)$

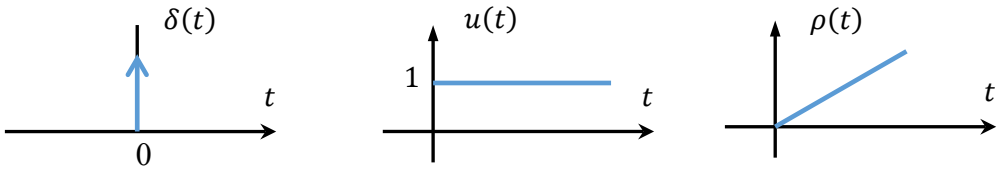


Fig. 2 - Funciones básicas de tiempo continuo

Delta de Dirac	**Escalón:**	**Rampa:**	**Signo:**
$\boldsymbol{\delta(t)} = \begin{cases} \mathbf{0} & \boldsymbol{t \neq 0} \\ \infty & \boldsymbol{t = 0} \end{cases}$ $\int_{-\infty}^{\infty} \delta(t).dt = 1$	$\boldsymbol{u(t)} = \begin{cases} \mathbf{1} & \boldsymbol{t > 0} \\ \mathbf{0} & \boldsymbol{t < 0} \end{cases}$	$\boldsymbol{\rho(t) = t.u(t)} = \begin{cases} \boldsymbol{t} & \boldsymbol{t \geq 0} \\ \mathbf{0} & \boldsymbol{t < 0} \end{cases}$	$sgn(t) = 2.u(t) - 1$
$f(t) * \delta(t - t_0) = f(t - t_0)$ $f(t).\delta(t - t_0) = f(t_0).\delta(t - t_o)$ Si $t_0 = 0$: $f(t).\delta(t) = f(0).\delta(t)$	$\frac{du(t)}{dt} = \delta(t)$ $\int_{-\infty}^{t} \delta(t).dt = u(t)$	$\frac{d\rho(t)}{dt} = u(t)$	$\delta(a.t) = \frac{1}{\lvert a \rvert}.\delta(t)$

En la Fig. 3 se muestran las Funciones Básicas de Tiempo Discreto

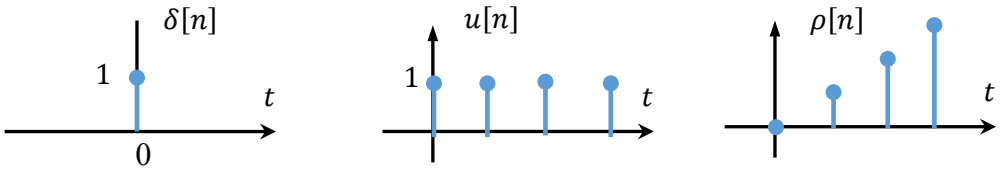


Fig. 3 - Funciones Básicas de Tiempo Discreto

Impulso Unitario Discreto	Escalón	Rampa
Delta de Kronecker $\delta[n] = \begin{cases} 0 & n \neq 0 \\ 1 & n = 0 \end{cases}$	$u[n] = \begin{cases} 1 & n \geq 0 \\ 0 & n < 0 \end{cases}$	$\rho[n] = n.u[n] = \begin{cases} n & n \geq 0 \\ 0 & n < 0 \end{cases}$

Convolución:

$$y(t) = x_1(t) * x_2(t) = \int_{-\infty}^{\infty} x_1(\tau).x_2(t-\tau).d\tau$$

Para Sistemas LTI:

$$y(t) = x(t) * h(t) = \int_{-\infty}^{\infty} x(\tau).h(t-\tau).d\tau \quad \text{(tiempo continuo)}$$

$$y[n] = x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k].h[n-k] \quad \text{(tiempo discreto)}$$

Propiedades de convolución: distributiva, conmutativa y asociativa

Funciones Básicas en Matlab®

Función Escalón: $u(t)$

```
% Archivo: U.m,  Función escalón u(t) o u[n]
function u=U(x)
u=(x>=0);
end
```

Función Rampa: $\rho(t)$

```
% Archivo R.m,   Función Rampa
function r=R(x)
r=x.*(x>0);
end
```

Ejercicios en MATLAB®

Ejercicio I

Formación de señales básicas

Sea la siguiente señal continua: $x_1(t) = \rho(t-1) - \rho(t-2) - u(t-5)$

a) Graficar la función $x_1(t)$ y su derivada
b) Graficar $x_2 = x_1(t-2)$ y $x_3 = x_1(-2*t+1)$

Resolución

```
%% Ejemplo uso de rampa y escalón. Graficar Función Original y Derivada
t= -4:  0.01 : 8 ;
x1= R(t-1) - R(t-2) - U(t-5) ;
%Derivada de función Original
dx = t(2)-t(1) ;
x2 = diff(x1) / dx ; % x2 es la derivada de x1
figure ;  subplot(211); plot(t,x1, 'Linewidth', 3);
grid on; title('Función Original')
subplot(212); plot(t(2:end), x2, 'Linewidth', 3);
ylim([-2 2]) ;  grid on; title('Función Derivada')
%% Ejemplo de desplazamiento e inversión de señales
% Calculamos y graficamos: x2 = x1(t-2) y x3 = x1( -2*t+1)
tx= t-2 ;    x2=R(tx-1)-  R(tx-2)-  U(tx-5) ;
tx= -2*t+1;   x3= R(tx-1)-  R(tx-2) - U(tx-5) ;
figure ;  subplot(211); plot(t,x2, 'Linewidth', 3);
grid on; title('Función Desplazada 2')
subplot(212); plot(t,x3, 'Linewidth', 3);
grid on; title('Funcion x3')
% Otra forma
t= -4:  0.01 : 8 ;
f=@(t) R(t-1) - R(t-2) - U(t-5) ;
figure; subplot(311) ;
plot(t,f(t), 'linewidth', 4) ;grid on; title('Señal Original f(t)')
subplot(312)
plot(t,f(t-2), 'linewidth', 4) ;grid on; title('f1')
subplot(313)
plot(t,f(-2*t+1), 'linewidth', 4) ;grid on; title('f2')
```

En la Fig. 4 se muestran los resultados del programa.

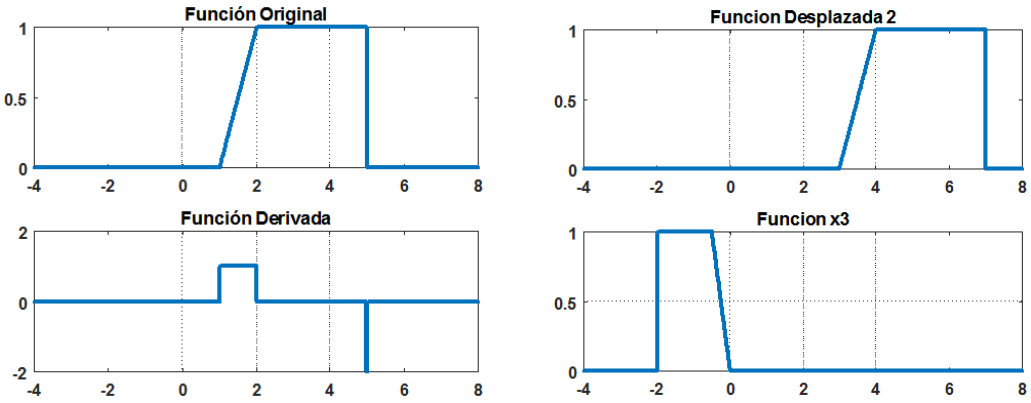


Fig. 4 – Ejemplo de armado de funciones

Ejercicio II

Señales básicas de tiempo continuo y de tiempo discreto

Sea $f(t) = \rho(t) - \rho(t-10) - 10.u(t-10)$. Se pide graficar en Matlab® u Octave:

a) Señal de tiempo continuo: $f(t)$

b) Señal de tiempo discreto $f[n]$

Resolución

```
%% Función de tiempo continuo:
dt= 0.01;    t= -5: dt: 15;
f=R(t)-R(t-10)- 10*U(t-10);
figure;  subplot(211); plot(t,f);
grid on;  title('Función de Tiempo Continuo')
%% Función de tiempo Discreto:
n= -5: 15 ;
f_discreta = R(n)-R(n-10)- 10*U(n-10);
subplot(212);  stem(n, f_discreta);
grid on; title('Funcion Discreta')
```

En la Fig. 5 se muestran los resultados.

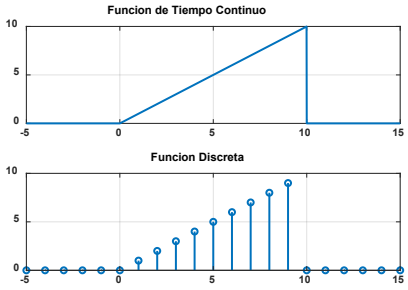


Fig. 5 – Funciones de tiempo continuo y tiempo discreto

Ejercicio III

Cálculo de Energía

Dadas las siguientes señales: $y_1(t) = e^{-2.t}u(t)$; $y_2[n] = 3.\left(\frac{1}{2}\right)^n u[n]$

a) Graficar la función usando *plot* para señal continua y *stem* para señal discreta
b) Calcular en forma numérica la Energía de cada señal

Resolución

```
%% Ejemplo de Señales de tiempo continuo t y de tiempo discreto n
dt= 0.001 ;
t= -5: dt: 5 ;
n= -5:5 ;
y1 = exp(-2*t) .*U(t) ;
y2 = 3* ( (1/2).^n) .* U (n) ;
figure ; subplot(211); plot(t,y1, 'Linewidth', 3) ;
title('Función de tiempo continuo t'); grid on
subplot(212); stem(n,y2, 'Linewidth', 3) ;
title('Función de tiempo discreto n'); grid on
%% Calculo de Energías
Energia_cont = sum(abs(y1).^2) *dt
Energia_disc = sum(abs(y2).^2)
```

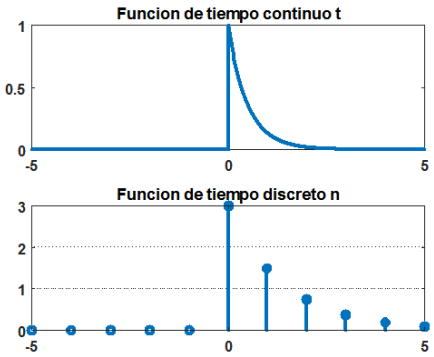


Fig. 6 – Función exponencial continua y discreta

Ejercicio IV

Micrófono y parlantes

_Mediante el micrófono de la PC, grabar 3 segundos de voz y reproducirlo en el parlante

Resolución en Matlab®

```
%% Prueba de Micrófono y parlantes
dt=1/8000;
```

```
t=0:dt:3-dt;
fs=1/dt;
% genero objeto para grabar audio con fs, nbits=16 y 1 canal
r=audiorecorder(fs,16,1);
% grabo desde micrófono 3 segundos / Función vieja: record(r,3);
recordblocking(r,3);
y=getaudiodata(r);
plot(t,y);
audiowrite('Sonido.wav',y,fs);
[y,fs]=audioread('Sonido.wav');
sound(y,fs);
pause(2)
sound(y,fs *2);
```

Ejercicio V

Generación de tono puro y reproducción en parlante

Tono Puro. Reproducir un tono puro de 560 Hz. Probar con distintas frecuencias

Resolución en Matlab®:

```
%% Ejercicio para reproducir sonido. Tono puro
%Generación de un tono correspondiente a un seno de 560 Hz, 5 seg.
%fs>=2*fmax
fmax=560; % fmax la única frecuencia dada.
fs=10*fmax; %Mientras más grande mejor
Ts= 1/fs %Periodo de muestreo
t= 0:Ts:5 ; % Duración del tono: 5 segundos
%Determinación del vector de amplitudes
y1 = sin(2*pi*560*t) % Vectores y,t
plot(t, y1) %Grafico discreto
% Reproducimos la señal
sound(y1, fs)
%player1 = audioplayer(y1,fs);  sound = (y1,fs);  wavplay = (y1,fs)
```

Ejercicio VI

Programa Puerto Serie Matlab®

```
puerto_serie =serial('COM10','BaudRate',9600);
fopen(puerto_serie) % abre el objeto s
pause(2);  dato = 'a' ;
fwrite(puerto_serie, dato) % envía dato(binario) por el puerto
% fprintf(s,'dato') envía string % x(i)= fread(puerto_serie,1,'uint8')
while(puerto_serie.BytesAvailable == 0)
end
disp('Recibiendo...')
leo = fscanf(puerto_serie)
tic
for i=1:10 ;
   while(puerto_serie.BytesAvailable ~= 0)
      fwrite(puerto_serie, dato) ;
      leo = fscanf(puerto_serie)
      i = i+1 ; dato =dato+1;
   end
end
toc
fclose( puerto_serie ); delete( puerto_serie )
```

Convolución

La convolución es un operador matemático con 2 funciones, generando una tercera función

$$\boxed{y(t) = x_1(t) * x_2(t) = \int_{-\infty}^{\infty} x_1(\tau).x_2(t-\tau).d\tau}$$ Tiempo Continuo

$$y[n] = x[n] * g[n] = \sum_{k=-\infty}^{\infty} x[k].g[n-k]$$ Tiempo Discreto

Para Sistemas LTI (Lineales e Invariantes en el tiempo):

Convolución tiempo Continuo: $\boxed{y(t) = x(t) * h(t) = \int_{-\infty}^{\infty} x(\tau).h(t-\tau).d\tau}$

Convolución tiempo Discreto: $\boxed{y[n] = x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k].h[n-k]}$

Propiedades de convolución: distributiva, conmutativa y asociativa

$$f(t) * \delta(t-t_0) = f(t-t_0)$$
$$f(t).\delta(t-t_0) = f(t_0).\delta(t-t_o)$$
$$Si\ t_0 = 0:\ f(t).\delta(t) = f(0).\delta(t)$$

Conexión de Sistemas LTI

En las Fig. 7 y Fig. 8 se muestran las respuestas impulsionales $h(t)$ para conexión serie y conexión paralelo de sistemas.

Conexión Serie:

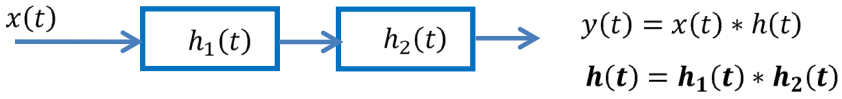


Fig. 7 – Conexión serie de sistemas

Conexión Paralelo:

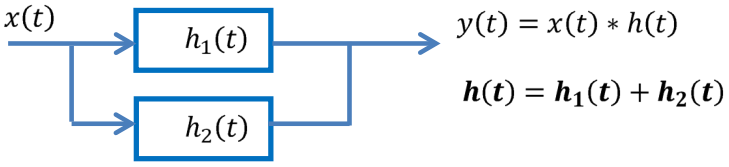


Fig. 8 – Conexión paralelo de sistemas

Ejercicio VII

Convolución de señales continuas

Obtener en Matlab® y Analíticamente la salida del sistema descripto por $h(t)$ y entrada $x(t)$.

$$h(t) = 2.e^{-2.t}.u(t) \quad ; x(t) = u(t) - u(t-2)$$

Resolución Analítica: $h(t) = 2.e^{-2.t}.u(t) \quad ; x(t) = u(t) - u(t-2)$ (ver Fig. 9)

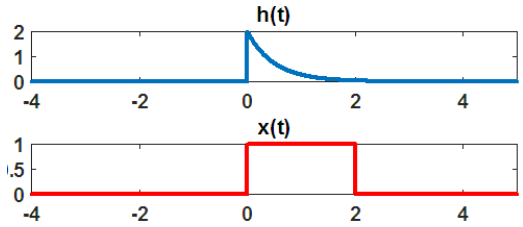


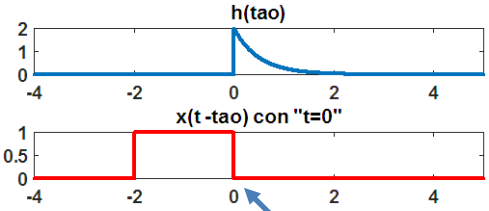
Fig. 9 – Ejemplo de convolución de señales

$$y(t) = x(t) * h(t) = \int_{-\infty}^{\infty} x(\tau).h(t-\tau).d\tau$$

Usamos: $y(t) = h(t) * x(t) = \int_{-\infty}^{\infty} h(\tau).x(t-\tau).d\tau$ (dejamos fija la señal $h(\tau)$)

Cambiamos variable y dejamos fija la señal más complicada: $h(\tau)$

Cambiamos variable e Invertimos la señal más simple: $x(-\tau)$



$x(t-\tau)|_{t=0} = x(-\tau)$

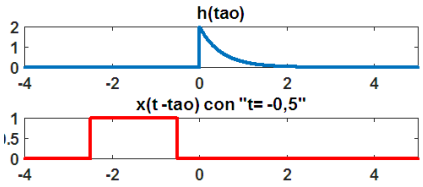
t Este punto en el orígen le asignamos t

Luego variamos t y la señal se desplaza:

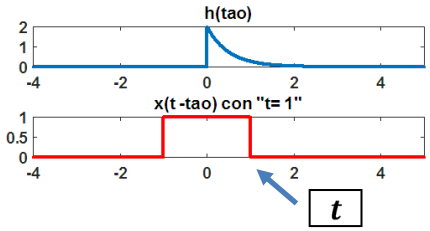
$x(t-\tau)|_{t=variable}$ Para t positivos la señal se desplaza a la derecha

$x(t-\tau)|_{t=variable}$ Para t negativo la señal se desplaza a la izquierda

$\boldsymbol{t<0}$ $\boldsymbol{y(t)=0}$ $y(t)=\int_{-\infty}^{\infty} h(\tau).x(t-\tau).d\tau = 0$



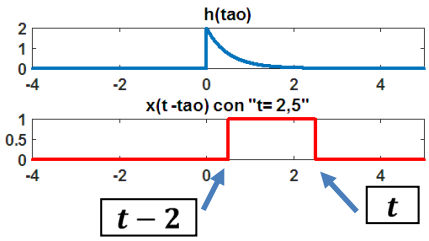
$\boldsymbol{0<t<2}$



Siendo: $h(t)=2.e^{-2.t}.u(t)$; $x(t)=u(t)-u(t-2)$

$y(t)=\int_{-\infty}^{\infty} h(\tau).x(t-\tau).d\tau = \int_{0}^{t} 2.e^{-2.\tau}.u(\tau).1.d\tau = 2.\left[\frac{1}{-2}e^{-2.\tau}\right]_{0}^{t} = -e^{-2.t}+1 = 1-e^{-2.t}$

$\boldsymbol{t>2}$



Siendo: $h(t)=2.e^{-2.t}.u(t)$; $x(t)=u(t)-u(t-2)$

$y(t)=\int_{-\infty}^{\infty} h(\tau).x(t-\tau).d\tau$

$y(t)=\int_{t-2}^{t} 2.e^{-2.\tau}.u(\tau).1.d\tau = 2.\left[\frac{1}{-2}e^{-2.\tau}\right]_{t-2}^{t} = -e^{-2.t}+e^{-2.(t-2)} = e^{-2.t}.(e^{+4}-1)$

$$y(t)=\begin{cases} 0 & t<0 \\ 1-e^{-2.t} & 0<t<2 \\ (e^{+4}-1).e^{-2.t} & t>2 \end{cases}$$

Resolución Numérica con Matlab®

```
% Usamos mismo tiempo en ambas señales, aunque podría ser distinto
dt = 0.01 ; t= -1: dt: 5 ; % largo L
x= U(t)-U(t-2);
h= 2* exp(-2*t) .* U(t);
y=conv(x,h)* dt; % largo resultante: L+L-1
subplot(311); plot(t,x) ; grid on;
subplot(312); plot(t,h) ; grid on;
% Convolución inicia en: suma de inicios, finaliza en: suma de finales
tc = (-1-1) : dt : (5+5) ; % Nuevo vector temporal más largo
subplot(313), plot(tc,y) ;  grid on; xlim([-1 5])
% Opcional gráfico analítico
hold on;  ya = (1-exp(-2*t)).*(U(t)-U(t-2)) + (exp(4)-1)*exp(-2*t).*U(t-2) ;
plot(t,ya)
```

Se muestran los resultados en la Fig. 10.

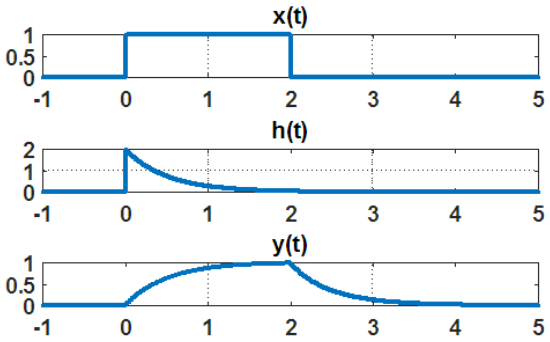


Fig. 10 – Resultado de convolución de señales

Ejercicio VIII

Autoconvolución y Autocorrelación

Dadas las señales: $x_1(t) = u(t) - u(t-2)$

a) Calcular en forma numérica su autoconvolución, es decir: $y(t) = x_1(t) * x_1(t)$

b) Calcular en forma numérica su autocorrelación, es decir: $y(tao) = x_1(tao) * x_1(-tao)$

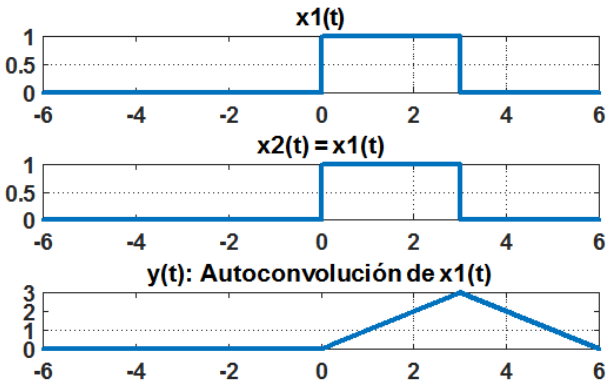
Nota: para calcular autocorrelación se invierte temporalmente la segunda señal

Resolución Numérica con Matlab®

```
% Calculamos Autoconvolución de x1
   dt= 0.001;   t= -6 :dt: 6 ;
   x1=U(t)-U(t-3) ;
   x2=x1;  %x2= ídem x1
```

```
   y=conv(x1,x2)*dt ;
% Convolución inicia en: suma de inicios, finaliza en: suma de finales
   tc = (-6-6) :dt: (6+6) ; % Nuevo vector temporal más largo
   subplot(311); plot(t,x1); grid on; title('x1(t)')
   subplot(312); plot(t,x2); grid on; title('x2(t) = x1(t)')
   subplot(313); plot(tc,y); grid on;
   title('y(t):  Autoconvolución de x1(t)');  xlim([-6 6])
%% Calculamos Autocorrelación de x1(t) , usamos conv
   x2 = U(-t)-U(-t-3) ; % x1 invertida
   y=conv(x1, x2)*dt ;
% Convolución inicia en: suma de inicios, finaliza en: suma de finales
   tc = (-6-6) :dt: (6+6) ; % Nuevo vector temporal más largo
   figure; subplot(411); plot(t,x1); grid on; title('x1(t)')
   subplot(412); plot(t,x2); grid on; title('x1 invertida(t)')
   subplot(413); plot(tc,y); grid on;
   title('y(t):  Autocorrelación de x1(t) - Usamos convolución');  xlim([-6 6])
%% Calculamos Autocorrelación de x1(t), usamos xcorr
  [Rxx, tau] = xcorr(x1) ;
   subplot(414);  plot(tau*dt , Rxx*dt );  grid on;
   title('y(t):  Autocorrelación de x1(t) - Usamos xcorr');  xlim([-6 6])
```

En la Fig. 11 se muestran los resultados.



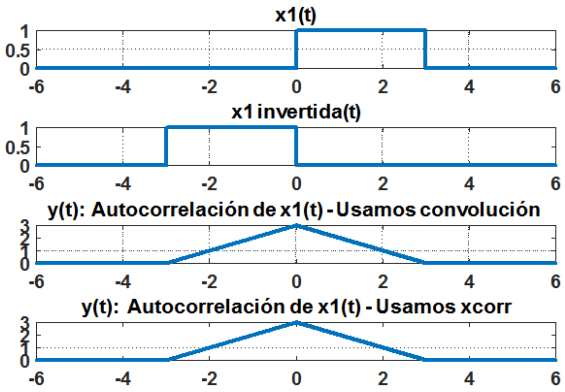


Fig. 11 – Resultado de autoconvolución y autocorrelación de señales

Ejercicio IX

Convolución de Señales Discretas

Sean la siguiente señal discreta, calcular la convolución y graficar:

$x_1[n] = u[n] - u[n-3]$

Calcular: $y[n] = x_1[n] * x_1[n]$

Resolución con Matlab® / Octave

```
% Convolución de Señales Discretas
n= -2 : 6 ;   %Vector discreto. Se utiliza dt=1
%Señales a convolucionar
x1=U(n)-U(n-3) ;
x2= x1 ; % dt=1
%Algoritmo de convolución: Utilizamos conv(v1,v2) provista por Matlab®
y=conv(x1,x2) ;
% Convolución inicia en: suma de inicios, finaliza en: suma de finales
nc = (-2-2) : (6+6) ; % Nuevo vector temporal más largo
subplot(311); stem(n,x1); grid on; title('x1[n]')
subplot(312); stem(n,x2); grid on; title('x2[n] =x1[n]')
subplot(313); stem(nc,y); grid on; title('y[n]'); xlim([-2 6])
```

En la Fig. 12 se muestran los resultados.

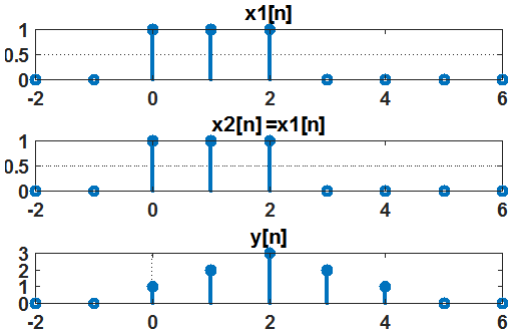


Fig. 12 – Resultados de la Convolución de Señales Discretas

Resolución Analítica

$x_1[n] = \delta[n] + \delta[n-1] + \delta[n-2]$

$y[n] = x_1[n] * x_1[n]$

$y[n] = (\delta[n] + \delta[n-1] + \delta[n-2]) * (\delta[n] + \delta[n-1] + \delta[n-2])$

$y[n] = \delta[n] + \delta[n-1] + \delta[n-2] \quad + \quad \delta[n-1] + \delta[n-2] + \delta[n-3] \quad + \delta[n-2] + \delta[n-3] + \qquad + \delta[n-4]$

$$\boxed{y[n] = \delta[n] + 2.\delta[n-1] + 3.\delta[n-2] + \ 2.\delta[n-3] \ + \delta[n-4]}$$

Clasificación de Sistemas

Sistema LINEAL: los sistemas lineales cumplen con la propiedad de SUPERPOSICIÓN. En la Fig. 13 se muestra un sistema con sus entradas y salidas.

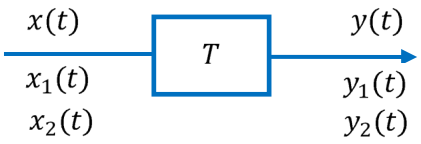


Fig. 13 – Sistema lineal, entradas y salidas

1) La respuesta correspondiente a $x_1(t) + x_2(t)$ es $y_1(t) + y_2(t)$ (ADITIVIDAD)

2) La respuesta correspondiente a $k.x_1(t)$ es $k.y_1(t)$ (HOMOGENEIDAD). Partiendo de esta última premisa, si la entrada es nula ($k = 0$) la salida es nula

$$\begin{cases} T\{x_1(t) + x_2(t)\} = T\{x_1(t)\} + T\{x_2(t)\} \\ T\{a.x(t)\} = a.T\{x(t)\} \end{cases}$$; Linealidad, Debe verificar estas ecuaciones

Sistema INVARIANTE EN EL TIEMPO: Su comportamiento y características permanecen inalterables en el tiempo. Un desplazamiento en la señal de entrada tiene el mismo desplazamiento en la señal de salida.

Un desplazamiento temporal en la señal de entrada tiene por respuesta el mismo desplazamiento en la señal de salida (ver Fig. 14)

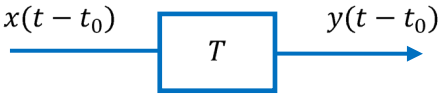


Fig. 14 – Sistema con entrada y salida desplazada

Se deben cumplir la siguiente ecuación para tener Invarianza temporal en tiempo continuo

$T\{x(t-t_0)\} = y(t-t_0) \quad ; \quad \forall t$

Se deben cumplir la siguiente ecuación para tener Invarianza temporal en tiempo discreto

$T\{x[n-n_0]\} = y[n-n_0] \quad ; \quad \forall n$

Sistema BIBO Estable. Los sistemas se denominan BIBO estable si cumple con lo siguiente: *para toda entrada acotada en amplitud, su salida resulta acotada*

BIBO (Boundary Input Boundary Output)

SLIT (sistema lineal e invariante en el tiempo) con y sin memoria:

Los sistemas SIN memoria son aquellos cuya salida solo depende de la entrada del sistema en ese mismo tiempo. Es decir, la salida no depende de la entrada en tiempos pasados o futuros.

En el caso discreto, debe cumplir con: $\boxed{h[n] = 0 \quad para\, n \neq 0}$ Sin memoria

La suma de la convolución se reduce a: $y[n] = k.x[n]$; donde $k = h[0]$ es una constante

Causalidad para los SLIT

El sistema es causal si cumple con lo siguiente: su salida para cualquier instante de tiempo solo depende de los valores que tiene la entrada en el momento presente y/o en el pasado. (No es anticipativo).

La respuesta impulsional de un SLIT causal discreto, teniendo en cuenta la definición debe cumplir:

$\boxed{h[n] = 0 \quad para\, n < 0}$ Causal

Que a su vez implica: $h[n] = \sum_{k=0}^{\infty} x[k].h[n-k]$

Resumen Sistemas LTI conociendo h:

Sin Memoria: $h[n] = A.\delta[n] \quad ; \quad h(t) = A.\delta(t)$

Estable: $\sum_{n=-\infty}^{\infty} |h[n]| < \infty \quad ; \quad \int_{-\infty}^{\infty} |h(t)|.dt < \infty$ (la respuesta al impulso unitario es absolutamente sumable)

Causal: $h[n] = 0$ si $n < 0$; $h(t) = 0$ si $t \leq 0$

Respuesta Impulsional y Respuesta Indicial

Para el sistema de la Fig. 15 se analiza su respuesta impulsional y su respuesta indicial

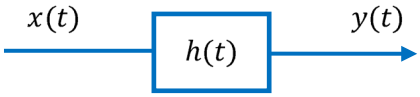


Fig. 15 – Entradas y salidas de un sistema

Sistema LTI, Entrada: $x(t)$ y Salida $y(t)$

Si $x(t) = \delta(t) \implies$ Respuesta IMPULSIONAL (respuesta al impulso): $\boxed{h(t) = y(t)|_{x(t)=\delta(t)}}$

Si $x(t) = u(t) \implies$ Respuesta INDICIAL: $\boxed{g(t) = y(t)|_{x(t)=u(t)}}$

Relación entre Respuesta Indicial y Respuesta Impulsional: $\boxed{h(t) = \frac{dg(t)}{dt}}$

Resolución Numérica:

Euler: $y(t_0) = y_0 \;\; ; \;\; \frac{dy}{dt} = y'(t) = f(y,t)$

Si $dt = h \longrightarrow \frac{y_{i+1} - y_i}{h} = dy_{(ti,xi)}$ Despejamos $\longrightarrow$ $\boxed{y_{i+1} = y_i + h.dy_{(ti,xi)}}$

Serie Exponencial de Fourier (S.E.F.)

$$f(t) = \sum_{n=-\infty}^{\infty} \left(C_n . e^{\,j.n.w_0.t}\right)$$

$$C_n = \frac{1}{T_0} \int_{T_0} f(t).e^{-j.n.w_0.t}.dt$$

$$Si\ n = 0 \rightarrow C_n|_{n=0} = C_0 = \frac{1}{T_0} \int_{T_0} f(t).e^{0}.dt = \frac{1}{T_0} \int_{T_0} f(t).dt$$

$$\rightarrow \; C_0 = \frac{1}{T_0} \int_{T_0} f(t).dt \;\; ; \;\; C_0: Valor\ medio\ de\ f(t)$$

Joseph Fourier

Matemático y físico de origen francés 1768-1830

Serie Trigonométrica de Fourier (S.T.F.)

$$f(t) = \frac{a_o}{2} + \sum_{n=1}^{\infty} [a_n.\cos(n.w_0.t) + b_n.\text{seno}(n.w_0.t)]$$

$$a_n = \frac{2}{T_0} \int_{T_0} f(t).\cos(n.w_0.t)\,.dt \;\; ; \;\; b_n = \frac{2}{T_0} \int_{T_0} f(t).\text{sen}(n.w_0.t)\,.dt$$

$$Si\ n = 0 \rightarrow a_n|_{n=0} = a_0 = \frac{2}{T_0} \int_{T_0} f(t).\cos(0).dt = \frac{2}{T_0} \int_{T_0} f(t).dt$$

$$\rightarrow \; a_0 = \frac{2}{T_0} \int_{T_0} f(t).dt \;\; ; \;\; \frac{a_0}{2}: Valor\ medio\ de\ f(t)$$

Ayuda:

$w_0 = \frac{2.\pi}{T_0}$ $\quad$ $Relación\ de\ Euler$: $\quad e^{j.x} = \cos(x) + j.sen(x)$

$\rightarrow\ cos(x) = \frac{e^{j.x}+e^{-j.x}}{2}$ $\quad ;\quad$ $\text{sen}(x) = \frac{e^{j.x}-e^{-j.x}}{2,j}$

$a_n = \frac{2}{T_0}\ \int_{T_0} f(t).cos(n.w_0.t)\,.dt$ $\quad ;$ Si f(t) Impar: $\quad a_n =$
$\frac{2}{T_0}\ \int_{T_0} f(t)|_{Impar}\,.\ cos(n.w_0.t)_{Par}\,.dt = 0$

-> Si f(t) Impar: $\quad a_n = 0$

$b_n = \frac{2}{T_0}\ \int_{T_0} f(t).sen(n.w_0.t)\,.dt;$ Si f(t) Par: $\quad b_n = \frac{2}{T_0}\ \int_{T_0} f(t)|_{Par}\,.\ \text{sen}(n.w_0.t)_{Impar}\,.dt = 0$

-> Si f(t) Par: $\quad b_n = 0$

Relación entre coeficientes de S.T.F. y S.E.F.

$$C_n = \frac{a_n - j.b_n}{2} \quad ; \quad Reales\{C_n\} = \frac{a_n}{2}\ ; \quad Imag\{C_n\} = -\frac{b_n}{2}$$

Serie Trigonométrica Alternativa de Fourier (S.T.A.F.)

$f(t) = \frac{a_o}{2} + \sum_{n=1}^{\infty}[A_n.\text{seno}(n.w_0.t + \theta_n)]$ $\quad ;\quad C_0 = \frac{a_o}{2}$ $\quad ; A_n = \sqrt{{a_n}^2 + {b_n}^2}$ $\quad \cos(\theta_n)$
$\text{seno}(\theta_n)$ $\quad \theta_n = arctg\frac{a_n}{b_n}$ $\quad ;\ \theta_n\ es\ distinto\ de\ \theta_{Cn}$

Nota: Para pasar de STF a STAF utilizo $\text{seno}(x \pm y) = sen(x).\cos(y) \pm sen(y).\cos(x)$

Teorema de Parseval: Cálculo de Potencias:

$$P = \frac{1}{T_0}\int_{T_0} \lfloor f(t) \rfloor^2\,.dt = \frac{1}{4}{a_0}^2 + \frac{1}{2}\sum_{n=1}^{\infty}\left({a_n}^2 + {b_n}^2\right)$$

$$P = \frac{1}{T_0}\int_{T_0} \lfloor f(t) \rfloor^2\,.dt = \sum_{n=1}^{\infty} \lfloor C_n \rfloor^2$$

Cálculo de la Respuesta de un sistema LTI, para una señal de entrada periódica

Dado el sistema representado en la Fig. 16, si se ingresa una señal periódica: $x(t)$, mediante SEF, la salida del sistema está dado por la siguiente ecuación:

$$y(t) = \sum_{n=-\infty}^{\infty} C_n.H_{(n.w_0).}e^{j.nw_0.t}$$

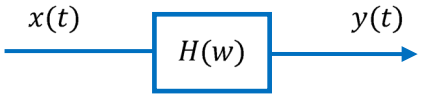


Fig. 16 – Sistema con respuesta en frecuencia $H(w)$

Con S.T.A.F. y $H_{(w)} = |H_{(w)}|.e^{j.\varphi(n.w_0)}$ la salida es

$y(t) = \frac{a_0}{2}.H_{(0)} + \sum_{n=1}^{\infty} A_n.|H_{(n.w_0)}|.sen(n.w_o t + \theta_n + \varphi_{(n.w_0)})$

Ejercicio X

Diseño de un filtro pasa bajo analógico RC

Dado el circuito simple RC de la Fig. 17, con entrada i(t) correspondiente a la señal graficada y salida vc(t). Se pide calcular:

a) Encontrar la Serie de Fourier de la entrada $f(t)$

b) i) Hallar la ecuación diferencial que relaciona entrada i(t) y salida Vo(t).

ii) Hallar la respuesta en frecuencia H(w).

iii) Proponer valores de R y C con el siguiente criterio: las primeras 11 armónicas no deben tener atenuación mayor al 50 %.

iv) Graficar con Matlab®.

$f(t) = \begin{cases} -1 & ; 0 \le t < 2 \\ 1 & ; 2 \le t < 4 \end{cases}$

Tren de pulsos con período T_0 =4, Amplitud +-1

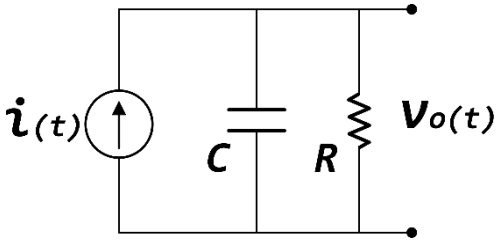


Fig. 17 – Ejemplo de análisis de circuito RC

Resolución

a) Hallar la Serie de Fourier de la entrada

S.T.F.: La función $f(t)$ es impar, entonces $a_n = 0$

$Si\ n = 0 \rightarrow a_n|_{n=0} = a_0 = \frac{2}{T_0}\int_{T_0} f(t).\cos(0).dt = \frac{2}{T_0}\int_{T_0} f(t).dt$

$a_0 = \frac{2}{T_0}\int_{T_0} f(t).dt = \frac{2}{4}\int_{-2}^{2} f(t).dt = 0$ ya que Valor medio de f(t) es 0

$w_0 = \frac{2.\pi}{T_0} = \frac{2.\pi}{4} = \frac{\pi}{2}$; $T_0 = 4$

$$b_n = \frac{2}{T_0} \int_{T_0} f(t).\text{sen}(n.w_0.t)\,.dt = \frac{2}{4}\left[\int_{-2}^{0} -1.\text{sen}(n.w_0.t)\,.dt + \int_{0}^{2} 1.\text{sen}(n.w_0.t)\,.dt\right]$$

$$b_n = \frac{2}{4.n.\frac{\pi}{2}}\left[\cos\left(n.\frac{\pi}{2}.t\right)\Big|_{-2}^{0} - \cos\left(n.\frac{\pi}{2}.t\right)\Big|_{0}^{2}\right]$$

$$b_n = \frac{1}{n.\pi}[\cos(0) - \cos(-\text{n}.\pi) - \cos(\text{n}.\pi) + \cos(0)] = \frac{1}{n.\pi}[2 - 2.(-1)^n]$$

$$b_n = \frac{2}{n.\pi}[1 - (-1)^n]$$

$f(t) = \frac{a_o}{2} + \sum_{n=1}^{\infty}[a_n.\cos(n.w_0.t) + b_n.\text{seno}(n.w_0.t)]$; $a_n = 0$; $a_0 = 0$

$$f(t) = \sum_{n=1}^{\infty}[b_n.\text{seno}(n.w_0.t)]$$

$$f(t) = \frac{4}{1.\pi}\text{seno}(1.w_0.t) + \frac{4}{3.\pi}\text{seno}(3.w_0.t) + \frac{4}{5.\pi}\text{seno}(5.w_0.t) \ldots..$$

$$f(t) = \frac{4}{\pi}\sum_{n=0}^{\infty}\left[\frac{1}{2.n+1}.\text{seno}((2.n+1).w_0.t)\right]$$

b) Resolución:

i) $i(t) = i_c(t) + i_R(t)$; $i_c(t) = C.\frac{dv_0(t)}{dt}$; $i(t) = C.\frac{dv_0(t)}{dt} + \frac{1}{R}v_0(t)$

$\boxed{\frac{dv_0(t)}{dt} + \frac{1}{R.C}v_0(t) = \frac{1}{C}i(t)}$ Resuelvo para tener la derivada de mayor orden con coeficiente 1

ii)

Opción 1 -Cálculo $H_{(w)}$ con autovectores,

Planteo Entrada: $x(t) = i(t) = e^{j.w.t}$; Salida: $y(t) = v_0(t) = H(w).e^{j.w.t}$

Reemplazo en ecuación diferencial $\frac{dv_0(t)}{dt} + \frac{1}{R.C}v_0(t) = \frac{1}{C}i(t)$

$-> \quad j.w.H(w).e^{j.w.t} + \frac{1}{R.C}.H(w).e^{j.w.t} = \frac{1}{C}e^{j.w.t}$

$H(w)\left[j.w + \frac{1}{R.C}\right] = \frac{1}{C}$ $H(w) = \frac{\frac{1}{C}}{j.w+\frac{1}{R.C}}$; $H(w) = \frac{R}{1+j.w.R.C}$

Opción 2 -Cálculo $H_{(w)}$ con Transformada de Fourier

Planteo: $i(t) \xleftrightarrow{F} I(w)$; $v_0(t) \xleftrightarrow{F} V_0(w)$ y su derivada: $\frac{dv_0(t)}{dt} \xleftrightarrow{F} j.w.V_0(w)$

Reemplazo en ecuación diferencial: $\frac{dv_0(t)}{dt} + \frac{1}{R.C}v_0(t) = \frac{1}{C}i(t)$

$j.w.V_0(w) + \frac{1}{R.C}V_0(w) = \frac{1}{C}I(w)$ **(*)**

Luego: $H(w) = \frac{Salida(w)}{Entrada(w)}$ en nuestro caso: $H(w) = \frac{V_0(w)}{I(w)}$, despejo de la ecuación anterior **(*)** obtengo H(w)

$\boxed{H(w) = \frac{V_0(w)}{I(w)} = \frac{R}{1+j.w.R.C}}$; Obtengo módulo y fase: $\lfloor H(w) \rfloor = \frac{R}{\sqrt{1+(.w.R.C)^2}}$;

$1 + j.w.R.C$ esta dividiendo, por eso coloco signo (-) ; $\frac{1}{e^{j.x}} = e^{-j.x}$

$\theta(w) = Fase\ de(R) - arctg\left(\frac{w.R.C}{1}\right)$; $\theta(w) = -arctg(w.R.C)$

iii)

$w_0 = \frac{2.\pi}{T_0} = \frac{2.\pi}{4} = \frac{\pi}{2}$; $T_0 = 4$

Para tener 11 armónicas: $w_{Corte} = n.w_0 = 11.\frac{\pi}{2} \cong 17{,}28\ \frac{rad}{seg}$

H(w) se reduce 50%, hallar R y C: $\frac{1}{2} = \frac{\lfloor H(w_{corte}) \rfloor}{\lfloor H(0) \rfloor} = \frac{\frac{R}{\sqrt{1+(.w_{corte}.R.C)^2}}}{R}$

$\frac{1}{2} = \frac{1}{\sqrt{1+(.w_{corte}.R.C)^2}}$; $1 + (w_{corte}.R.C)^2 = 4$; $C = \frac{\sqrt{3}}{w_{corte}.R}$

Si tomo R=10 KOhm: $C = \frac{\sqrt{3}}{17{,}28\ x\ .10.000} \cong 1{,}002x10^{-5} \cong 10^{-5} Faradios$

Resolviendo en Matlab®

```
% Señal de entrada
T0 = 4 ;    w0 = 2*pi / T0 ;
% Adopto estos valores de RC de manera de tener 11 Armónicos que no bajen más del 50% su amplitud
R= 10e3 ; C= 1e-5 ;

% Respuesta en frecuencias con w continua
w= -50*w0 :0.01:  50*w0 ;
H = R ./ (1+j.*w*R*C) ;
figure ; subplot(2,1,1) ;
plot (w, abs(H)) ;
title('Respuesta en frecuencias con w continua',...
   'Fontsize', 14)
ylabel('modulo(H(w))' , 'Fontsize', 14);
xlabel('w [rad/seg]' , 'Fontsize', 14);grid
subplot(2,1,2) ;
plot (w, angle(H), 'r' );
ylabel('Fase(H(w))' , 'Fontsize', 14); grid
```

Los resultados se muestran en la Fig. 18.

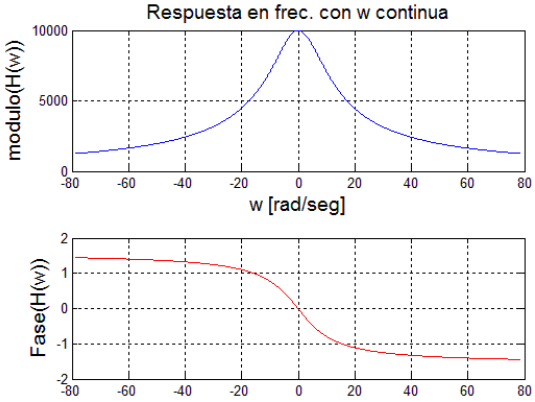


Fig. 18 – Ejemplo de Respuesta en frecuencia para circuito RC pasa bajos

```
% Respuesta en frec. con w discreta
figure ; subplot(2,1,1) ;
n= -50:50 ;
%w0= 10* 2*pi ;
H1 = R ./ (1+j.* n*w0 *R*C) ;
stem (n, abs(H1)) ;
title('Respuesta en frec. con w discreta',...
   'Fontsize', 14)
ylabel('modulo(H(n.w0))' , 'Fontsize', 14);
subplot(2,1,2) ;
stem(n, angle(H1), 'r');
ylabel('Fase(H(n.w0))' , 'Fontsize', 14);
xlabel('n' , 'Fontsize', 16);
```

Los resultados del programa se muestran en la Fig. 19.

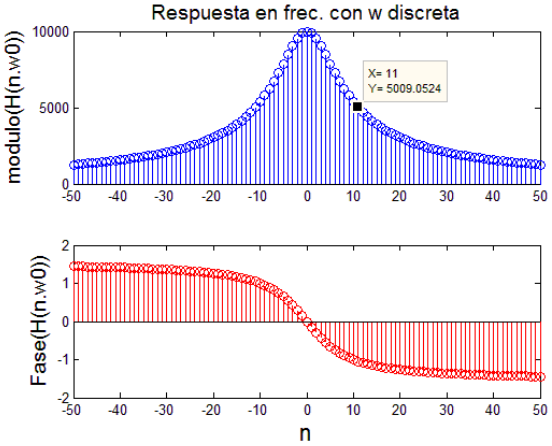


Fig. 19 – Ejemplo de respuesta en frecuencia discreta

Notar que T_0=4; f_0=1/4; w_0=π/2= 1,5708

Ver límites de los ejes x: n=11 (discreta) equivale a 17,28 rad/seg (continua)

11 * π/2 = 17,28

Ver que la amplitud máxima de H(n.wo0), esta correlacionada con los valores de la resistencia R.

Sin embargo, al utilizar Transformada Rápida de Fourier (FFT), se normaliza la amplitud de la respuesta y no depende de los valores de la resistencia R.

Para una señal Real, el módulo del espectro da par y la fase impar

Transformada Continua de Fourier

La transformada de Fourier se puede definir como una operación que transforma o convierte una señal en el *dominio temporal* y obtiene una señal en *el dominio de frecuencia* y viceversa. En la vida cotidiana las señales se describen en el dominio temporal, es decir la señal se expresa en términos de tiempo. En cambio, en el dominio frecuencial, una señal se expresa y/o se muestra respecto a la frecuencia.

$$f(t) \longleftrightarrow F(w)$$

$F(w) = \int_{-\infty}^{\infty} f(t).e^{-i.w.t}.dt$ Transformada de Fourier

Se puede calcular la Transformada de Fourier por definición mediante la expresión anterior, pero en la mayoría de los casos conviene usar tabla de transformadas y propiedades.

$f(t) = \frac{1}{2.\pi}\int_{-\infty}^{\infty} F(w).e^{i.w.t}.dw$ Antitransformada de Fourier

La Transformada inversa de Fourier o anti transformada se puede calcular por definición, pero en muchos casos conviene usar tabla de transformadas y propiedades. En algunos casos se deberá usar fracciones simples y/o división de fracciones para obtener señales similares a las tablas.

Transformada de Fourier para señales Discretas

Series y Transformadas de Fourier		
	Dominio Temporal	***Dominio Frecuencial: Espectro***
Serie Trigonométrica de Fourier (S.T.F.)	$f(t) = \frac{a_o}{2} + \sum_{n=1}^{\infty} [a_n . \cos(n . w_0 . t) + b_n . \text{seno}(n . w_0 . t)]$	$a_0 = \frac{2}{T_0} \int_{T_0} f(t) . dt$ $a_n = \frac{2}{T_0} \int_{T_0} f(t) . \cos(n . w_0 . t) . dt$ $b_n = \frac{2}{T_0} \int_{T_0} f(t) . \text{sen}(n . w_0 . t) . dt$
Serie Exponencial de Fourier (S.E.F.) $C_n = \frac{a_n - j . b_n}{2}$	$\text{f(t)} = \sum_{\text{n}=-\infty}^{\infty} \left(\text{C}_\text{n} . \text{e}^{\text{j.n.w}_0.\text{t}}\right)$	$\text{C}_\text{n} = \frac{1}{\text{T}_0} \int_{\text{T}_0} \text{f(t)} . \text{e}^{-\text{j.n.w}_0.\text{t}} . \text{dt}$
Serie Discreta de Fourier	$f[n] = \sum_{k=\{N_0\}} [C_k . e^{j.k.\Omega_0.n}]$; $\Omega_0 = \frac{2.\pi}{N_0}$	$C_k = \frac{1}{N_0} \sum_{k=\{N_0\}} [f[n] . e^{-j.k.\Omega_0.n}]$
Transformada Continua de Fourier $f(t) \longleftrightarrow F(w)$	$f(t) = \frac{1}{2.\pi} . \int_{-\infty}^{\infty} F(w) . e^{j.w.t} . dw$	$F(w) = \int_{-\infty}^{\infty} f(t) . e^{-j.w.t} . dt$
Transformada de Fourier de una secuencia (TFS) o también llamada Transformada de Fourier en tiempo discreto (TFTD) $f[n] \longleftrightarrow F(\Omega)$ $f[n] \longleftrightarrow F(e^{j.w})$	$f[n] = \frac{1}{2.\pi} \int_{-\pi}^{\pi} F(\Omega) . e^{j.\Omega.n} . d\Omega$	$F(\Omega) = \sum_{n=-\infty}^{\infty} f[n] . e^{-j.\Omega.n}$
Transformada Discreta de Fourier (DFT) $f[n] \longleftrightarrow F[k]$	$f[n] = \frac{1}{N_0} \sum_{k=\{N_0\}} [F[k] . e^{j.k.\Omega_0.n}]$ $\Omega_0 = \frac{2.\pi}{N_0}$; $n = \{0.1, 2 \ldots N_0 - 1\}$	$F[k] = \sum_{k=\{N_0\}} [f[n] . e^{-j.k.\Omega_0.n}]$
La ***TFTD, es decir la Transformada Fourier en tiempo discreto***, puede analizarse ***discretizando*** $F(\Omega)$:		$\boxed{F(\text{k} . \Omega_0) = N_0 . C_k}$ **FFT:** Si $N_0 = 2^k$ puede implementarse un algoritmo optimizado, rápido, denominado FFT (Transformada Rápida de Fourier: Fast Fourier Transform)

Ejercicios de Transformadas de Fourier.

Ejercicio XI

Repaso y comparación de Transformadas

Partiendo de la Serie Exponencial de Fourier (S.E.F.), obtener la Serie Discreta de Fourier y también las Transformadas de Fourier. Analizar las señales (periodicidad y dominio discreto o continuo)

Serie Exponencial de Fourier (S.E.F.)	$f(t)$ Cn	t continuo Cn discreto	$f(t)$ Periódica Cn Aperiódicas	Frecuencias angulares discretas: $w = n.w_0$ $w_0 = \frac{2.\pi}{T_0}$

Partiendo de S.E.F., discretizamos t:

Serie Discreta de Fourier (similar a DFT)	$f[n]$ C_k $\Omega_0 = \frac{2.\pi}{N_0}$	n discreto k discreto	$f[n]$ Periódica C_k Periódica N_0: Período temporal $\Omega_0 = \frac{2.\pi}{N_0}$	**Partiendo de S.E.F., discretizamos t, el espectro se vuelve Periódico** Es decir, si muestreamos la señal $f(t)$ obtenemos Serie de Fourier Discreta:

Partiendo de S.E.F., Si $T_0 \to \infty$, entonces $w_0 \to 0$,tenemos w continua, el espectro se vuelve continuo

Transformada Continua de Fourier (TFC)	$f(t) \longleftrightarrow F(w)$	t y w continuos	$f(t)$ y $F(w)$ Aperiódicas	Tomamos S.E.F. con $\boldsymbol{T_0 \to \infty}$, obtuvimos TFC ***$f(t)$ se convierte en Aperiódica*** ***w continuo*** (Integrales en vez de Sumatorias)

Partiendo de TFC, Si muestreamos la señal $f(t)$ obtenemos TFS: discretizamos t

TFS	$f[n] \longleftrightarrow F(\Omega)$ $f[n] \longleftrightarrow F(e^{j.w})$	n discreto Ω continuo $\boxed{\Omega = \mathrm{w}.T_s = \frac{2.\pi.w}{w_s}}$ Frec. Normalizada	$f(t)$ Aperiódicas $F(\Omega)$ Periódica con período $2.\pi$	Tomamos TFC, ***Discretizamos $t \to n$, el espectro se vuelve periódico*** (Muestreo $\boldsymbol{n.T_s}$), obtuvimos TFS
DFT	$f[n] \longleftrightarrow F[k]$	n discreto k discreto	$f[n]$ Periódica $F[k]$ Periódica con período N_0 $\Omega_0 = \frac{2.\pi}{N_0}$	Tomamos TFS, ***Discretizamos $\Omega \to k.\Omega_0$*** $F(\mathrm{k}.\Omega_0) = N_0.C_k$ ***$f[n]$ se vuelve periódica,*** obtuvimos DFT

Continuación para Transformadas de Laplace y Z

Partiendo de Transformada de Laplace, donde s $= \sigma + j.w$: $f(t) \longleftrightarrow F(s)$ Si la ROC contiene al eje j.w , entonces podemos tomar s $= j.w$ para analizar «Respuesta en Frecuencia $\mathrm{F}(w)$» (TFC)
- Partiendo de TFS, $f[n] \longleftrightarrow F(\Omega)$ o $f[n] \longleftrightarrow F\left(e^{j.w}\right)$, si tomamos $z = e^{j.w}$ obtenemos Transformada Z: $f[n] \longleftrightarrow F(z)$
Partiendo de Transformada Z: $f[n] \longleftrightarrow F(z)$, si la ROC contiene a la Circunferencia de Radio Unitario $\|z\| = 1$, entonces podemos reemplazar $z = e^{j.w}$ para analizar «Respuesta en Frecuencia $F(\Omega)$» (TFS)

Serie Discreta de Fourier

Dada una secuencia periódica $x[n]$ con período fundamental N_0, $k = \{N_0\}$ o $k = \{0.1, 2, \ldots . N_0 - 1\}$

$$C_k = \frac{1}{N_0} \sum_{k=\{N_0\}} \left[f[n].e^{-j.k.\Omega_0.n}\right]$$

Anti-Transformada Discreta de Fourier: $f[n] = \sum_{k=\{N_0\}} \left[C_k.e^{j.k.\Omega_0.n}\right]$; $\Omega_0 = \frac{2.\pi}{N_0}$

Transformada Discreta de Fourier: DFT

$$f[n] \longleftrightarrow F[k]$$

$f[n] = \frac{1}{N_0} \sum_{k=\{N_0\}} \left[X[k].e^{j.k.\Omega_0.n}\right]$; $\Omega_0 = \frac{2.\pi}{N_0}$; $n = \{0.1, 2, \ldots . N_0 - 1\}$

$$F[k] = \sum_{k=\{N_0\}} \left[f[n].e^{-j.k.\Omega_0.n}\right]$$

Transformada Fourier de una secuencia (TFS). También conocida como Transformada de Fourier en tiempo discreto (siglas en inglés TFTD)

$f[n] \longleftrightarrow F(\Omega)$ $\boxed{F(\Omega) = \sum_{n=-\infty}^{\infty} f[n].e^{-j.\Omega.n}}$; $\boxed{f[n] = \frac{1}{2.\pi} \int_{-\pi}^{\pi} F(\Omega).e^{j.\Omega.n}.d\Omega}$

Parseval : $\boxed{\sum_{n=-\infty}^{\infty} |f[n]|^2 = \int_{-\pi}^{\pi} |F(\Omega)|^2.d\Omega}$

La *TFTD (Transformada de Fourier en tiempo discreto)*, puede analizarse *discretizando* $F(\Omega)$:

$$\boxed{F(\mathrm{k}.\Omega_0) = N_0.C_k}$$

Filtros FIR e IIR:

Filtro de Respuesta Finita al Impulso, conocido como filtro FIR (en inglés Finite Impulse Response)

Ejemplo: $h[n] = \frac{\delta[n] + \delta[n-1] + \delta[n+1]}{3}$ Posee una cantidad *Finita* de términos

Filtro IIR (Respuesta Infinita al Impulso: Infinite Impulse Response)

Ejemplo: $h[n] = a^n.u[n]$

Ver descripción más completa de FIR e IIR en Práctica de Transformada Z

Teorema de Muestreo de Nyquist-Shannon

$$\boxed{f_s > 2.f_{max} \equiv 2.B} \quad ; \quad \boxed{w_s > 2.w_{max}}$$

Normalizando tenemos esta otra nomenclatura: $\boxed{\Omega = \mathrm{w}.T_s = \frac{2.\pi.w}{w_s} = \frac{2.\pi.f}{f_s}}$ Frecuencia Normalizada

Si $\Omega_M < \frac{\pi}{T} \quad \rightarrow \mathrm{w}_M < \pi$

En la Fig. 20 los espectros no se deben superponer.

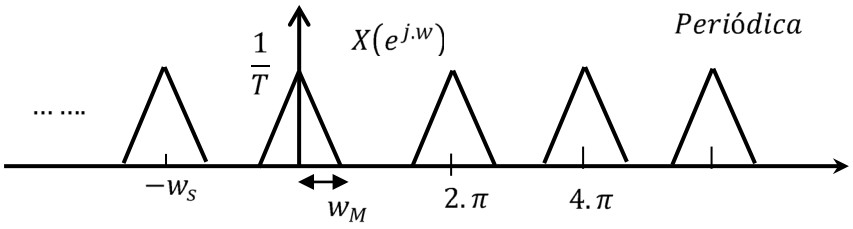


Fig. 20 – Análisis del Teorema de Nyquist mediante un ejemplo de espectro periódico

Transformada Fourier de una Secuencia: TFS

Ejercicio XII

Análisis de filtros con Transformada Fourier de una secuencia

Graficar la respuesta en frecuencia para el sistema determinado por la siguiente ecuación

$$y[n] = x[n] + x[n - N_1]$$

Resolución:

Resolvemos por Tablas y propiedades de la TFS (Transformada Fourier de una Secuencia):

$x[n] \longleftrightarrow X(\Omega)$

$\boxed{x[n - n_0] \longleftrightarrow e^{-j.\Omega.\ n_0}.X(\Omega)}$ (tablas)

Transformamos la Ecuación en diferencias: $y[n] = x[n] + x[n - N_1]$

$Y(\Omega) = X(\Omega) + e^{-j.\Omega.N_1}.X(\Omega) = X(\Omega)\left(1 + e^{-j.\Omega.N_1}\right)$

$H(\Omega) = \frac{Y(\Omega)}{X(\Omega)} = 1 + e^{-j.\Omega.N_1}$; Sacamos Factor común $e^{-j.\Omega.\frac{N_1}{2}}$ para obtener un coseno

$H(\Omega) = \frac{Y(\Omega)}{X(\Omega)} = e^{-j.\Omega.\frac{N_1}{2}}.\left(e^{j.\Omega.\frac{N_1}{2}} + e^{-j.\Omega.\frac{N_1}{2}}\right) = e^{-j.\Omega.\frac{N_1}{2}}.2.\cos\left(\Omega.\frac{N_1}{2}\right)$

$$\boxed{H(\Omega) = 2.\cos\left(\Omega.\frac{N_1}{2}\right).e^{-j.\Omega.\frac{N_1}{2}}}$$

Buscamos los ceros del coseno:

$\cos\left(\Omega.\frac{N_1}{2}\right) = 0 \quad \rightarrow \quad \Omega.\frac{N_1}{2} = \frac{\pi}{2} \quad \rightarrow \quad \boxed{\Omega_{cero} = \frac{\pi}{N_1}}$

Miramos la ecuación En diferencias y siendo sistema LTI

$\rightarrow \quad \boxed{h[n] = \delta[n] + \delta[n - N_1]}$ *Filtro FIR (Respuesta Finita al Impulso)*

Recordando Respuesta impulsional:

$y[n] = x[n] * h[n]$; Si Tomamos: $x[n] = \delta[n]$ obtenemos $h[n]$

```
w= -pi: pi/5000:  pi;
N=4 ;
f=  2*cos(N/2*w).*exp(-j*N/2*w) ;
subplot(2,1,1)
plot(w,abs(f), 'linewidth', 3)
grid on; title('absoluto(H)')
subplot(2,1,2)
plot(w,angle(f), 'linewidth', 3)
grid on; title('Fase(H)')
```

En la Fig. 21 se muestra la respuesta, se observa que es un filtro elimina bandas

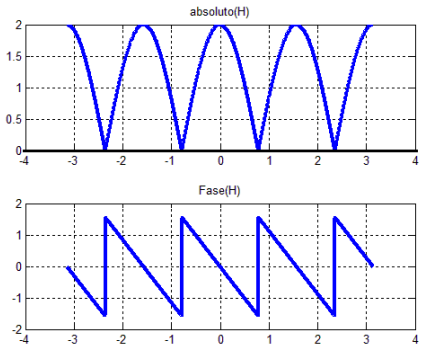


Fig. 21 –Respuesta en frecuencias para un Filtro Elimina Bandas

Ejercicio XIII

Análisis de sistemas con Transformada Fourier de una secuencia

Graficar la respuesta en frecuencia del sistema determinado por la siguiente ecuación

$$y[n] - a.y[n-1] = x[n]$$

Resolución

Resolvemos por Tablas y propiedades TFS: $x[n] \longleftrightarrow X(\Omega)$

$\boxed{x[n-n_0] \longleftrightarrow e^{-j.\Omega.\ n_0}.X(\Omega)}$ (tablas)

Transformamos la Ecuación en diferencias:

$Y(\Omega) - a.e^{-j.\Omega.1}.Y(\Omega) = X(\Omega)$; $Y(\Omega)\left(1 - a.e^{-j.\Omega.1}\right) = X(\Omega)$

$\boxed{H(\Omega) = \frac{Y(\Omega)}{X(\Omega)} = \frac{1}{1-a.e^{-j.\Omega}}}$

Antitransformando obtenemos $h[n]$

$a^n.u[n] \longleftrightarrow \frac{1}{1-a.e^{-j.\Omega}}$ (tablas TFS) ; $|a| < 1$

Si $|a| < 1$ $\boxed{h[n] = a^n.u[n]}$ *Filtro IIR (Respuesta Infinita al Impulso: Infinite Impulse Response)*

Graficamos: $H(\Omega) = \frac{Y(\Omega)}{X(\Omega)} = \frac{1}{1-a.e^{-j.\Omega}}$

$$|H(\Omega)| = \frac{1}{|1-a.e^{-j.\Omega}|} = \frac{1}{|1-a.\cos(\Omega)-j.asen(\Omega)|} = \frac{1}{\sqrt{(1-a.\cos(\Omega))^2+.a^2.(sen(\Omega))^2}}$$

Cuando $\Omega = \pi$ el denominador es máximo → $|H(\Omega)|$ es mínimo

```
w= -3*pi: pi/5000:  3*pi;
a=0.5 ;
f=  1 ./ ( a-exp(-j*w) ) ;
subplot(2,1,1)
plot(w,abs(f), 'linewidth', 3)
grid on; title('absoluto(H)')
subplot(2,1,2)
plot(w,angle(f), 'linewidth', 3)
grid on;   title('Fase(H)')
```

En la Fig. 22 se muestran los resultados obtenidos para el filtro IIR

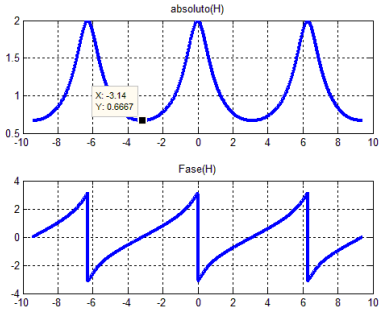


Fig. 22 – Ejemplo de Respuesta en frecuencia de un filtro IIR

Ejercicio XIV

Análisis de Filtro de media móvil con Transformada Fourier de una secuencia

Calcular, la respuesta en frecuencia del sistema LTI de entrada $x[n]$ y salida $y[n]$. Calcular además la respuesta al impulso $h[n]$ y graficar un bosquejo de la respuesta de módulo

$$y[n] = \frac{1}{4}.(x[n] + x[n-1] + x[n-2] + +x[n-3])$$

Corresponde a Filtro de Media Móvil

Resolución:

Utilizamos TFS: $x[n] \longleftrightarrow X(\Omega)$ Resolvemos por Tablas y propiedades TFS

$x[n] \longleftrightarrow X(\Omega)$; $x[n-n_0] \longleftrightarrow e^{-j.\Omega.\ n_0}.X(\Omega)$ (tablas)

Transformamos Ecuación en diferencias:

$$Y(\Omega) = \frac{1}{4}.\left(X(\Omega) + e^{-j.\Omega}.X(\Omega) + e^{-2j.\Omega}.X(\Omega) + e^{-3j.\Omega}.X(\Omega)\right)$$

$$Y(\Omega) = \frac{1}{4}.X(\Omega)\left(1 + e^{-j.\Omega} + e^{-2j.\Omega} + e^{-3j.\Omega}\right)$$

$$\boxed{H(\Omega) = \frac{Y(\Omega)}{X(\Omega)} = \frac{1+e^{-j.\Omega}+e^{-2j.\Omega}+e^{-3j.\Omega}}{4}} \rightarrow \boxed{h[n] = \frac{1}{4}.(\delta[n] + \delta[n-1] + \delta[n-2] + \delta[n-3])}$$

$h[n]$ Tiene cantidad Finita de Impulsos: Filtro FIR

```
w= -2*pi: pi/5000:  2*pi;
f= 1/4* (1+ exp(-i*w)+exp(-2*i*w) +exp(-3*i*w) )  ;
subplot(2,1,1)
plot(w,abs(f), 'linewidth', 3); grid on
xlim([-2*pi  2*pi])
subplot(2,1,2)
plot(w,angle(f), 'linewidth', 3)
```

En la Fig. 23 se muestran los resultados, se observa que es un filtro pasa bajos.

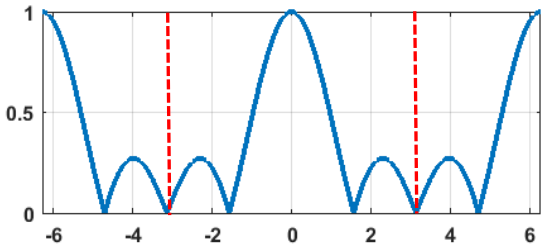


Fig. 23 – Respuesta en frecuencia del Filtro de Media Móvil

Se observa que es un *filtro de Respuesta Finita al Impulso (FIR: Finite Impulse Response)*, cuya respuesta en frecuencia puede identificarse con la respuesta de un *Filtro Pasa Bajos.*

$$\boxed{H(\Omega) = \frac{Y(\Omega)}{X(\Omega)} = \frac{1+e^{-j.\Omega}+e^{-2j.\Omega}+e^{-3j.\Omega}}{4}}$$ Reemplazamos $z = e^{j.\Omega}$

$$\rightarrow H(z) = \frac{1}{4}.(1 + z^{-1} + z^{-2} + z^{-3}) \quad ; \quad H(z) = \frac{1}{4}.\frac{z^3+z^2+z^1+1}{z^3} = \frac{\left[\frac{1}{4}\ \frac{1}{4}\ \frac{1}{4}\ 1\right]}{[1\ 0\ 0\ 0]}$$

$$\boxed{H(z) = \frac{1}{4}.\left(\frac{1+z^{-1}+z^{-2}+z^{-3}}{1}\right) = \frac{\left[\frac{1}{4}\ \frac{1}{4}\ \frac{1}{4}\ 1\right]}{[1]}}$$

Resolvemos en Matlab®

```
fs=100 ; %Frecuencia de Muestreo
dt =1/fs ; % Ts
t=0: dt: 5-dt ;
x = sin(2*pi*1*t); % seno de 1 Hz
% Agregamos Ruido
x1 = x +0.5 *rand(1,length(x));
% Filtramos la señal
Num=[1/4  1/4  1/4  1/4];
Den=[1 ];
x_filtrada = filter(Num, Den, x1) ;
subplot(211) ; plot(t, x1); axis tight;
subplot(212) ; plot(t, x_filtrada); axis tight;
```

En la Fig. 24 se muestran los resultados

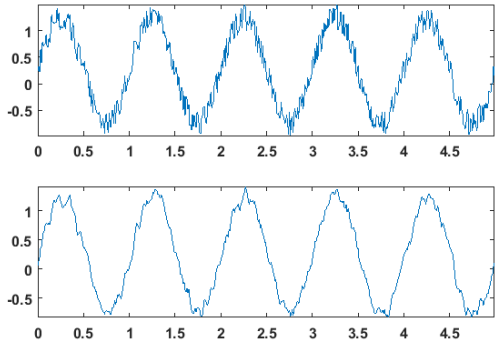


Fig. 24 – Ejemplo de filtrado de un tono puro contaminado con ruido

Ejercicio XV

Filtro Multi eco de Sonido

Diseño de un Filtro Multi eco de Sonido

Ecuación en Diferencias del Filtro Multi eco de Sonido:

$$y[n] - \alpha . y[n - N] = x[n - N]$$

Resolución:

Utilizamos TFS (Transformada Fourier de una Secuencia): $x[n] \longleftrightarrow X(\Omega)$

$\boxed{x[n - n_0] \longleftrightarrow e^{-j.\Omega.\ n_0}.X(\Omega)}$ (tablas)

Transformamos la Ecuación en diferencias:

$$Y(\Omega) - \alpha . e^{-j.\Omega.N}.Y(\Omega) = e^{-j.\Omega.N}.X(\Omega) \quad ; \quad Y(\Omega)\left(1 - \alpha . e^{-j.\Omega.N}\right) = \alpha . e^{-j.\Omega.N}.X(\Omega)$$

$$\boxed{H(\Omega) = \frac{Y(\Omega)}{X(\Omega)} = \frac{e^{-j.\Omega.N}}{1-\alpha.e^{-j.\Omega.N}} = \frac{1}{e^{j.\Omega.N}-\alpha}}$$

Antitransformando obtenemos $h[n]$

$a^n.u[n] \longleftrightarrow \frac{1}{1-a.e^{-j.\Omega}}$ (tablas TFS) ; $|a| < 1$

$x[n \pm n_0] \longleftrightarrow e^{\pm j.\Omega.n_0}.X(\Omega)$ (tablas)

Si $|a| < 1$ $\boxed{h[n] = a^{(n-N)}.u[n-N]}$ *Filtro IIR (Respuesta Infinita al Impulso)*

Graficamos: $H(\Omega)$

```
w= -pi: 0.001:  pi;
% Tomamos estos valores de ejemplo
N=5 ;  alfa = 0.8 ;
f=exp(-i*w*N) ./ ( 1- alfa * exp(-i*w*N)) ;
plot(w,abs(f), 'linewidth', 3); grid on;
```

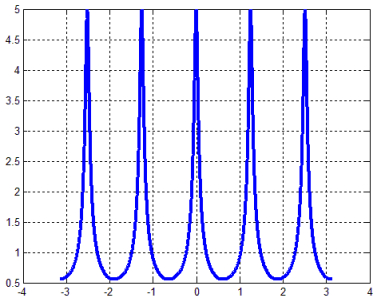


Fig. 25 – Respuesta en frecuencia del filtro multi eco analizado

Diagrama en bloques en el tiempo:

$y[n] - \alpha.y[n-N] = x[n-N]$; Sistema LTI, Desplazamos N

$y[n+N] - \alpha.y[n] = x[n]$

$y[n+N] = x[n] + \alpha.y[n]$

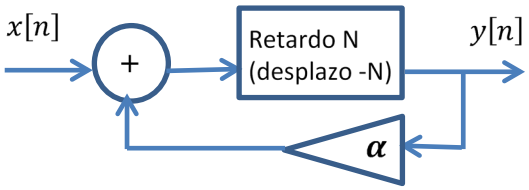


Fig. 26 – Diagrama en bloques temporal del filtro multi eco

Diagrama en bloques TFS (Transformada Fourier de una Secuencia):

$Y(\Omega) - \alpha.e^{-j.\Omega.N}.Y(\Omega) = e^{-j.\Omega.N}.X(\Omega)$

Multiplicamos por $e^{j.\Omega.N}$ $Y(\Omega).e^{j.\Omega.N} - \alpha.Y(\Omega) = X(\Omega)$

$$Y(\Omega).e^{j.\Omega.N} = X(\Omega) + \alpha.Y(\Omega)$$

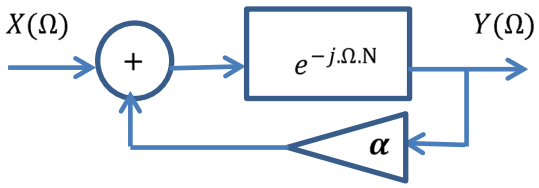


Fig. 27 – Diagrama en bloques TFS del filtro multi eco

Diagrama en bloques TZ (Transformada Z):

$$Y(\Omega).e^{j.\Omega.N} = X(\Omega) + \alpha.Y(\Omega)$$

Reemplazamos: $z = e^{j.\Omega}$

$$Y(z).z^{N} = X(z) + \alpha.Y(z)$$

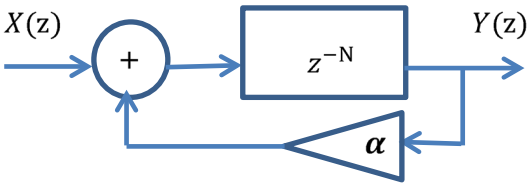


Fig. 28 – Diagrama en bloques del filtro multi eco para análisis con Transformada Z

$$\boxed{H(\Omega) = \frac{Y(\Omega)}{X(\Omega)} = \frac{1}{e^{j.\Omega.N} - \alpha}} \quad ; \quad \boxed{H(z) = \frac{Y(z)}{X(z)} = \frac{1}{z^{N} - \alpha}}$$

Ejemplo aplicado a un filtro Multi eco real de sonido:

$$\alpha = 0{,}8 \quad ; \; N = 2901 \qquad \boxed{H(z) = \frac{Y(z)}{X(z)} = \frac{1}{z^{2901} - 0{,}8} = \frac{[0\ 0\ 0\ \ldots\ldots\ldots\ldots\ldots 1\ ]}{[1\ 0\ 0\ 0\ \ldots\ldots\ldots\ldots -0{,}8]}}$$

```
%% Filtro
% Leemos archivo y coeficientes del filtro
  [x, fs]=audioread('Hello.mp3');
   num=[1];  den=[1,zeros(1,2900),-0.8];
% Mostramos Transferencia: H(z):
   Hz=tf(num,den,fs) % r=44100 (fs) % en continuo: Hs=tf(num,den)
    freqz(num, den) ; xlim([0  0.1])
% Filtramos la señal
   x_filtrada =filter(num, den, x);
   audiowrite( 'salida.wav' , x_filtrada , fs )
   sound( 0.1* x , fs ) ;
   pause() ;
```

```
   sound( 0.15* x_filtrada , fs ) ;
% Agregado para analizar el filtro con N=11
% Respuesta al impulso
   I =[1,zeros(1,60)];
   numi=[0,zeros(1,10),1];  deni=[1,zeros(1,10),-0.8];
% Muestro Transferencia: H(z):
   Hz2=tf(numi,deni,fs)  % r=44100 (fs)
% Filtro
   d2= filter(numi,deni,I);
   figure(1);  stem(d2);grid;
   xlabel('Número de muestra');    ylabel('Amplitud');
   title(['Respuesta al impulso']);
% Magnitud de la Respuesta en frecuencia
   figure(2);
   [h1,w] = freqz(numi,deni,512);
   plot(w/pi,20*log10(abs(h1)));grid;
   xlabel('Frecuencia normalizada');    ylabel('Magnitud');
   title(['Respuesta en frecuencia']);
```

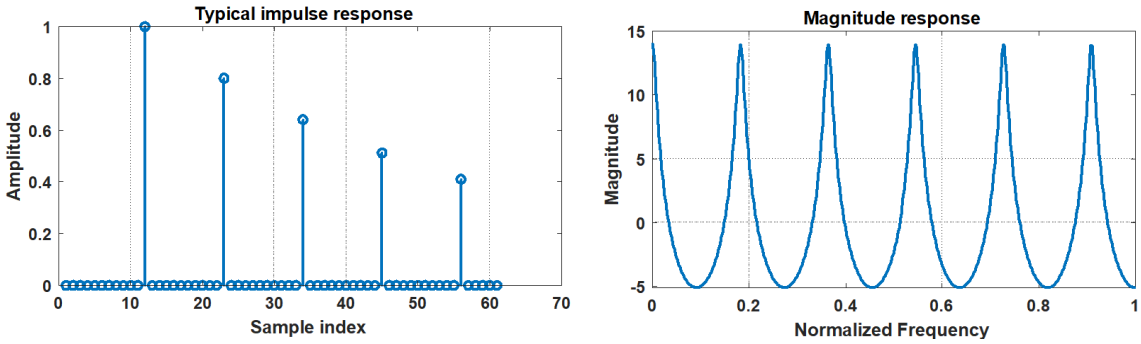


Fig. 29 – Respuesta en frecuencia y respuesta al impulso del filtro multi eco

Ejercicio XVI

Diseño y comparación de Filtros con la herramienta filterDesigner de Matlab®. Breve introducción

Dada la siguiente señal diseñar distintos filtros que eliminen las componentes de 300 y 400 Hz. Comprar las respuestas de cada filtro y las señales obtenidas

y= sin(2*pi*200*t) + sin(2*pi*300*t) + sin(2*pi*400*t) ;

a) En la consola de Matlab® escribir:

>> *filterDesigner*

b) Seleccionar y completar:

FIR ; Specify order: 10; Fs: 1000; Fpass: 220; Fstop: 250

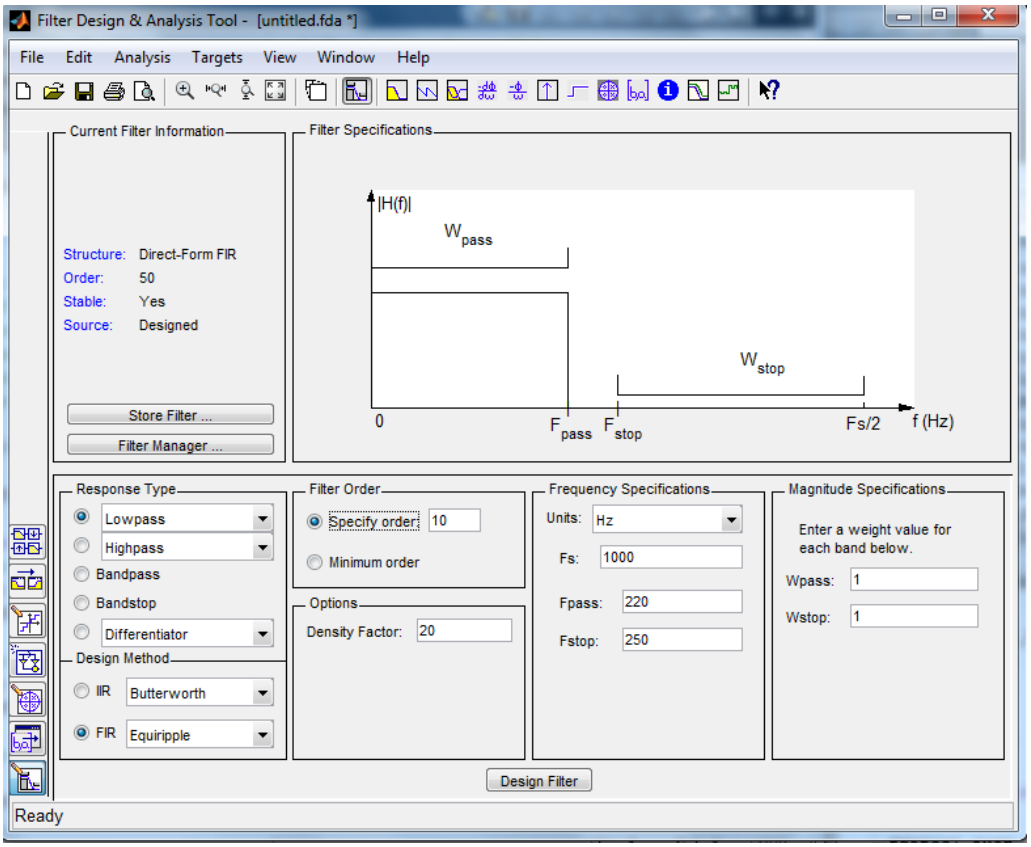


Fig. 30 – Pantalla principal de la herramienta filterDesigner de Matlab®

c) Luego: *Design filter*

d) Probar los siguientes botones:

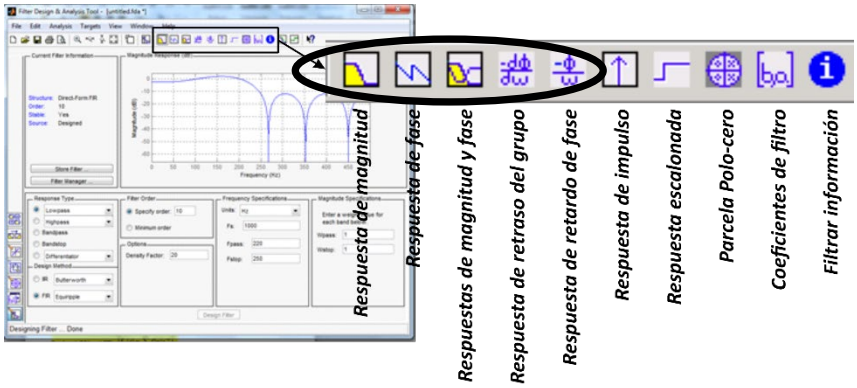


Fig. 31 – Botones de análisis de la herramienta filterDesigner

e) Luego: File-> Export

Para filtro FIR Selecciono: Workspace y Coefficients

Numerator: Num1

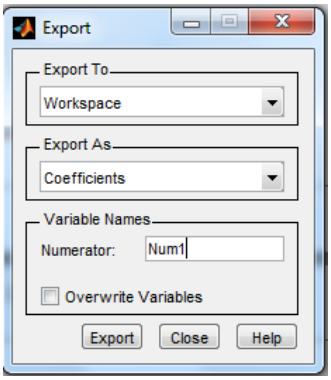


Fig. 32 – Pantalla de exportación de la herramienta filterDesigner

f) Cambiar Orden del filtro: 15

Design filter

Luego: File-> Export

Para filtro FIR Selecciono: Workspace y Coefficients

Numerator: Num2

g) Cambiar a filtro IIR

iir-> butterworth, orden 5, Fs=1000, Fpass=220

Design filter

Luego: File-> Export

Para filtro IIR Selecciono: Workspace y Objects

Nombre: Hd1

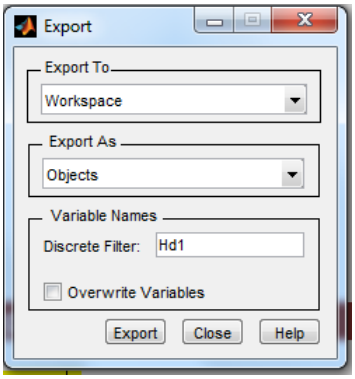


Fig. 33 – Pantalla de exportación de objetos de la herramienta filterDesigner

h) Comparamos las Respuestas de los Filtros con *filterDesigner*: Amplitud y Fase (ver paso 4)

FIR, orden 10

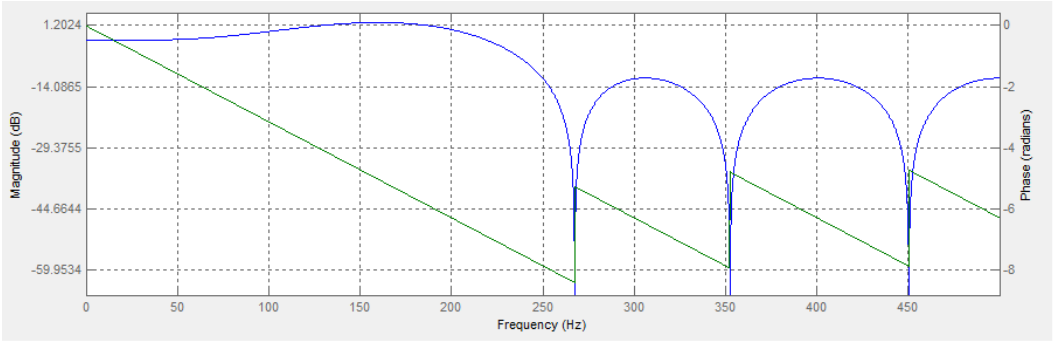


Fig. 34 – Respuesta en frecuencia para el filtro FIR de orden 10

FIR, orden 15

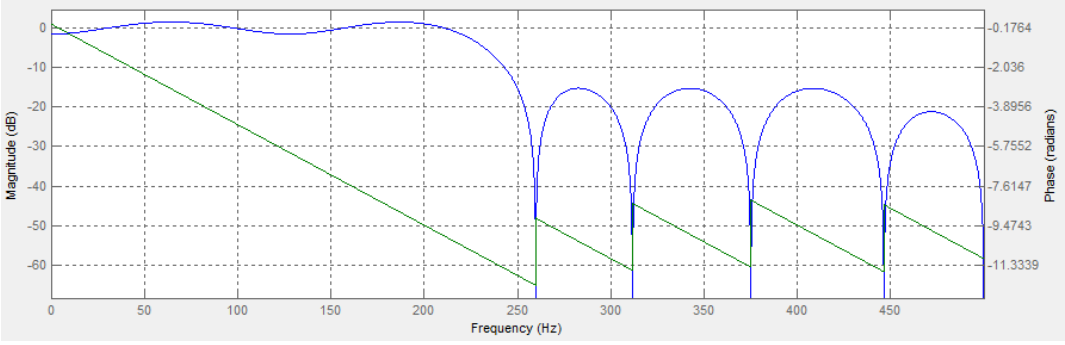


Fig. 35 – Respuesta en frecuencia para el filtro FIR de orden 15

IIR, orden 5

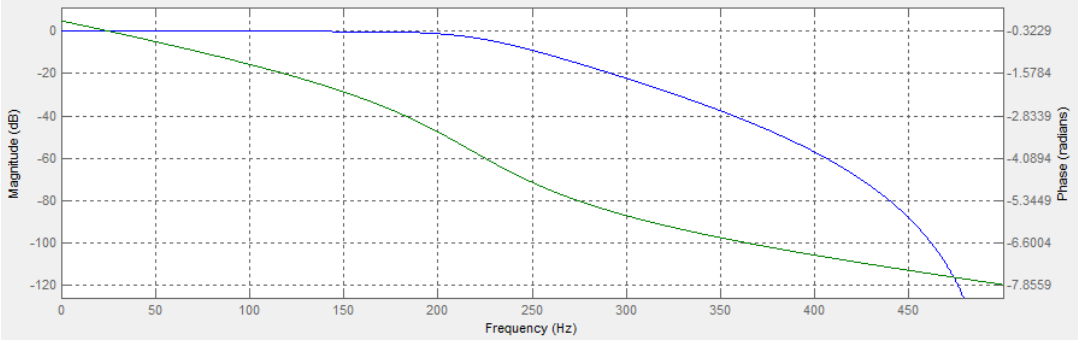


Fig. 36 – Respuesta en frecuencia para el filtro IIR de orden 5

i) en consola de Matlab®:

```
save('coeficientes_filtros.mat')
```

j) Copiar y ejecutar el siguiente script:

```
close all; clear all;
dt = 1/1000 ;
t=0: dt: 1 ;
y= sin(2*pi*200*t)  + sin(2*pi*300*t)  + sin(2*pi*400*t) ;
N = length(y) ;
Esp_y =   abs(  fft(y ,N)/ N   ) ;
fs = 1/ dt ;
f= 0: fs/N : fs*(1-1/N) ;
%% Función butter
% % [z, p, k] =  butter (n, Wn)  esta función diseña un filtro digital Butterworth, orden n, pasa bajos
% con frecuencia de corte normalizada Wn. Devuelve los ceros y los polos en z y p, con longitud n y k.
% La frecuencia de corte es aquella frecuencia donde la respuesta de magnitud del filtro
% cae 1 / raíz (2). La frecuencia de corte normalizada Wn tiene que estar en el rango de 0 a 1, donde
% el valor 1 corresponde a la frecuencia de Nyquist.


%% Función fft
% Y = fft (X, n) esta función devuelve la DFT con n puntos.
% Para longitudes de X menores que n, el vector X se completa con ceros hasta tener longitud n.
% Para longitudes de X mayores que n, se trunca la secuencia X. En el caso de matrices,  la longitud
% de sus columnas se ajusta de la misma forma.
%%
[b ,a] = butter( 4, 220/500 , 'Low' ) ;  % 250/500
yf1 = filter(b,a,y) ;
Esp_yf1 =   abs(  fft(yf1)/ N   ) ;
```

```
load('coeficientes_filtros') ;
% Usamos filterDesigner --> exportamos como coeficientes: Num1:    fir orden 10, Fs=1000,
%Fpass=220, Fstop=250
yf2 = filter(Num1, 1, y) ;
Esp_yf2 =   abs(  fft(yf2)/ N   ) ;
 % Usamos filterDesigner --> exportamos como coeficientes: Num2:    fir orden 15, Fs=1000,
%Fpass=220, Fstop=250
yf3 = filter(Num2, 1, y) ;
Esp_yf3 =   abs(  fft(yf3)/ N   ) ;
% Usamos filterDesigner --> exportamos como objeto Hd1:    iir-> butterworth, orden 5,
%Fs=1000, Fpass=220
yf4 = filter(Hd1, y) ;
Esp_yf4 =   abs(  fft(yf4)/ N   ) ;

figure; subplot(321)
plot( f, Esp_y ) , axis tight ; title('Señal sin filtrar')
subplot(322)
plot( f, Esp_yf1 ), axis tight ;
title('Señal filtrada con funcion butter-> butterworth, orden 4')
subplot(323)
plot( f, Esp_yf2 ), axis tight ;
title('Señal filtrada con fir orden 10')
subplot(324)
plot( f, Esp_yf3 ), axis tight ;
title('Señal filtrada con fir orden 15')
subplot(325)
plot( f, Esp_yf4 ), axis tight ;
title('Señal filtrada con iir-> butterworth, orden 5')
```

Comparar las distintas respuestas de los filtros de la Fig. 37

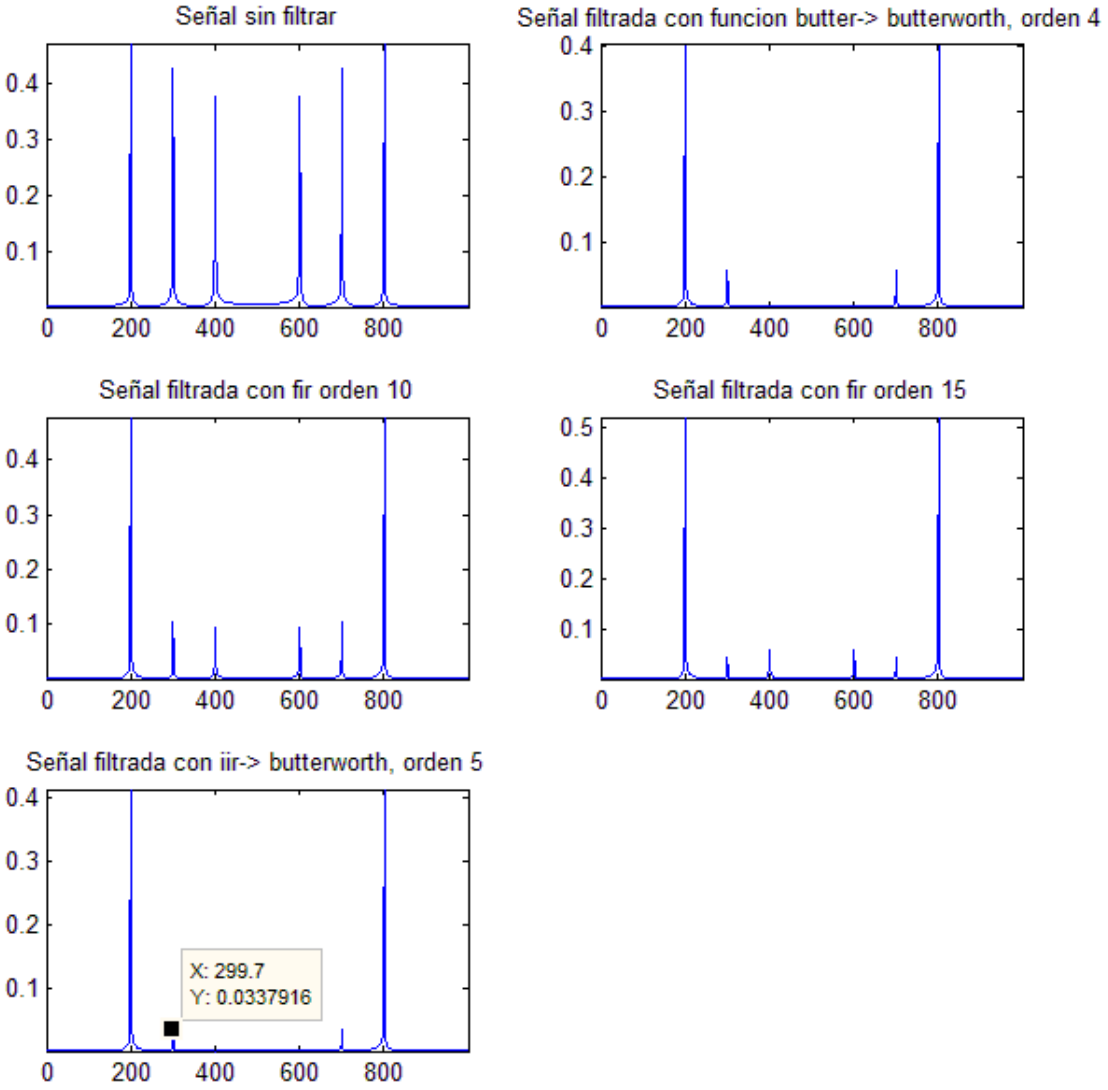


Fig. 37 – Comparación de resultados del filtrado con diferentes filtros

Transformada de Laplace

Mostramos la definición de la transformada de Laplace (T.L.) unilateral, bilateral y T.L. inversa

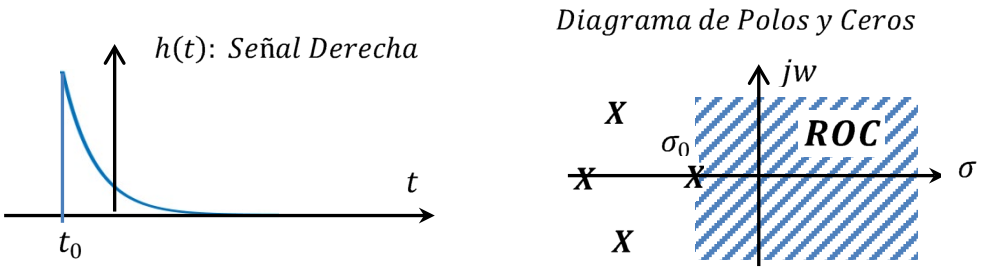
$$F(s) = \int_0^{+\infty} f(t).e^{-s.t}dt \quad \text{T.L. Unilateral}$$

$$F(s) = \int_{-\infty}^{+\infty} f(t).e^{-s.t}dt \quad \text{T.L. Bilateral}$$

$$f(t) = \frac{1}{2.\pi.j} \int_{\sigma-j.\infty}^{\sigma+j.\infty} F(s).e^{s.t}.ds \;;\; \boxed{s = \sigma + j.w}, \text{ T.L. Inversa}$$

Propiedades que debe cumplir la Región de Convergencia (ROC)

- *No contiene polos* (constituyen el límite pues en ellos la función diverge).
  Si $h(t) = 0$ en $t < t_0$ la *señal es derecha y* la ROC resulta de la forma $\sigma > \sigma_0$, con σ_0 parte real del polo que se encuentre más a la derecha.
  Ejemplo:



Nota: Los sistemas Causales tienen Señal $h(t)$ Derecha.
Sin embargo señal derecha no implica Sistema Causal (en el ej. No se cumple)
Repasando: Sistema Causal: Si $h(t) = 0\ para\ t < 0$

Fig. 38 – Ejemplo de Región de convergencia para señales derechas

- Si $h(t) = 0$ en $t > t_0$ *(señal izquierda)* la ROC resulta de la forma $\sigma < \sigma_0$, con σ_0 parte real del polo que se encuentre más a la izquierda

  Ejemplo:

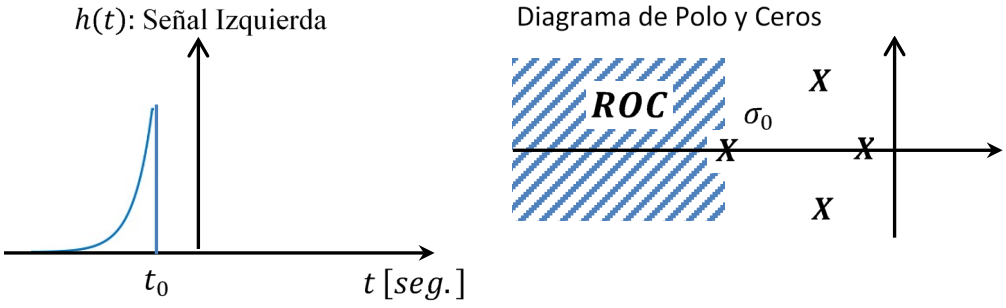


Fig. 39 – Ejemplo de Región de convergencia para señales izquierdas

- Si $h(t)$ *tiene duración finita*, la ROC resulta todo el plano S, exceptuando $\sigma = \sigma_0$ o $\sigma \to \infty$
- Si $h(t)$ *es de duración infinita (bilateral),* la ROC resulta de la forma $\sigma_1 < \sigma < \sigma_0$ (cinta vertical en el plano S)

Algunas ventajas de la Transformada de Laplace

1) Permite transformar un conjunto más grande de funciones (menos limitaciones de convergencia)

2) Permite calcular estados transitorios ya que tiene en cuenta las "condiciones iniciales"

(la transformada de Fourier estudia sistemas solo en régimen permanente)

3) Se toma $s = \sigma + j.w$, simplifica cuentas, utilizo menos números imaginarios. Además, el diagrama de polos y ceros permite un análisis rápido de la función temporal, frecuencial y de la estabilidad del sistema.

Se puede obtener la Transformada de Fourier mediante la Transformada de Laplace con $\sigma = 0$

Si la ROC (región de convergencia) abarca al eje $j.w$, entonces si se toma $\sigma = 0$ y reemplazo $j.w$, mediante la Transformada de Laplace se obtiene la Transformada de Fourier.

$$s|_{\sigma=0} = [\sigma + j.w]_{\sigma=0} \quad \to \quad s|_{\sigma=0} = j.w \quad \to \quad \boxed{H(s)|_{s=j.w} = H(j.w)} \quad \to \quad H_1(w)$$

Con esto se puede graficar la respuesta en frecuencia (modulo y fase de $H(w)$)

Si el eje $j.w$ no pertenece a la ROC, la función no posee Transformada de Fourier, pero sí posee Transformada de Laplace

La Transformada de Laplace tiene numerosas aplicaciones en el campo de la Ingeniería. Una de ellas es: Análisis de Estabilidad de Sistemas

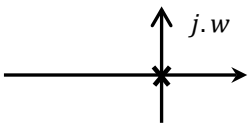
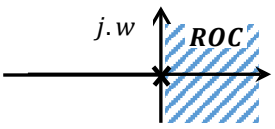
Ejercicio XVII

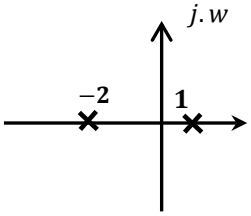
Comparación de sistemas con Transformada de Laplace

Dados los siguientes Diagramas de Polos y Ceros correspondientes a transferencias de sistemas $H(s)$, se pide:

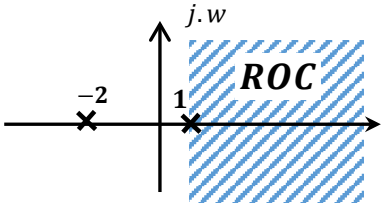
a) Determinar $H(s)$, Región de Convergencia y Estabilidad del Sistema

b) Hallar la respuesta impulsional $h(t)$ sin calcular todas las constantes (colocar letras genéricas para las constantes que falten)

Enunciado	Resolución
 Sistema Causal	 Sistema Causal, Condicionalmente Estable (la ROC está en el límite del eje $j.w$) $u(t) \longleftrightarrow \frac{1}{s} \quad \Re e(s) > 0$ Un polo en el orígen $h(t) = A.u(t) \quad H(s) = A.\frac{1}{s}$



Sistema Causal



No Estable (la ROC no contiene al eje $j.w$)

$H(s) = k.\frac{1}{(s+2).(s-1)}$;

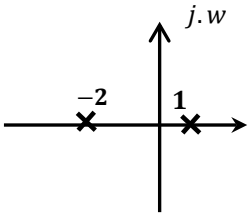
Polos: $s_1 = 1$, $s_2 = -2$

Por Fracciones Simples: $H(s) = k.\frac{1}{(s+2).(s-1)} = k.\left[\frac{A}{s+2} + \frac{B}{s-1}\right]$

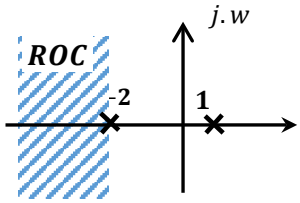
$e^{-a.t}.u(t) \longleftrightarrow \frac{1}{s+a}$; $\quad \Re e(s) > -\Re e(a)$ (tablas)

$h(t) = k.(A.e^{-2.t} + B.e^{t}).u(t)$ (ver ej. 3.2 se calcula A y B con $k = 1$)

Suma de 2 exponenciales



Sistema Anticausal



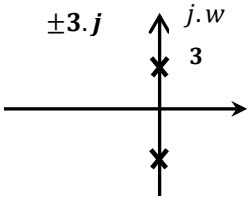
No Estable (la ROC no contiene al eje $j.w$)

$H(s) = k.\frac{1}{(s+2).(s-1)}$; Polos: $s_1 = 1$, $s_2 = -2$

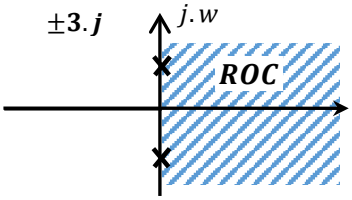
Por Fracciones Simples: $H(s) = k.\frac{1}{(s+2).(s-1)} = k.\left[\frac{A}{s+2} + \frac{B}{s-1}\right]$

$-e^{-a.t}.u(-t) \longleftrightarrow \frac{1}{s+a}$; $\Re e(s) < -\Re e(a)$

$h(t) = k.(-).(A.e^{-2.t} + B.e^{t}).u(-t)$ Suma de 2 exponenciales



Sistema Causal



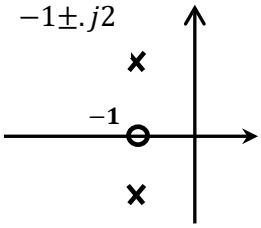
Sistema Causal, Condicionalmente Estable (la ROC está en el límite del eje $j.w$)

$\text{sen}(w_0.t).u(t) \longleftrightarrow \frac{w_0}{s^2+{w_0}^2}$; $\Re e(s) > 0$

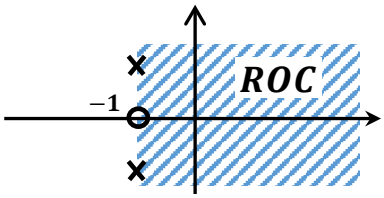
(tablas) Dos polos complejos Conjugados ubicados en el eje $j.w$. No tengo ceros.

$H(s) = A.\frac{3}{s^2+3^2}$; $w_0 = 3$

$h(t) = \text{sen}(3.t).u(t)$



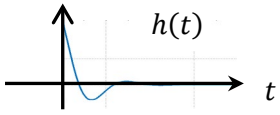
Sistema Causal



Del Diagrama de Polos y Ceros

$z_0 = -1$

$z_{1,2} = -1 \pm .j2$

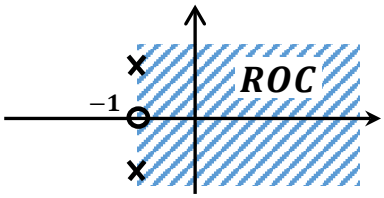


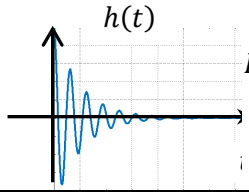
Sistema Causal. Estable (la ROC no contiene al eje $j.w$)

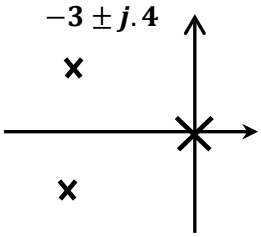
$e^{-a.t}.\cos(w_0.t).u(t) \longleftrightarrow \frac{s+a}{(s+a)^2+{w_0}^2}$ (tablas) Contiene 1 cero y 2 polos complejos conjugados

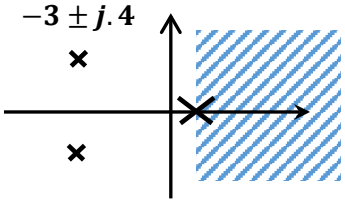
$h(t) = e^{-t}.\cos(2.t).u(t) \longleftrightarrow H(s) = \frac{s+1}{(s+1)^2+2^2}$ $F(s)$ tiene los polos $z_{1,2}$

Con los polos $z_{1,2} = -1 \pm .j2$ obtengo: $\begin{cases} e^{-t} \\ \cos(2.t) \end{cases}$

<table>
<tr>
<td>Polos: $z_{1,2} = -1 \pm .j.10$
Ceros: $z_0 = -1$
Sistema Causal</td>
<td> 

$Del\ diagrama\ de\ Polos\ y\ Ceros$:
$z_0 = -1$
$z_{1,2} = -1 \pm .j10$
 

$h(t) = e^{-t}.\cos(10.t).u(t) \longleftrightarrow H(s) = \frac{s+1}{(s+1)^2+10^2}$ $H(s)$ tiene los polos $z_{1,2}$</td>
</tr>
<tr>
<td> 

Sistema Causal</td>
<td> 

Sistema Causal, Condicionalmente Estable (la ROC está en el límite del eje $j.w$)
$e^{-a.t}.\text{sen}(w_0.t).u(t) \longleftrightarrow \frac{w_0}{(s+a)^2+{w_0}^2}$
(tablas)
$u(t) \longleftrightarrow \frac{1}{s}$; $\Re e(s) > 0$ (tablas) Un polo en el orígen
$H(s) = \frac{1}{s} + \frac{4}{(s+3)^2+4^2}$; en este ejemplo a $= -3$ y $w_0 = 4$
$h(t) = (A + B.e^{-3.t}.\text{sen}(4.t)).u(t)$</td>
</tr>
</table>

Transformada Z

Definimos la transformada Z (T.Z.) bilateral, unilateral e inversa

Transformada Z Bilateral (T.Z.B.) : $\boxed{F_b(z) = \sum_{n=-\infty}^{\infty} f[n].z^{-n}}$

Transformada Z Unilateral (T.Z.U.): $\boxed{F(z) = \sum_{n=0}^{\infty} f[n].z^{-n}}$

Transformada Z Inversa: $\boxed{f[n] = \frac{1}{2.\pi.j}.\int_C F(z).z^{-n}.dz \quad ; \quad n \in Z}$

Filtros FIR e IIR

Filtros FIR <u>Respuesta Finita al Impulso (FIR: Finite Impulse Response)</u>	**Filtros IIR** <u>Respuesta infinita al impulso (IIR: Infinite Impulse Response)</u>
La salida se obtiene solamente con las entradas actuales y anteriores. Los Filtros FIR resultan No recurrentes (No realimento con la salida). Los IIR si lo son.	La salida del filtro depende de las entradas actuales y pasadas, y también de las salidas en instantes anteriores. Esto se obtiene realimentando la salida. Los Filtros IIR son recurrentes (se realimentan con la salida)

$$\boxed{y[n] = \sum_{k=0}^{N-1} b_k . x[n-k]}$$ $N-1$ es el orden del filtro, coincide con la cantidad de términos no nulos y con la cantidad de coeficientes del filtro b_k	$$\boxed{y[n] = \sum_{k=0}^{N} b_k . x[n-k] - \sum_{i=1}^{M} a_i . y[n-i]}$$ $y[n] = b_0 . x[n] + b_1 . x[n-1] + \ +b_N . x[n-N] \ - a_1 . y[n-1] - a_2 . y[n-2] - \ \ldots\ldots\ldots\ldots - a_M . x[n-M]$ a y b representa los coeficientes del filtro. El orden es el valor máximo entre los valores de M y N. Donde Mdetermina la cantidad de polos y N determina la cantidad de ceros de la función de transferencia.
La salida se puede expresar como la Convolución entre la señal de entrada $x[n]$ y la respuesta impulsional $h[n]$: $$y[n] = \sum_{k=0}^{N-1} h_k . x[n-k]$$ para respuesta impulsional ($x[n] = \delta[n]$) $$h[n] = \sum_{k=0}^{N-1} h_k . \delta[n-k]$$ Viendo $h[n]$, tiene "cantidad finita de impulsos" (respuesta impulsiva finita) Transformando la expresión anterior: $$\boxed{H(z) = \sum_{k=0}^{N-1} h_k . z^{-k}}$$ $H(z) = h_0 + h_1 . z^{-1} + \ \ldots\ldots\ldots . + h_{N-1} . z^{-(N-1)}$	Para respuesta impulsional ($x[n] = \delta[n]$), La salida $y[n]$ pasa a ser $h[n]$ $$h[n] = \sum_{k=0}^{N} b_k . \delta[n-k] - \sum_{i=1}^{M} a_i . h[n-i]$$ Aplicando la Transformada Z a $h[n]$ y operando obtenemos: $$\boxed{H(z) = \frac{\sum_{k=0}^{N} b_k . z^{-k}}{1 + \sum_{k=1}^{M} a_k . z^{-k}}}$$ Los IIR tienen $h[n]$ con infinitos términos: respuesta impulsiva infinita La expresión $H(z)$presenta denominador con polos fuera del origen
Viendo la expresión anterior: <u>Los filtros FIR tienen todos sus polos en el origen, siempre son estables.</u> Si se quiere tener fase lineal se diseña el filtro con sus ceros en pares recíprocos.	<u>Presenta polos fuera del origen.</u> También pueden tener polos en el origen. Pueden ser Inestables Mediante los polos y ceros se determina la estabilidad y la causalidad del filtro. Si todos los ceros y polos del filtro están en el interior de la circunferencia unitaria, resulta de fase mínima, y el sistema es estable y causal. Si todos sus ceros se encuentran en el exterior es de fase máxima. Si tiene uno o más polos fuera de la circunferencia unitaria, entonces el sistema es inestable.

Estructura

Existen muchas maneras de implementar los filtros FIR e IIR. La estructura influye en las características del filtro. Al diseñar filtros, se debe tener en cuenta el gasto computacional

Estructura del Filtro FIR

$y[n] = \sum_{k=0}^{N-1} b_k . x[n-k]$

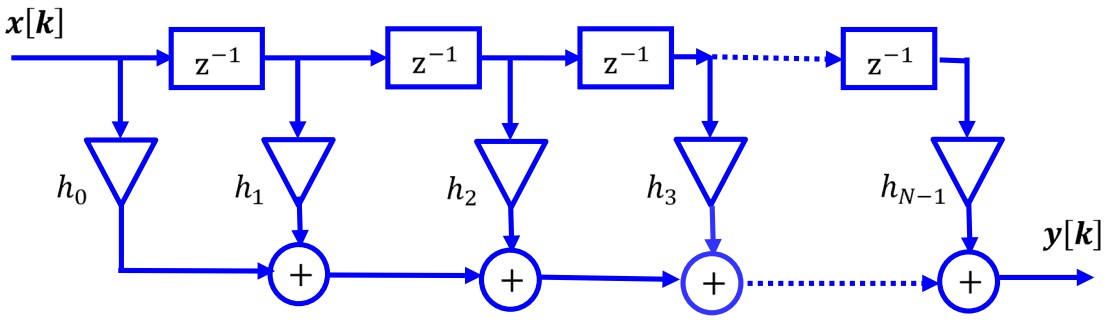


Fig. 40 – Estructura del Filtro FIR con h

En la figura los términos $h[n]$ son los coeficientes y los z^{-1} son los retardos de tiempo T. Existen distintas variantes para esta estructura, se pueden construir interconectando filtros en serie, en cascada, etc.

Otra estructura básica de un Filtro FIR: $y[n] = \sum_{k=0}^{N-1} b_k . x[n-k]$

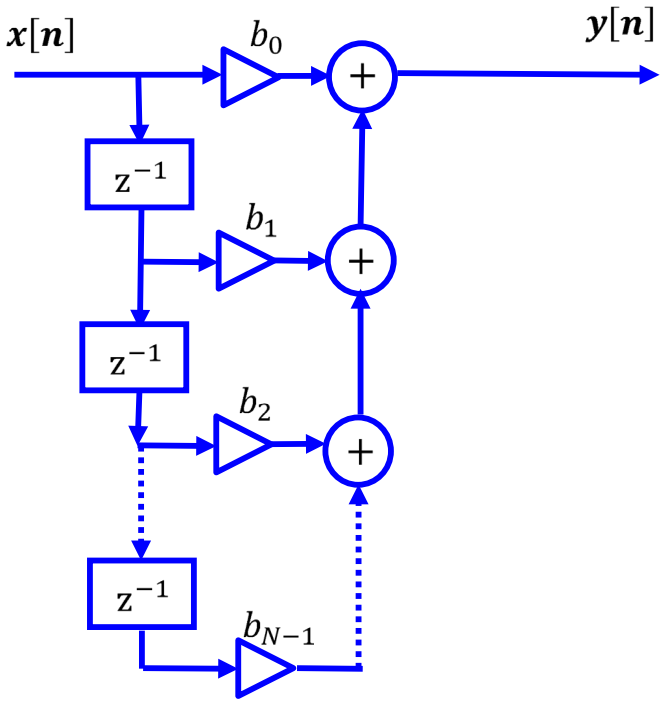


Fig. 41 – Estructura del Filtro FIR

Estructura del Filtro IIR (se agrega la realimentación de la salida)

$y[n] = \sum_{k=0}^{N} b_k . x[n-k] - \sum_{i=1}^{M} a_i . y[n-i]$

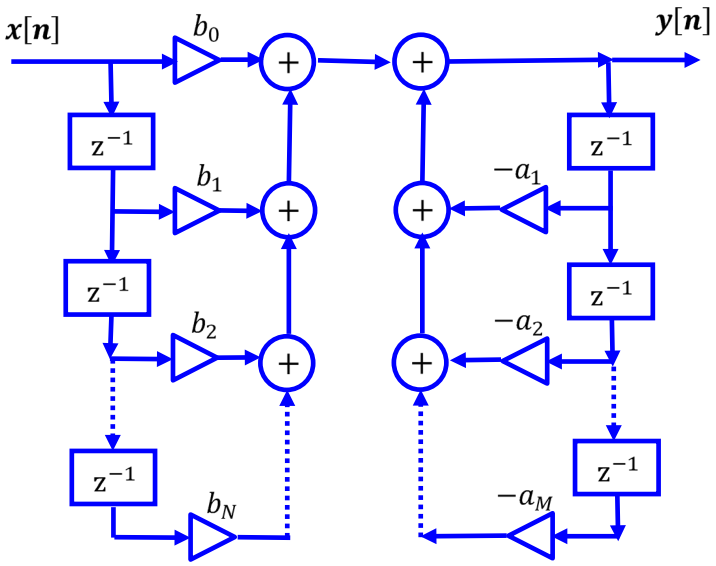


Fig. 42 – Estructura del filtro IIR

Ejercicio XVIII

Análisis de filtros con Transformada Z

Calcular la respuesta en frecuencia (modulo y fase) utilizando el método más conveniente

$y[n] = \frac{1}{2} . (x[n] + x[n-1])$

Resolución:

$Y(z) = \frac{1}{2} . X(z)(1 + z^{-1})$

$H(z) = \frac{Y(z)}{X(z)} = \frac{1}{2} + \frac{1}{2} . z^{-1} = \frac{1+z^{-1}}{2}$; $H(z) = \frac{z+1}{2.z}$

Reemplazamos $\boxed{z = e^{j.\Omega}}$ en $H(z) = \frac{1+z^{-1}}{2}$; $\boxed{H(e^{j.\Omega}) = \frac{1+e^{-j.\Omega}}{2}}$

Método Gráfico:

$$\boxed{|H(\Omega)| = \frac{\prod(\text{distancia de ceros a } \Omega)}{\prod(\text{distancia de polos a } \Omega)} = \frac{|z_{cero\ 1} - z_\Omega| . |z_{cero\ 2} - z_\Omega| . \dots \dots |z_{cero\ n} - z_\Omega|}{|z_{polo\ 1} - z_\Omega| . |z_{polo\ 2} - z_\Omega| \dots \dots \dots \dots \dots |z_{polo\ m} - z_\Omega|}}$$

$\prod$: Productoria ; Ω o z_Ω : es el punto analizado

$Fase\{H(\Omega)\} =< H(\Omega)$

$$\boxed{< H(\Omega) = \sum (Fases\ de\ \text{ceros a } \Omega) - \sum (Fases\ de\ \text{polos a } \Omega)}$$

$< H(\Omega) = Fase(z_{cero\ 1} - z_\Omega) + Fase(z_{cero\ 2} - z_\Omega) + \dots \dots + Fase(z_{cero\ n} - z_\Omega) -$
$[Fase(z_{polo\ 1} - z_\Omega) + Fase(z_{polo\ 2} - z_\Omega) + \dots \dots + Fase(z_{polo\ m} - z_\Omega)]$

$\boxed{H(z) = \frac{1}{2} . \frac{z+1}{z}}$; Cero en $z = -1$, Polo en $z = 0$

Método Gráfico			
Ω	$z = e^{j.\Omega}$	$\lvert H(\Omega)\rvert = \left(\frac{1}{2}\right).\frac{\prod(\text{distancia de ceros a }\Omega)}{\prod(\text{distancia de polos a }\Omega)}$	$< H(\Omega)$
0	$e^{j.0} = 1$	$\left(\frac{1}{2}\right).\frac{2}{1} = 1$	$0 - 0 = 0$
$\frac{\pi}{2}$	$e^{j.\frac{\pi}{2}} = j$	$\left(\frac{1}{2}\right).\frac{\sqrt{2}}{1} = \frac{\sqrt{2}}{2}$	$\frac{\pi}{4} - \frac{\pi}{2} = -\frac{\pi}{4}$
π	$e^{j.\pi} = -1$	$\left(\frac{1}{2}\right).\frac{0}{1} = 0$	Tomamos: $\Omega \to 179°$ $\cong \frac{\pi}{2} - \pi = -\frac{\pi}{2}$

Método Analítico				
Ω	$z = e^{j.\Omega}$	$H(z) = \frac{1}{2}.\frac{z+1}{z}$	$\lvert H(z)\rvert$	$< H(z)$
0	$e^{j.0} = 1$	$\frac{1}{2}.\frac{1+1}{1}=1$	$\lvert 1\rvert = 1$	0
$\frac{\pi}{2}$	$e^{j.\frac{\pi}{2}} = j$	$\frac{j+1}{2.j} = \frac{1}{2} - \frac{1}{2}.j$	$\frac{\sqrt{2}}{2}$	$-\frac{\pi}{4}$
π	$e^{j.\pi} = -1$	$\frac{0}{-2} = 0$	0	....
$0{,}99.\pi$	$e^{j.0{,}99.\pi} = -0.9995 + 0.0314i$	$\frac{1}{2}.\frac{0.0005 + 0.0314i}{-0.9995 + 0.0314i}$	$\cong 0$	$\cong \frac{\pi}{2} - \pi = -\frac{\pi}{2}$

Graficamos usando la Tabla. La señal es Periódica con período $2.\pi$

$\lvert H(z)\rvert$ Par ; $< H(z)$ Impar

Los resultados de respuesta en frecuencia se muestran en la Fig. 43

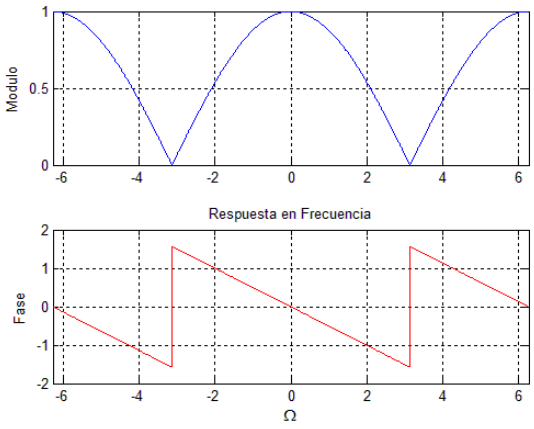


Fig. 43 – Respuesta en frecuencia de un filtro simple pasa bajos

Resolviendo en Matlab®

Se obtiene el gráfico anterior, Diagrama de Polos y ceros y Respuesta al Impulso

```
w= -2*pi: 0.01 :  2*pi ;
H=1/2* (1+exp(-j*w)) ;
figure(1) ;  subplot(2,1,1) ;plot(w, abs(H)) ;
grid on; xlim( [-2*pi , 2*pi] ); ylabel('Modulo') ; ; title('Respuesta en Frecuencia')
subplot(2,1,2)  ;plot(w, angle(H), 'r') ; grid on;  xlim( [-2*pi , 2*pi] );
ylabel('Fase') ; xlabel('\Omega')
Num=[1 1];   Den=[2 0];
%Función transferencia (igual que en Laplace, pero con Ts=1)
H=tf(Num,Den,1)
p= pole(H) ; z= zero(H) ; r= residue(Num,Den) %Polos, ceros y residuos
figure(2); zplane(Num,Den); %Diagrama de Polos y Ceros
figure(3); freqz(Num,Den); %Respuesta en frecuencia
figure(4);   n=0:1:20; impulse(H,n);  %Respuesta al impulso
```

Ejercicio XIX

Análisis de sistemas con Transformada Z

Un sistema de segundo orden inestable posee la siguiente función transferencia: $H(z) = \frac{1}{1-z^{-1}+1{,}44.z^{-2}}$

Se lo desea estabilizar colocándolo en el sistema realimentado de la Fig. 44.

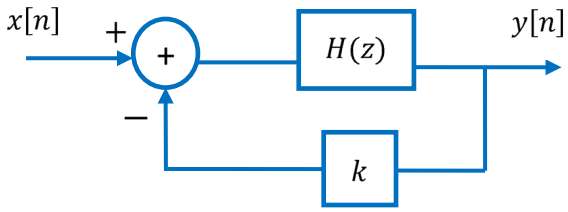


Fig. 44 – Sistema realimentado

Se pide:

a) Encontrar la función transferencia del nuevo sistema

b) Hallar el valor de k que asegure la estabilidad del sistema

Resolución:

a) Encontrar la función transferencia del nuevo sistema

A la salida del sumador asignamos la variable $M(z)$

$M(z) = X(z) - Y(z).k \quad ; \quad Y(z) = M(z).H(z)$

Despejamos: $M(z) = \frac{Y(z)}{H(z)} = X(z) - Y(z).k \quad \rightarrow \quad Y(z)\left[\frac{1}{H(z)} + k\right] = X(z)$

$Y(z)\left[\frac{1+k.H(z)}{H(z)}\right] = X(z) \quad ; \quad \boxed{H_T(z) = \frac{Y(z)}{X(z)} = \frac{H(z)}{1+k.H(z)}}$

Reemplazamos $H(z)$ en $H_T(z)$; Siendo $H(z) = \frac{1}{1-z^{-1}+1{,}44.z^{-2}} = \frac{z^2}{z^2-z+1{,}44}$

$H_T(z) = \frac{\frac{z^2}{z^2-z+1{,}44}}{1+k.\frac{z^2}{z^2-z+1{,}44}} = \frac{\frac{z^2}{z^2-z+1{,}44}}{\frac{z^2-z+1{,}44+k.z^2}{z^2-z+1{,}44}} = \frac{z^2}{z^2-z+1{,}44+k.z^2} \quad ; \quad \boxed{H_T(z) = \frac{z^2}{z^2.(k+1)-z+1{,}44}}$

b) Hallar el valor de k que asegure la estabilidad del sistema

Resolución:

$z_{1,2} = \frac{1\pm\sqrt{1-4x1{,}44.(k+1)}}{2.(k+1)} = \frac{1\pm\sqrt{1-5{,}76.(k+1)}}{2.(k+1)}$

$z_{1,2} = \frac{1\pm\sqrt{-5{,}76.k-4{,}76}}{2.(k+1)}$

Para tener Raíces complejas Conjugadas, se debe cumplir $(-5{,}76.k - 4{,}76) < 0$

$-4{,}76 < 5{,}76.k \; ; \quad k > -0{,}82638$

Suponemos $k > 0$, sino sería realimentación positiva

Con $k > 0$ siempre tenemos Raíces complejas Conjugadas

$\sqrt{-1} = \pm j; \qquad z_{1,2} = \frac{1}{2.(k+1)} \pm j.\frac{\sqrt{5{,}76.k+4{,}76}}{2.(k+1)}$

Para un Sistema Causal, los polos deben estar dentro de la Circunferencia Unitaria: $|z_{1,2}| < 1$

$\rightarrow \quad |z_{1,2}| = \sqrt{\frac{1}{4.(k+1)^2} + \frac{5{,}76.k+4{,}76}{4.(k+1)^2}} \quad < 1$

$\rightarrow \quad \frac{5{,}76.k+5{,}76}{4.(k+1)^2} < 1 \qquad ; \; 5{,}76.k + 5{,}76 < 4.(k+1)^2 \qquad ; \; 5{,}76.(k+1) < 4.(k+1)^2$

$5{,}76 < 4.(k+1) \quad ; \quad \frac{5{,}76}{4} - 1 < k \quad ;$ Sistema Estable para $\boxed{k > 0{,}44}$

Ejercicio XX

Análisis de filtros con Transformada Z

Calcular y graficar la respuesta en frecuencia (modulo y fase) del siguiente filtro:

$y[n] = \frac{1}{3}(x[n] + x[n-1] + x[n-2])$

Respuesta: $H(z) = \frac{1}{3}\left(\frac{z^2+z+1}{z^2}\right)$; $\Omega_{ceros} = \pm 2{,}094\,.k$ con $k = 1{,}2{,}3 \ldots\ldots$

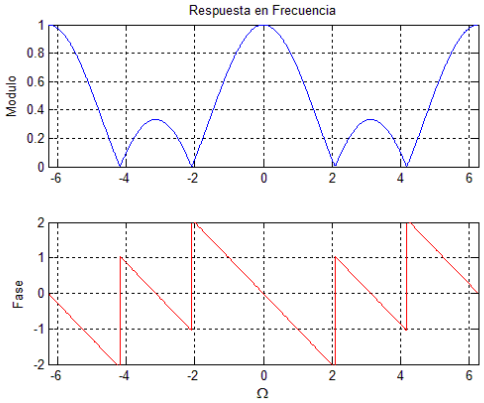


Fig. 45 – Respuesta en frecuencia de un filtro pasa bajos simple

Ejercicio XXI

Análisis de filtros con Transformada Z

Calcular y graficar la respuesta en frecuencia (modulo y fase) del siguiente filtro:

$$y[n] = \frac{1}{3}(x[n+1] + x[n] + x[n-1])$$

Resolución:

Transformamos la ecuación en diferencias:

$$f[n \pm a] \longleftrightarrow z^{\pm a}.F(z) \quad \text{; para CIN (Condiciones Iniciales Nulas) (tablas)}$$

$$Y(z) = \frac{1}{3}.X(z)(z+1+z^{-1}) \quad ; \quad H(z) = \frac{Y(z)}{X(z)} = \frac{1}{3}.(z+1+z^{-1}).\frac{z}{z} \quad ; \boxed{H(z) = \frac{1}{3}\left(\frac{z^2+z+1}{z}\right)}$$

Reemplazamos $\boxed{z = e^{j.\Omega}}$ → $H\left(e^{j.\Omega}\right) = \frac{1}{3}\left(\frac{e^{j.\Omega.2}+e^{j.\Omega}+1}{e^{j.\Omega}}\right)$

Buscamos los ceros de $H(z)$ que son las raíces del numerador $z^2 + z + 1 = 0$

En Matlab®: roots([1 1 1]) ans = -0.5000 + 0.8660i -0.5000 - 0.8660i

Calculamos módulo y fase para expresar en forma polar:

$$z_{1,2} = -0{,}5 \pm j.\frac{\sqrt{3}}{2} = \sqrt{\frac{1}{4}+\frac{3}{4}}.e^{j.\frac{2}{3}.\pi} = e^{\pm j.\frac{2}{3}.\pi} = \begin{cases} e^{j.\frac{2}{3}.\pi} \\ e^{-j.\frac{2}{3}.\pi} = e^{j.\frac{4}{3}.\pi} \end{cases}$$

$$\rightarrow z_{1,2} = e^{j.\Omega} = \begin{cases} e^{j.\frac{2}{3}.\pi} \\ e^{j.\frac{4}{3}.\pi} \end{cases} \quad \text{Aplicamos logaritmo} \rightarrow \quad \Omega_{Ceros\ 1,2} = \begin{cases} \frac{2}{3}.\pi = 2{,}094 \\ \frac{4}{3}.\pi = 4{,}19 \end{cases}$$

Como el Módulo del espectro es par: $\boxed{\Omega_{Ceros\ 1,2} = \begin{cases} \pm 2{,}094 \\ \pm 4{,}19 \end{cases}}$

Método Gráfico:

$$\boxed{|H(\Omega)| = \frac{\prod(\text{distancia de ceros a } \Omega)}{\prod(\text{distancia de polos a } \Omega)} = \frac{|z_{cero\ 1} - z_\Omega|.|z_{cero\ 2} - z_\Omega|. \dots \dots \dots |z_{cero\ n} - z_\Omega|}{|z_{polo\ 1} - z_\Omega|.|z_{polo\ 2} - z_\Omega| \dots \dots \dots \dots \dots \dots |z_{polo\ m} - z_\Omega|}}$$

$\prod$: Productoria ; Ω o z_Ω : es el punto analizado

$Fase\{H(\Omega)\} =< H(\Omega)$

$$\boxed{< H(\Omega) = \sum(Fases\ de\ \text{ceros a}\ \Omega\ ) - \sum(Fases\ de\ \text{polos a}\ \Omega\ )}$$

$< H(\Omega) = Fase(z_{cero\ 1} - z_\Omega) + Fase(z_{cero\ 2} - z_\Omega) + \cdots \ldots + Fase(z_{cero\ n} - z_\Omega) \ - \ \left[Fase(z_{polo\ 1} - z_\Omega) + Fase(z_{polo\ 2} - z_\Omega) + \cdots \ldots + Fase(z_{polo\ m} - z_\Omega) \right]$

$H(z) = \frac{1}{3}\left(\frac{z^2+z+1}{z}\right)$; Ceros: $z_{1,2} = -0{,}5\ \pm j.\frac{\sqrt{3}}{2}$; Polo en $z = 0$

Método Gráfico			
Ω	$z = e^{j.\Omega}$	$\lvert H(\Omega)\rvert = \left(\frac{1}{2}\right).\frac{\prod(\text{distancia de ceros a }\Omega)}{\prod(\text{distancia de polos a }\Omega)}$	$< H(\Omega)$
0	$e^{j.0} = 1$	$\frac{1}{3}.\frac{\sqrt{1{,}5^2 + \left(\frac{\sqrt{3}}{2}\right)^2}.\sqrt{1{,}5^2 + \left(\frac{\sqrt{3}}{2}\right)^2}}{1} = 1$	$-30° + 30° - (0°) = 30°$
$\frac{\pi}{2}$	$e^{j.\frac{\pi}{2}} = j$	Realizamos cuentas con Matlab®: 1/3 * sqrt((1-0.866)^2+0.5^2) * sqrt((1+0.866)^2+0.5^2) / 1 = 1/3	atand((1-0.866) /0.5)+atand((1+ 0.866)/0.5) - 90= 0
π	$e^{j.\pi} = -1$		
$\pm 2{,}094$	$-0{,}5\ \pm j.\frac{\sqrt{3}}{2}$	0	
$\pm 4{,}19$	$-0{,}5\ \pm j.\frac{\sqrt{3}}{2}$	0	

Método Analítico				
Ω	$z = e^{j.\Omega}$	$H(z) = \frac{1}{3}\left(\frac{z^2+z+1}{z}\right)$	$\lvert H(z)\rvert$	$< H(z)$
0	$e^{j.0} = 1$	$\frac{1}{3}.\frac{3}{1} = 1$	1	0
$\frac{\pi}{2}$	$e^{j.\frac{\pi}{2}} = j$	$\frac{1}{3}.\left(\frac{-1+j+1}{j}\right) = \frac{1}{3}$	$\frac{1}{3}$	0
π	$e^{j.\pi} = -1$	$\frac{1}{3}.\left(\frac{1-1+1}{-1}\right) = -\frac{1}{3}$	$\frac{1}{3}$	π
$\pm 2{,}094$ $\pm 4{,}19$	$-0{,}5\ \pm j.\frac{\sqrt{3}}{2}$	0	0	

Graficamos usando la Tabla. La señal es Periódica con período $2.\pi$

$|H(z)|$ Par ; $< H(z)$ Impar

```
w= -2*pi: 0.01 :  2*pi ;
H=1/3* (exp(j*w)+ 1+ exp(-j*w)) ;
figure(1) ;  subplot(2,1,1)  ;plot(w, abs(H)) ;
grid on; xlim( [-2*pi , 2*pi] );
ylabel('Modulo') ; ; title('Respuesta en Frecuencia')
subplot(2,1,2)  ;plot(w, angle(H), 'r') ;
grid on;  xlim( [-2*pi , 2*pi] );
ylabel('Fase') ; xlabel('\Omega') ;  %ylim( [-2 , 2] );
```

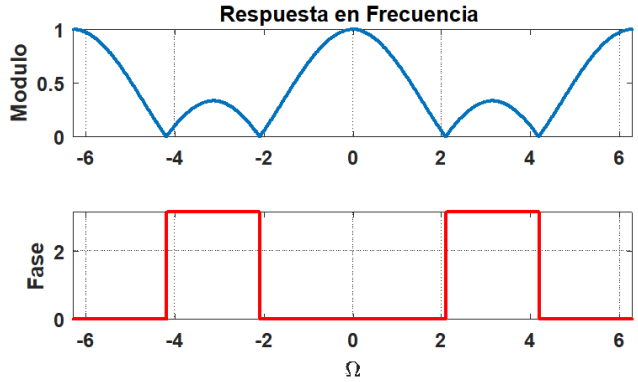


Fig. 46 – Respuesta en frecuencia del filtro

Ejercicio XXII

Ejemplo de un ecualizador implementado en Matlab®

```
%% Ejemplo de Ecualizador
%% Utilizamos la función fir1 (filtro FIR)
% Importamos la señal de audio
   [x,fs] = audioread('sample.wav') ;%  Debe encontrarse este archivo .wav en directorio actual
   %[x,fs]=wavread('sample.wav'); % para versiones viejas de Matlab®
N = 62; % Cantidad de polos
% Filtro pasa bajo
   wc1 = .20;  % W de corte normalizada entre 0 y 1
   LP = fir1(N,wc1);   % por defecto es pasa bajo
   y1 = conv(LP,x); % señal filtrada con filtro pasa bajo
% Filtro pasa banda
   wc2 = [.20, .50];  % W de corte normalizada entre 0 y 1 para filtro pasa banda
```

```
BP = fir1(N,wc1);
y2 = conv(BP,x); % señal filtrada pasa banda
% Filtro pasa alto
wc3 = .50;  % W de corte normalizada entre 0 y 1
HP = fir1(N,wc3,'high'); % pasa alto
y3 = conv(HP,x); % señal filtrada pasa alto
% Graficamos la respuesta en frecuencia de cada filtro
figure(1); freqz(LP);
figure(2); freqz(BP);
figure(3); freqz(HP);
% Definimos las ganancia para cada banda (se modifican las amplitudes, es decir el volumen)
ganancia1 = 0.4;
ganancia2 = 1.5;
ganancia3 = 1.5;
y_f_baja = ganancia1 * y1;  % pasa bajo
%wavwrite(y_f_baja,fs,'Equalizer1');
y_f_media= ganancia2 * y2;  % pasa banda
%wavwrite(y_f_media, fs,'Equalizer2');
y_f_alta= ganancia3 * y3;  % pasa alto
%wavwrite(y_f_alta,fs,'Equalizer3');
% Sumamos la señal de cada banda para obtener “Respuesta Ecualizada”
y_ecualizada = y_f_baja  +  y_f_media +   y_f_alta;
%wavwrite(y_ecualizada,fs,'Equalizer4');
% Reproducimos en el parlante las señales obtenidas
sound( x , fs ) ; % señal original
disp('señal original - presione una tecla'); pause;
sound( 0.001* x , fs ) ;
disp('presionar una tecla'); pause;
sound( y_f_baja , fs ) ; % pasa bajo
disp('presionar una tecla'); pause;
sound( y_f_media , fs ) ; % pasa banda
disp('presionar una tecla'); pause;
sound( y_f_alta , fs ) ; % pasa bajo
disp('presionar una tecla'); pause;
sound( y_ecualizada , fs ) ; % SUMATORIA
```

Ejercicio XXIII

Análisis de filtros con Transformada Z

Se dispone de un sistema embebido para adquirir datos provenientes de señales de audio. Este sistema utiliza una frecuencia de muestreo de 20 KHz. También se disponen de 4 filtros, cada uno tiene 2 polos

Filtro 1: $z_{1,2} = 0$, Filtro 2: $z_{1,2} = \pm 1/2$, Filtro 3: $z_{1,2} = \pm 0.98$ y Filtro 4: $z_{1,2} = \pm 3/2$

Se pide:

a) Para cada uno de los 4 filtros:

- agregar *dos ceros* de modo de eliminar las frecuencias 0 Hz y 10 KHz.
- Graficar Diagrama de polos y ceros. Determinar $H(z)$
- ¿Qué tipo de filtro resulta? Analizar Estabilidad

Solo para el Filtro 1 obtener su respuesta impulsional $h_1[n]$.

Resolución a)

$f_{muestreo} = f_s = 20\ KHz$

La frecuencia máxima es $\Omega = \pi$ correspondiente a $\frac{f_s}{2} = 10\ KHz$

→ Para eliminar frecuencias 0 Hz y 10 KHz, colocamos ceros en $\Omega = 0$ y $\Omega = \pi$

$z_{ceros} = \begin{cases} e^{j.0} = 1 \\ e^{j.\pi} = -1 \end{cases}$

$H_1(z) = \frac{(z-1).(z+1)}{z^2} = \frac{z^2-1}{z^2}$; FIR Estable

$H_2(z) = \frac{(z-1).(z+1)}{\left(z-\frac{1}{2}\right).(z+1/2)} = \frac{z^2-1}{z^2-1/4}$; IIR Estable

$H_3(z) = \frac{(z-1).(z+1)}{(z-0.98).(z+0.98)} = \frac{z^2-1}{z^2-0.98^2}$; IIR Estable

$H_4(z) = \frac{z^2-1}{\left(z-\frac{3}{2}\right).(z+3/2)}$; IIR Inestable ¡!! La ROC no contiene la CRU $|z| = 1$

En la Fig. 47 se muestran los diagramas de polos y ceros de los 4 filtros.

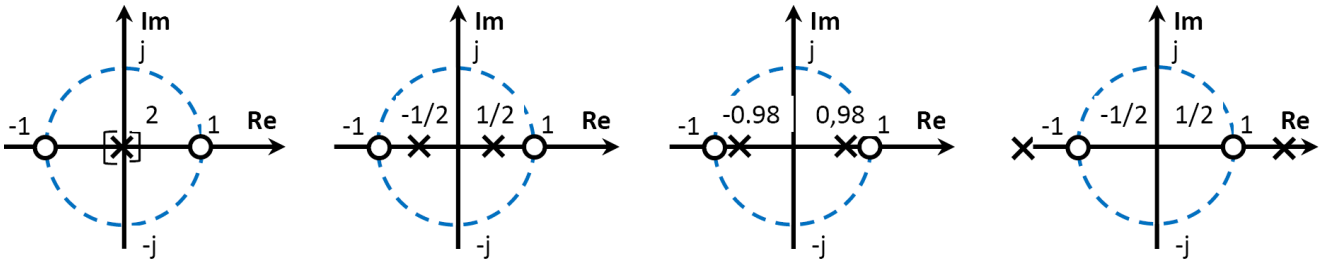


Fig. 47 – Diagrama de polos y ceros de 4 filtros distintos

$H(z) = \frac{(z-1).(z+1)}{z^2} = \frac{z^2+z-z-1}{z^2} = \frac{z^2-1}{z^2} = 1 - z^{-2}$

Antitransformando $\rightarrow h_1[n] = \delta[n] - \delta[n-2]$

b) **Opcional:** Mediante Matlab® graficar el **módulo** de su respuesta en frecuencia

Resolución b)

```
wn= -pi: 0.001 : pi ;

f= wn/(2*pi) *20000 ;  % fs=20000 Hz

z= exp(j*wn) ;

H1= (z.^2 -1)  ./ z.^2 ;  % FIR con polos en el origen

H2= (z.^2 -1)  ./ (z.^2-1/4)  ;

H3= (z.^2 -1)  ./ ( (z-0.98).*(z+0.98))  ;

H4= (z.^2 -1)  ./ ( (z-1.5).*(z+1.5))  ;

figure ;  subplot(2,1,1)  ;

plot(f, abs(H1),'b',  'linewidth', 3) ; hold on

plot(f, abs(H2),'r',  'linewidth', 3)  ;

plot(f, abs(H3),'g',  'linewidth', 3)  ;

plot(f, abs(H4),'y',  'linewidth', 3)  ;

grid on; ylabel('Modulo')

legend('Filtro 1','Filtro 2','Filtro 3','Filtro 4')

title('Respuesta en Frecuencia')

subplot(2,1,2)

plot(f, angle(H1), 'b',  'linewidth', 3) ; hold on

plot(f, angle(H2), 'r',  'linewidth', 3) ;

plot(f, angle(H3), 'g',  'linewidth', 3) ;

plot(f, angle(H4), 'y',  'linewidth', 3) ;

grid on

ylabel('Fase') ; xlabel('Hz')
```

c) En el punto anterior se obtiene el gráfico de la Fig. 48 con la respuesta en frecuencia de los 4 filtros

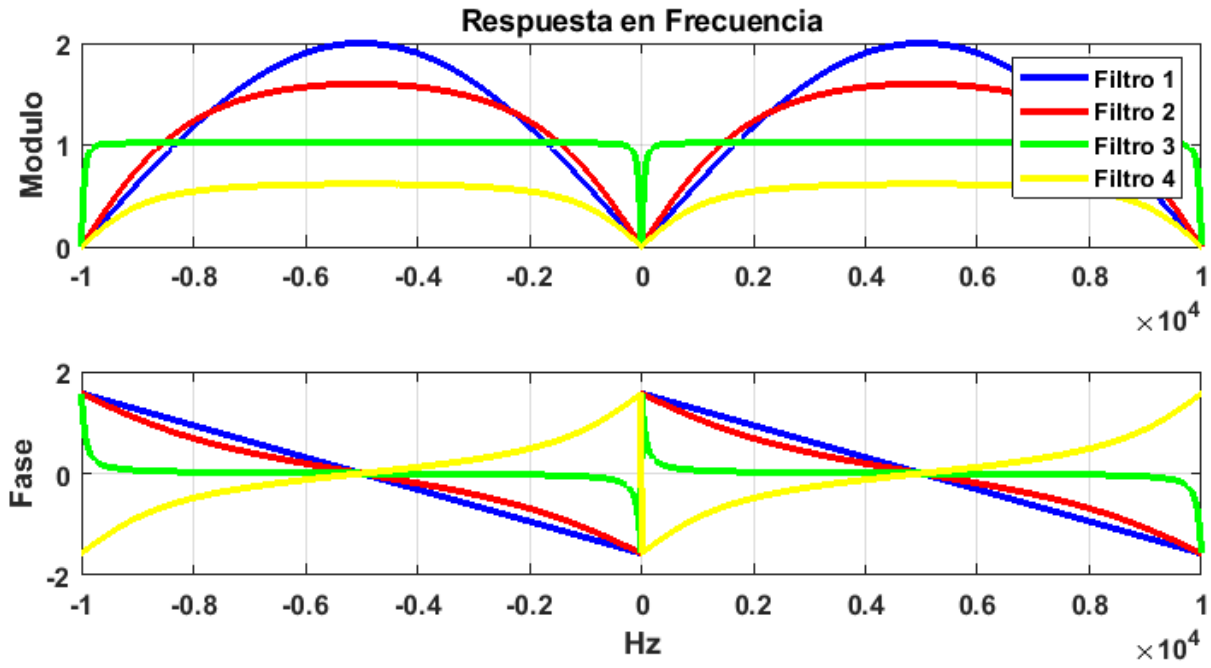


Fig. 48 – Respuesta en frecuencia de los 4 filtros analizados

Teniendo en cuenta que se requiere una respuesta lo más plana posible (ganancia constante) y respuesta Estable, seleccionar el mejor filtro. Recuerde que solo debe filtrar las frecuencias 0 Hz y 10 KHz

¿Qué frecuencia (en Hz) se ve menos afectada en los 4 filtros?

Justificar

Resolución c)

El filtro 4 No es Estable

Seleccionamos Filtro 3 ya que tiene respuesta plana y es Estable IIR

Frecuencia menos afectada: $f = 5\ KHz$

d) Para el filtro seleccionado, Analíticamente obtener el módulo de la respuesta en frecuencia para 0 KHz, para 5 KHz y para 10 KHz. Verificar comportamiento del filtro.

Resolución d)

Ω	f	$z = e^{j.\Omega}$	$H_3(z) = \frac{z^2 - 1}{z^2 - 0.98^2}$	$\lvert H(z)\rvert$	$< H(z)$
0	$0\ Hz$	$e^{j.0} = 1$	0	0	0
$\frac{\pi}{2}$	$5\ KHz$	$e^{j.\frac{\pi}{2}} = j$	$\frac{-1-1}{-1-0.98^2} = \frac{-2}{-1{,}96} = 1.02$	1.02	0
π	$10\ KHz$	$e^{j.\pi} = -1$	$\frac{1-1}{z^2 - 0.98^2} = 0$	0	0

e) Para el filtro 1, modificar la posición de los ceros de manera de obtener filtro que elimine solamente la frecuencia de 5 KHz. Obtener $H_1(z)$ Modificado , Analizar Estabilidad, tipo de filtro y Graficar Diagrama de Polos y Ceros

Resolución e)

Sistema Original: $H_1(z) = \frac{(z-1).(z+1)}{z^2}$; Modifico los ceros

La frecuencia máxima es $\Omega = \pi$ correspondiente a $\frac{f_s}{2} = 10\ KHz$

→ Para eliminar frecuencias 5 KHz, colocamos ceros en $\Omega = \pm\pi/2$

$$z_{ceros} = \begin{cases} e^{j.\pi/2} = j \\ e^{-j.\pi/2} = -j \end{cases}$$

$H_{1\ Modificaddo}(z) = \frac{(z-j).(z+j)}{z^2} = \frac{z^2+1}{z^2}$; FIR Estable

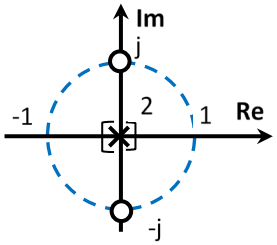


Fig. 49 – Diagrama de polos y ceros del filtro FIR

Ejercicio XXIV

Filtros de media móvil

El siguiente sistema digital de la Fig. 50 representa un *Filtro de Media Móvil* (ampliamente utilizado en el Procesamiento Digital de Señales).

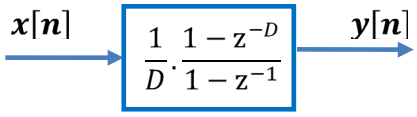


Fig. 50 – Diagrama del filtro de media Móvil

Se pide:

a) Calcular y graficar la Respuesta al Impulso de este filtro, es decir, $h[n]$.
b) ¿Dónde quedan ubicados los Polos y los Ceros de $H(z)$? Ejemplifique graficando la ubicación de estos con D = 5 .
c) Graficar la Respuesta en Amplitud respecto de w de este filtro digital, es decir, $H(e^{j.\Omega})$. Usar como en el inciso anterior, D=5, para propósitos gráficos.
d) Calcular el valor de la Salida de este filtro para tiempos discretos altos, es decir $\lim_{n\to\infty} y[n]$, si a la entrada se coloca la señal: $x[n] = A.seno(\Omega_0.n).u[n]$

Respuestas: a) $h[n] = \frac{1}{D}.\left[u[n] - u[n-D]\right]$

Filtro FIR

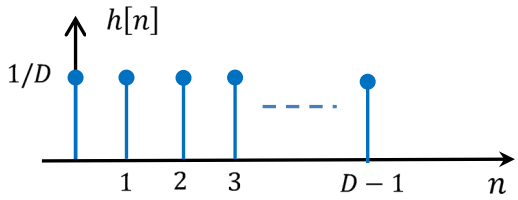


Fig. 51 – Respuesta impulsional del filtro de media móvil

b) Con D=5

Ceros: $z = e^{\frac{2.k.\pi}{5}}$; k= 1, 2, 3, 4 72º, 144º, 216º, 288º

Polos: Cantidad 4= D-1 -→ 4 Polos en z=0

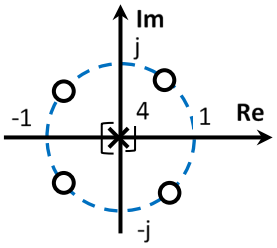


Fig. 52 – Diagrama de polos y ceros correspondiente al filtro

c) $H(e^{j.w}) = \frac{1}{D}.\frac{sen(w.{}^{D}/_{2})}{sen({}^{w}/_{2})}.e^{-j.\frac{w(D-1)}{2}}$

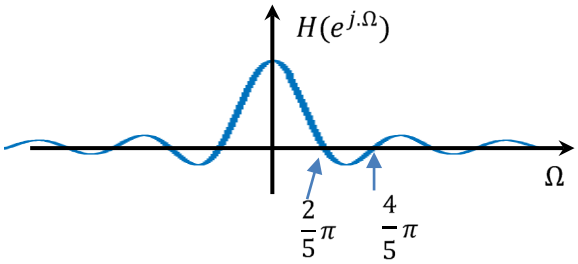


Fig. 53 – Respuesta en frecuencia del filtro de media móvil

d) $\lim_{n\to\infty}(y[n]) = 0$

Filtros de media móvil

Los filtros de media móvil se utilizan mucho para suavizar datos ruidosos. La señal filtrada con filtro de media movil se calcula mediante la siguiente ecuación:

$X = \frac{\sum_{i=1}^{N} x_i}{N};$

Los filtros de media móvil desplazan una ventana de longitud fija a través de los datos. En cada paso se calculan los promedios correspondientes a los datos contenidos en la ventana.

Ejercicio XXV

Ejemplo de filtrado de señal ECG

```
%% Filtro de Media Móvil
 %load('X.mat');
  clear all
  % Señal de ECG: Electrocardiograma
  ECG = [90,  95, 124, 153, 182, 211, 241, 230, 202, 173, 143, 114, ...
     89,  83, 75,  67, 70,  78, 85,  90, 90,  90, 90,  90, ...
     92, 100, 107, 113, 118, 122, 124, 125, 124, 121, 117, 111, 104, ...
     97,  90,  90,  90,  90,  90,  90,  90, 90,  90,  90,  92,  93, ...
     93,  92, 90,  90, 90,  90,  90,  90, 90,  90, 90,  90,  90, ...
     90,  90,  90,  90,  90,  90, 90,  97, 104, 109, 112, 113, 111, ...
     107, 100,  91, 90,  90,  90,  90,  90,  90] ;
  x= [ECG ECG] ; % Tomo 2 períodos de ECG
  fs = 100 ; dt = 1/fs ;  % Debo conocer fs para generar el vector de tiempo
  t =(0: size(x,2)-1 ) *dt ;
  x= x + 30*rand(size(t));  % Le sumo ruido
  % Algoritmo Propio
  % % Coeficientes del filtro
  %    n=5 ;  % tamaño de ventana, modificar n dependiendo de la señal
  %    p=ones(n,1) ;  p=p/(sum(p));  %p=[1/5  1/5  1/5  1/5   1/5 ];
  %    P=x+NaN;  % vector vacío
  %    LARGO=length(x); % largo de la señal
  %    largo_ventana =length(p); % largo de la ventana
  %    largo1 =fix(largo_ventana /2); % la mitad del anterior y redondeo hacia abajo
  %   % Filtro
  %    for i= largo1 +1: LARGO -( largo1+1)
  %        x_filtrada(i)=x(i- largo1:i+ largo1) *p;   %
  %    end
  windowSize = 5;
  b = (1/windowSize)*ones(1,windowSize);
  a = 1;
  x_filtrada = filter(b,a,x);
```

```
% gráficos
  figure(100)   % si uso guide:  axes(handles.axes1) ;
  plot(t, x) ;  hold on ;
  t =(0: size(x_filtrada,2)-1 ) *dt ;
  plot(t, x_filtrada, 'r')
  grid on
  legend('Señal ECG con Ruido', 'Señal Filtrada')
```

En la Fig. 54 se muestra la señal ECG ruidosa y la señal filtrada

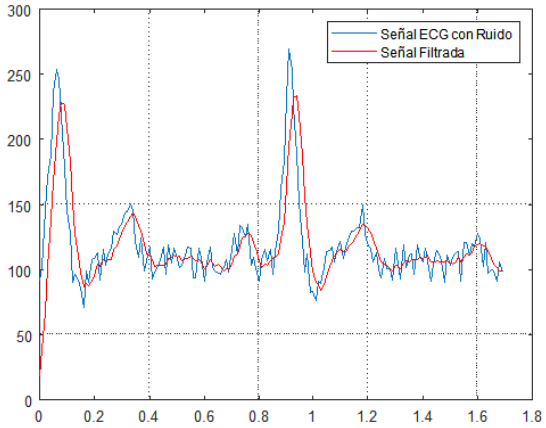


Fig. 54 – Ejemplo de filtrado de señal ECG

Ejercicio XXVI

Ejemplo simple de filtrado

Mediante la función filter de Matlab, se pide implementar un filtro de media móvil con tamaño de ventana igual a 7.

Se genera un vector de datos correspondiente a una señal senoidal contaminado con ruido aleatorio.

$$y[n] = \frac{1}{7}.(x[n] + x[n-1] + x[n-2] + x[n-3] + x[n-4] + x[n-5] + x[n-6])$$

Reemplazando $x[n] = \delta[n]$, se obtiene $h[n]$ (respuesta impulsional):

$$h[n] = \frac{1}{7}.(\delta[n] + \delta[n-1] + \delta[n-2] + \delta[n-3] + \delta[n-4] + \delta[n-5] + \delta[n-6])$$

Transformamos esta ecuación:

$$y[n] = \frac{1}{7}.(x[n] + x[n-1] + x[n-2] + x[n-3] + x[n-4] + x[n-5] + x[n-6])$$

$$x[n-n_0] \longleftrightarrow e^{-j.\Omega.\, n_0}.\ X(\Omega) \quad \text{(tablas)}$$

$$Y(\Omega) = \frac{1}{7}.X(\Omega)\left(1 + e^{-j.\Omega.} + e^{-j.\Omega.\,2} + e^{-j.\Omega.\,3} + e^{-j.\Omega.4} + e^{-j.\Omega.5} + e^{-j.\Omega.6}\right)$$

$$H(\Omega) = \frac{Y(\Omega)}{X(\Omega)} = \frac{1}{7}.\left(1 + e^{-j.\Omega.} + e^{-j.\Omega.\,2} + e^{-j.\Omega.\,3} + e^{-j.\Omega.4} + e^{-j.\Omega.5} + e^{-j.\Omega.6}\right) \quad ; \quad z = e^{j.\Omega}$$

$$Y(z) = \frac{1}{7}.X(z)\left(\frac{1 + z^{-1} + z^{-2} + z^{-3} + z^{-4} + z^{-5} + z^{-6}}{1}\right)$$

Función filter utiliza esta forma (help Matlab®):

$$Y(z) = X(z)(b(1) + b(2).z^{-1} + b(3).z^{-2} + b(4).z^{-3} + b(5).z^{-4} + \cdots) - Y(z)(a(2).z^{-1} + a(3).z^{-2} + \cdots a(n).z^{-n+1})$$

$$H(z) = \frac{Y(z)}{X(z)} = \frac{1}{7}.\left(\frac{1+z^{-1}+z^{-2}+z^{-3}+z^{-4}+z^{-5}+z^{-6}}{1}\right) = \frac{\left[\frac{1}{7}\ \frac{1}{7}\ \frac{1}{7}\ \frac{1}{7}\ \frac{1}{7}\ \frac{1}{7}\ \frac{1}{7}\right]}{1}$$

```
% Ejemplo simple de Filtro de Media Móvil con ventana
t  =  -pi:0.05:pi ;
rng default  % Inicializamos el generador de números aleatorios
% Generamos una señal senoidal con ruido aleatorio
x = 2*sin(t)+ 0.8* rand(size(t));
% Elegimos un tamaño de ventana 7 y calculamos los coeficientes del filtro, es decir que obtenemos
% el numerador y el denominador correspondiente a la transferencia del filtro.
largo_ventana = 7;
b = (1/largo_ventana) *ones(1,largo_ventana); % Numerador del filtro
a = 1;  % Denominador del filtro
y =  filter(b,  a,  x);
% Graficamos las señales
plot(t,x, 'linewidth',2) ; hold on; grid on
plot(t,y, 'linewidth',2) ;
legend('Señal Original', 'Señal Filtrada')
axis tight
```

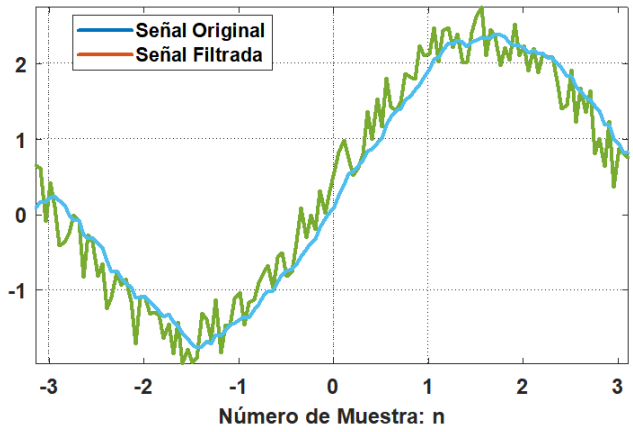


Fig. 55 – Ejemplo de filtrado de señales

Descripción función filter de Matlab®

y = filter(b, a, x) ;

La función filter filtra los datos del vector x, mediante una función de transferencia. Siendo b un vector con los coeficientes del numerador y a contiene los coeficientes del denominador.

Si a(1) es distinto a 1, filter normaliza los coeficientes dividiendo a cada uno por el coeficiente a(1). El coeficiente a(1), para poder dividir, no puede ser nulo.

- Si x es un vector, la función filter devuelve un vector del mismo largo con los datos filtrados.
- Si se ingresa una matriz x, entonces, filter toma cada columna y devuelve los datos filtrados.
- Si se ingresa una matriz x multidimensional, entonces filter busca la primera dimensión con tamaño distinto de 1 y actúa sobre ella.

Ejercicio XXVII

Ecualizador implementado en Python

Se desea implementar un ecualizador de audio mediante filtros FIR de orden 62. Verificar el funcionamiento con un archivo de audio de corta duración. Reproducir en el parlante de la computadora para verificar el correcto funcionamiento de filtrado.

Resolución en Python

```
def boton_25_Ej_Filtro_FIR_Audio():
    import matplotlib.pyplot as plt
  # Ejemplo de filtros FIR con Audio
    import numpy as  np
    from scipy.io import  wavfile
    from scipy.signal import freqz
    from scipy import  signal
    from numpy import  pi,  absolute,  arange
   #  Leemos el archivo de audio
    sr, x = wavfile.read('Audio1.wav')      # xx-bit mono o estéreo xx khz
  # Calculo el espectro de la señal
    from scipy.fftpack import fft
    Fs = sr
    N=len(x)  # Number of sample points
    Ts = 1.0 /  Fs    # tiempo de muestreo
    t =  np.linspace(0.0 , N*Ts, N)
    plt.figure()
```

```
  plt.plot(t, x, label='Señal Original !')
  plt.xlabel('t') ;       plt.ylabel('y')
  plt.title('Señal Original\n x')
  plt.legend(),  plt.show()
  xf =  fft(x)
  frec = np.linspace(0.0, 1.0/(2.0*Ts), N//2)
  plt.figure()
  plt.plot(frec, 2.0/N * np.abs(xf[0:N//2]))
  plt.grid() ; plt.xlabel("Frecuencia Hz"); plt.ylabel('Amplitud')
  plt.title('Transformada de Fourier - FFT valor absoluto')
  plt.show()
# Prueba del Filtro con Orden muy bajo  !!!
  b =  signal.firwin( 5, cutoff=1000, fs=sr, pass_zero= False)  #
  x1 = signal.lfilter( b, [1.0],  x)  #
  wavfile.write('sampleX.wav', sr, x1.astype(np.int16))
# Orden de los filtros
  N = 62
 # Filtro Pasa Bajo
  fn = 0.2    # Frecuencia normalizada de corte
  b_LowPass = signal.firwin(N, fn, pass_zero=True)
  x_LowPass = signal.lfilter(b_LowPass, [1.0], x)
  wavfile.write('sample_LP.wav', sr, x_LowPass.astype(np.int16))
# Filtro Pasa Banda
  fn = [0.20,  0.50]   # Frecuencia normalizada de corte
  b_BandPass = signal.firwin(N, fn, pass_zero=False)
  x_BandPass = signal.lfilter(b_BandPass, [1.0], x)
  wavfile.write('sample_BP.wav', sr, x_BandPass.astype(np.int16))
# Filtro Pasa Alto
  fn = 0.40   # Frecuencia normalizada de corte
  b_HighPass = signal.firwin(N +1 , fn, pass_zero=False)
  x_HighPass = signal.lfilter(b_HighPass, [1.0], x)
  wavfile.write('sample_HP.wav', sr, x_HighPass.astype(np.int16))
# Grafico
  plt.figure()
  plt.suptitle('Ecualizador de Audio - Respuesta en Frecuencia')
```

```
    w, h = freqz(b_LowPass, worN=1000)  # freqz usado para filtro digital
    plt.plot( w/(pi) , absolute(h),'b', linewidth=2 , label='FIR pasabajo')
    w, h = freqz(b_BandPass, worN=1000)  # freqz usado para filtro digital
    plt.plot( w/(pi) , absolute(h),'r', linewidth=2 , label='FIR pasabanda')
    w, h = freqz(b_HighPass, worN=1000)  # freqz usado para filtro digital
    plt.plot( w/(pi) , absolute(h),'g', linewidth=2 , label='FIR pasaalto')
    plt.ylim(-0.05, 1.05) ;   plt.grid(True)
    plt.ylabel('Ganancia') ;  plt.xlabel('Frecuencia Normalizada')
    plt.legend(loc=1, prop={'size': 6})
  # Ganancia para cada banda
    gLP = 0.4  ;   gBP = 1.5  ;    gHP = 1.0
    yA= gLP * x_LowPass   #pasabajo
    yB= gBP * x_BandPass  # pasabanda
    yC= gHP * x_HighPass #  pasalto
  #  Reconstruyo la señal
    yD = yA + yB + yC
    wavfile.write('Audio_Reconst.wav', sr, yD.astype(np.int16))
  # Reproduzco
    import os
    os.system("Audio_Reconst.wav")
boton_25_Ej_Filtro_FIR_Audio()
```

En la Fig. 56 y Fig. 57 se muestra el espectro de la señal y la respuesta en frecuencia de los 3 filtros.

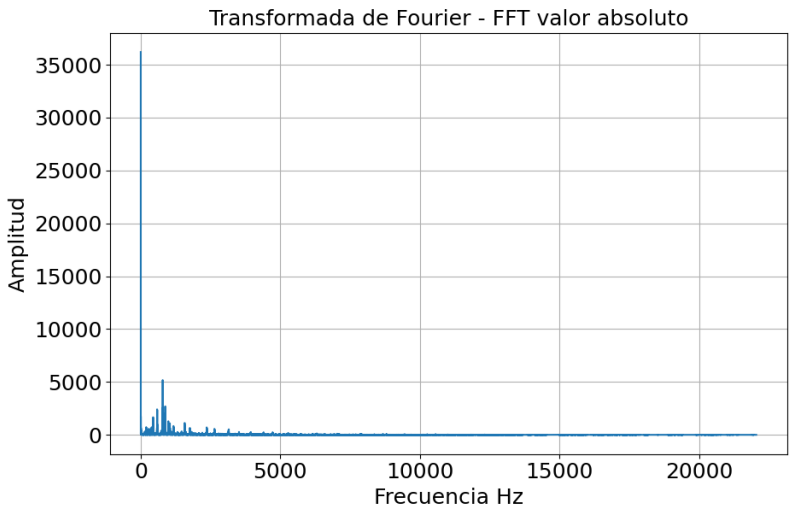


Fig. 56 – Espectro de la señal de audio

Fig. 57 – Respuesta en frecuencia normalizada de los filtros pasa bajos, pasa banda y pasa altos del ecualizador

Descarga de los códigos de los ejercicios

I – Introducción a Sistemas Adaptativos

El problema del filtrado

El filtro adaptativo se puede definir como un sistema que modela la relación entre las señales trabajando en tiempo real, y en forma progresiva.

En Procesamiento de Señales, el término filtrar se utiliza para modificar el comportamiento espectral de una señal. Los filtros tradicionales tienen comportamiento lineal e invariantes en el tiempo. Realizan operaciones lineales sobre la entrada, para obtener una salida basada en los coeficientes del filtro. Estos sistemas son Invariantes en el tiempo, debido a que sus parámetros no varían con el tiempo.

Los filtros adaptativos, pueden no cumplir con la condición de Invarianza en el tiempo, es decir que sus parámetros se modifican en el tiempo mediante un algoritmo de adaptación y ajuste. Los coeficientes del filtro se ajustan para obtener una mejor respuesta del sistema.

Al diseñar el filtro se desconocen sus coeficientes, estos coeficientes se calculan al implementar el filtro y se ajustan en cada iteración.

Debido a que los sistemas adaptativos no son invariantes en el tiempo y tampoco lineales dificulta las operaciones y salvo algunas excepciones no se puede aplicar las transformadas en frecuencia, en Laplace y en dominio Z. Las transformadas se utilizan en el filtro, pero no en el sistema completo.

A continuación, se muestran distintas definiciones de la palabra filtro:

a) El filtro no deja pasar ciertos elementos y permite pasar otros.

b) Sistema que elimina o deja pasar ciertas frecuencias de un espectro, correspondiente a un sistema eléctrico, térmico, hidráulico, mecánico, acústico, etc.

c) Sistema de selección con criterios previamente determinados.

El filtro es un sistema que se utiliza para extraer información a partir de datos ruidosos. También se utiliza para eliminar valores atípicos en las mediciones (outliers). Los filtros se utilizan en diferentes aplicaciones, por ejemplo: mediciones eléctricas, comunicaciones, radar, economía, bioingeniería, entre otras.

En la Fig. 58 se muestra un esquema de transmisión, donde la fuente de transmisión puede ser analógica o digital. La degradación del canal puede dar lugar a interferencias entre símbolos, la modificación de los sucesivos pulsos (que representan la secuencia transmitida de unos y ceros) entre sí puede generar resultados erróneos y pérdida de información. La señal recibida también puede tener ruido, este ruido puede ser interno al sistema, como en el caso del ruido térmico

generado por un amplificador, o externo al sistema debido a señales interferentes perturbadoras que provienen de otras fuentes.

Utilizamos $x(t)$ para señales de tiempo continuo y $x[n]$ para señales de tiempo discreto

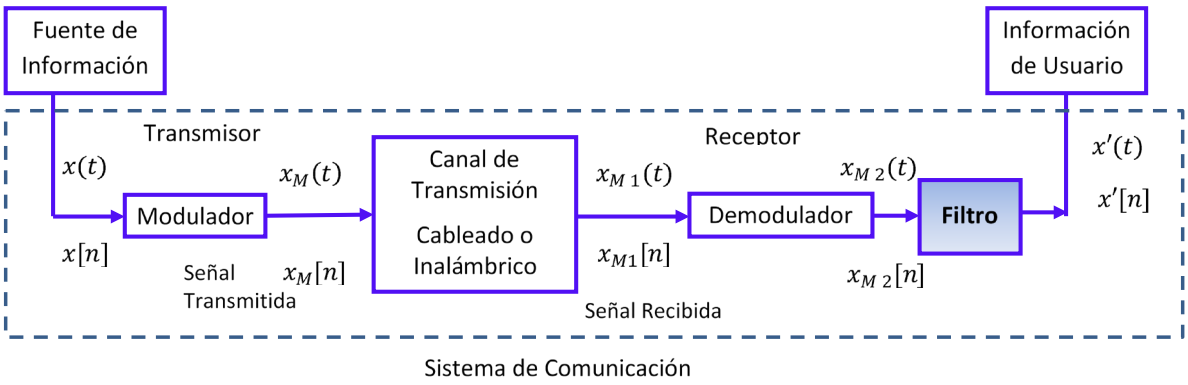


Fig. 58– Diagrama en Bloques genérico de un sistema de comunicación.

El sistema puede tener ruido, interferencias y distorsiones. Las perturbaciones no intencionales de las señales se denominan ruido. La interferencia está dada por la contaminación de señales con formas similares a la señal de datos original, generalmente estas interferencias se generan artificialmente. La distorsión se debe a la alteración de la señal deseada provocada por la respuesta deformada del sistema a esa señal. A diferencia de la interferencia y el ruido, solo se tiene distorsión cuando se aplica la señal (desaparece cuando no hay señal de datos).

La señal recibida en la salida del canal es ruidosa y está distorsionada. El receptor debe entregar una señal confiable como una *estimación de la señal del mensaje original a un usuario* en la salida del sistema.

Otro ejemplo se puede dar en los errores debidos al sistema de medición, donde se suele indicar un valor medio o valor más probable y una desviación estándar o franja de error. Para predecir estos valores se utilizan estimaciones.

Se tienen 3 tipos básicos de estimación: filtrado, predicción y suavizado.

- ✓ El *filtrado* es una operación que implica la extracción de información en un tiempo de interés t mediante la medición de datos hasta tiempo incluido t.
- ✓ La *predicción* es la parte de pronóstico de la estimación. Su objetivo es obtener información sobre cómo serán los datos de interés en algún momento $t + \tau$ en el futuro (para algún $\tau > 0$) mediante el uso de datos medidos hasta el tiempo t incluido.
- ✓ El *suavizado* es una estimación a posteriori (es decir, después del hecho), en la que los datos medidos después del tiempo de interés se utilizan en la estimación. Específicamente, la estimación suavizada en el tiempo t' se obtiene utilizando datos medidos en el intervalo [0, t'], donde $t' < t$. Por tanto, hay un retraso de $t - t'$ involucrado en los cálculos de la estimación suavizada. El beneficio obtenido al esperar que se acumulen más datos es que el suavizado puede producir una estimación más precisa que el filtrado.

Introducción a Filtros adaptativos

Un filtro adaptativo es un filtro capaz de ajustar sus coeficientes automáticamente mediante un algoritmo adaptativo. Puede adaptarse a la señal de entrada para obtener una salida deseada, o también es capaz de identificar los parámetros que tiene un sistema desconocido. Los filtros adaptativos desempeñan un papel importante en los productos modernos de procesamiento de señales digitales (DSP) en áreas como la cancelación de eco telefónico, cancelación de ruido, ecualización de canales de comunicación, mejora de señal biomédica, control activo de ruido y sistemas de control adaptativo.

Los filtros adaptativos funcionan generalmente para la adaptación de entornos de cambio de señal, superposición espectral entre ruido y señal y ruido desconocido o variable en el tiempo. Por ejemplo, cuando el ruido de interferencia es fuerte y su espectro se superpone al de la señal deseada, ver Fig. 59, el enfoque convencional fallará en preservar el espectro de señal deseado mientras se elimina la interferencia usando un filtro tradicional. Sin embargo, un filtro adaptativo puede resolver este problema. En este capítulo se introducen algunos fundamentos del tema, es decir, filtros adaptativos de respuesta de impulso finito (FIR) con un algoritmo de mínimos cuadrados mínimos (LMS) simple y una breve introducción a RLS.

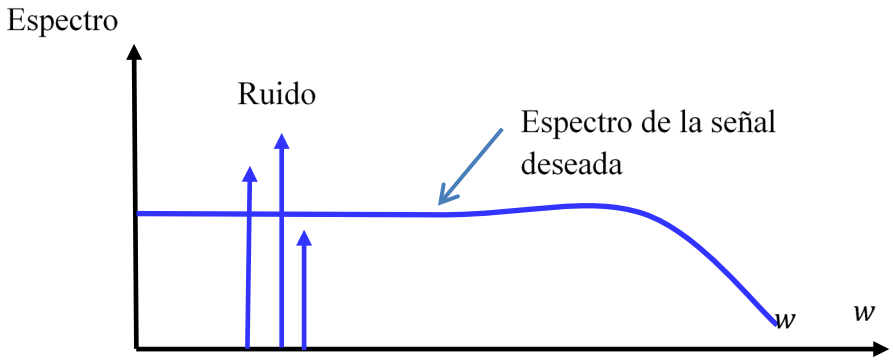


Fig. 59 – Superposición de espectros entre la señal deseada y el ruido

Un sistema adaptativo está compuesto por el filtro adaptativo, el algoritmo de adaptación y uno o varios sumadores, como se observa en la Fig. 60. A la salida del filtro se tiene la señal $y[n]$ a partir de la entrada $x[n]$. A la señal $y[n]$ se le resta la señal de referencia $d[n]$, generando la señal $e[n]$, el algoritmo adaptativo ajusta los coeficientes del filtro muestra a muestra para minimizar el error en forma progresiva. Generalemnte $d[n]$ es la señal deseada o señal de referencia y la señal $e[n]$ se trata de minimizar en pasos sucesivos. Para un sistema digital se pueden utilizar filtro FIR o IIR. Se suelen utilizar los filtros FIR debido a que son más estables y más simples de implementar.

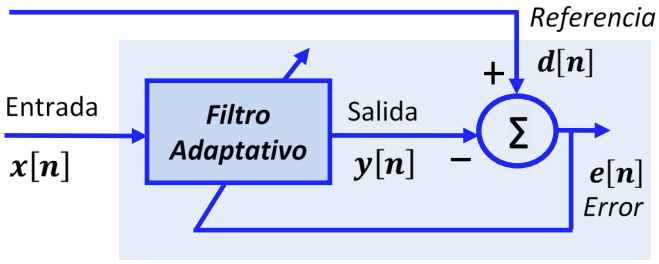


Fig. 60 - Sistema adaptativo con alimentación a priori (feedforward)

En forma resumida se muestra su funcionamiento para un filtro de un solo coeficiente. Al sistema le llegan 2 señales $x[n]$ y $e[n]$. Siendo $e[n]$ la señal de error, esta resulta de la resta de la señal deseada $d[n]$ y la señal de salida del filtro $y[n]$

La salida del filtro se obtiene mediante:

$$y[n] = w[n].x[n] \qquad (1)$$

Siendo $w[n]$ los coeficientes del filtro

Se obtiene la Ecuación para el cálculo de la señal de error con un solo coeficiente

$$e[n] = d[n] - y[n] \qquad (2)$$

Generalmente se trata de que la señal $e[n]$ sea cero, para esto a partir de la señal $x[n]$ se genera $y[n]$ y se van ajustando los coeficientes del filtro para que $y[n]$ sea igual a la señal esperada $d[n]$.

Los sistemas adaptativos contienen filtros. Podemos clasificar los filtros en lineales o no lineales. El filtro es lineal si la cantidad filtrada, suavizada o predicha en la salida del filtro es una función lineal de las entradas del filtro. De lo contrario, el filtro no es lineal.

En el enfoque estadístico para la solución del problema de filtrado lineal, asumimos la disponibilidad de ciertos parámetros estadísticos (por ej.: funciones de media y correlación) de la señal útil y el ruido aditivo no deseado. La idea es diseñar un filtro lineal basado para datos de entrada con ruido y reducir los efectos del ruido en la salida del filtro de acuerdo con algún criterio estadístico. Un enfoque útil para este problema es optimizar el filtro minimizando el valor cuadrático medio de la señal de error, esta señal está definida como la diferencia o la resta entre la respuesta deseada y la señal de salida real del filtro. Para las entradas estacionarias, la solución resultante se conoce comúnmente como filtro de Wiener, se dice que es óptimo en el sentido del error cuadrático medio. Un gráfico del valor cuadrático medio para la señal de error frente a los parámetros ajustables de un filtro lineal se denomina superficie de comportamiento de error. El punto mínimo de esta superficie representa la solución de Wiener.

El filtro Wiener es inadecuado para tratar situaciones en las que la **no estacionariedad de la señal y / o el ruido** son intrínsecas al problema. En tales situaciones, el filtro óptimo tiene que asumir una forma variable en el transcurso del tiempo. Una solución exitosa a este problema difícil puede ser el filtro de **Kalman**, que es un sistema potente con una amplia variedad de aplicaciones en la ingeniería.

La teoría del filtro lineal, que abarca los filtros de Wiener y Kalman, está bien desarrollada en la bibliografía para señales de tiempo continuo y de tiempo discreto. Por razones técnicas influenciadas por la amplia disponibilidad de computadoras y el uso cada vez mayor de dispositivos de procesamiento de señales digitales, en la práctica la representación de señales en tiempo discreto es a menudo el método más utilizado. En consecuencia, solo analizamos la versión de tiempo discreto de los filtros de Wiener. En el tiempo discreto, las señales de entrada y las de salida, así como los parámetros de los propios filtros, se definen todas en instantes discretos de tiempo. En cualquier caso, las señales de tiempo continuo siempre se pueden representar por una secuencia de muestras que se derivan de la observación de la señal en instantes de tiempo uniformemente espaciados. No se incurre en pérdida de información en la conversión siempre que, por supuesto, se cumpla el conocido **teorema de Muestreo.** Este teorema dice que la frecuencia de muestreo tiene que ser mayor al doble de la frecuencia máxima de la señal de tiempo continuo. Por tanto, podemos representar en el tiempo continuo una señal $x(t)$ por la secuencia $x[n]$ con $n = 0; 1; 2; 3; \ldots\ldots$, donde por conveniencia se normaliza el período de muestreo al valor 1 (uno), una práctica que se utiliza a lo largo del libro.

Filtros Adaptativos

El diseño de un filtro Wiener necesita información a priori de las estadísticas correspondientes a los datos que va a procesar. El filtro es óptimo solo cuando las características estadísticas de los datos de entrada coinciden con la información a priori en la que se construyó el filtro. Sin embargo, cuando esta información no se conoce completamente, seguramente no será posible diseñar el filtro Wiener o el diseño no resulta óptimo. Un enfoque sencillo que podemos utilizar en estos casos es el procedimiento de "estimación y conexión". Se trata de un proceso de dos etapas en el que el filtro en primer lugar "estima" las estadísticas relevantes de las señales. Y luego "conecta" los resultados así obtenidos en una fórmula no recursiva para calcular los parámetros del filtro. Para la operación en tiempo real, se necesita mucha carga computacional con hardware costos y complicado. Para mitigar esta limitación, podemos utilizar un ***filtro adaptativo***. Por tal sistema nos referimos a uno que se diseña a sí mismo en el sentido de que el filtro adaptativo se basa para su funcionamiento en un algoritmo recursivo. Esto permite que el filtro funcione correctamente en un entorno donde no se tiene de un conocimiento completo de las características relevantes de la señal. El algoritmo parte de un conjunto predeterminado de condiciones iniciales, que representan todo lo que sabemos sobre el medio ambiente. En un ***entorno estacionario***, encontramos que después de sucesivos ciclos de adaptación del algoritmo, converge a la solución óptima de Wiener en algún sentido estadístico. En un ***entorno no estacionario***, el algoritmo ofrece una capacidad de seguimiento, debido a que puede realizar un seguimiento de las ***variaciones temporales*** en las estadísticas de los datos de entrada, siempre que las variaciones sean lo suficientemente lentas.

Como consecuencia directa de la aplicación de un algoritmo recursivo mediante el cual los parámetros del filtro adaptativo se actualizan entre un ciclo de adaptación al siguiente, los parámetros se vuelven dependientes de los datos. **Esto, por tanto, significa que un filtro adaptativo es en realidad un sistema no lineal, en el sentido de que no cumple con el principio de superposición.**

A pesar de esta propiedad, los filtros adaptativos se clasifican comúnmente como lineales o no lineales.

Se dice que un filtro adaptativo es lineal si su mapa de entrada-salida obedece al principio de superposición siempre que sus parámetros se mantengan fijos. De lo contrario, se dice que el filtro adaptativo no es lineal.

Estructuras de filtros lineales

El funcionamiento de un algoritmo de ***filtrado adaptativo lineal*** implica dos procesos básicos: (1) un proceso de filtrado diseñado para producir una salida en respuesta a una secuencia de datos de entrada y (2) un proceso adaptativo, cuyo propósito es proporcionar un mecanismo para el control adaptativo de un conjunto ajustable de parámetros utilizados en el proceso de filtrado. Estos dos procesos funcionan de forma interactiva entre sí. Naturalmente, la elección de una estructura para el proceso de filtrado tiene un efecto profundo en el funcionamiento del algoritmo en su conjunto.

La respuesta impulsional (respuesta al impulso) de un filtro lineal determina la memoria del filtro. Sobre esta base, podemos clasificar los filtros lineales en filtros de respuesta al impulso de duración finita (FIR) y de respuesta al impulso de duración infinita (IIR), que se caracterizan

respectivamente por una memoria finita y una memoria infinitamente larga, pero que decrece temporalmente.

Filtros lineales con memoria finita

Se tienen 3 tipos de estructuras de filtro en el contexto de un filtro adaptativo con memoria finita:

1 - Filtro FIR. También conocido como filtro de línea de retardo con derivación o filtro transversal, el filtro FIR consta de tres elementos básicos, como se muestra en la Fig. 61: (a) elementos de retardo unitario: z^{-1}, (b) multiplicadores y (c) sumadores. El número de elementos de retardo utilizados en el filtro determina la duración finita de su respuesta al impulso. El número de elementos de retardo, se denomina comúnmente orden del filtro. En esta figura, cada uno de los elementos de retardo está identificado por el operador de retardo unitario z^{-1}. En particular, cuando z^{-1} opera en la entrada $x[n]$, la salida resultante es $x[n]$. La función de cada multiplicador en el filtro es multiplicar la entrada del bloque por el peso del bloque (coeficiente del filtro). Por lo tanto, un multiplicador conectado a la k-ésima entrada de un bloque $x[n-k]$ produce $b_k.x[n-k]$, donde b_k es el respectivo peso del bloque con $k = 0,1,2 \dots, N$. Los pesos o coeficientes b_k pueden ser números complejos. Mediante los sumadores, se suman las salidas de los multiplicadores individuales para hallar la respuesta del filtro.

Para el filtro FIR que se muestra, la salida viene dada por:

$y[n] = \sum_{k=0}^{N-1} b_k.x[n-k]$

La ecuación se denomina suma de convolución finita en el sentido de que convoluciona la respuesta de impulso de duración finita del filtro, b_k, con la entrada de filtro $x[n]$ para producir la salida de filtro $y[n]$.

En la figura los términos b_k son los coeficientes y los z^{-1} son los retardos de tiempo T. Existen muchas variaciones de esta estructura. Se pueden interconectar varios filtros en serie, en cascada, etc.

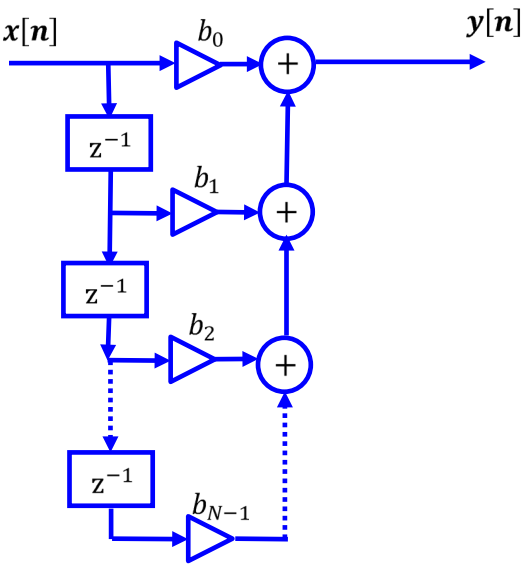


Fig. 61– Estructura del Filtro FIR

Partiendo de esta ecuación:

$$\boxed{y[n] = \sum_{k=0}^{N-1} b_k . x[n-k]}$$

$N-1$ es el orden del filtro, coincide con la cantidad de términos no nulos y coincide con la cantidad de coeficientes b_k que tiene el filtro

Aplicamos «Transformada Z» en ambos miembros:

$$\boxed{f[n \pm a] \longleftrightarrow z^{\pm a} . F(z)} \quad \text{; para CIN}$$

$$Y(z) = \sum_{k=0}^{N-1} b_k . z^{-k} . X(z)$$

$$H(z) = \frac{Y(z)}{X(z)} = \left(\sum_{k=0}^{N-1} b_k . z^{-k} \right) \frac{z^{N-1}}{z^{N-1}}$$

$$\boxed{H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{k=0}^{N-1} w_k . z^{-k+N-1}}{z^{N-1}}}$$

Los filtros FIR solo tienen polos en el Origen. También llamado «Todo Ceros» (los polos son fijos, los ceros caracterizan la respuesta del filtro)

Respuesta Impulsional $h[n]$:

La respuesta impulsional es la salida del sistema cuando la entrada es: $\mathrm{x}[n] = \delta[n]$

$$h[n] = h[n]|_{\mathrm{x}[n]=\delta[n]}$$

Se puede expresar la salida en función de h: $y[n] = \sum_{k=0}^{N-1} h_k . x[n-k]$

2 - Lattice predictor: Predictor de celosía (enrejado). Un predictor de “celosía” tiene una estructura modular, en el sentido de que consta de una serie de etapas individuales, cada una de las cuales tiene la apariencia de una celosía, de ahí el nombre "celosía" como descriptor estructural. La Fig. 62 representa un predictor de red que consta de M etapas; el número M indica el orden de predicción. La m-ésima etapa del predictor de celosía que se muestra se describe mediante el par de relaciones de entrada-salida (asumiendo el uso de datos de entrada estacionarios de sentido amplio y valores complejos).

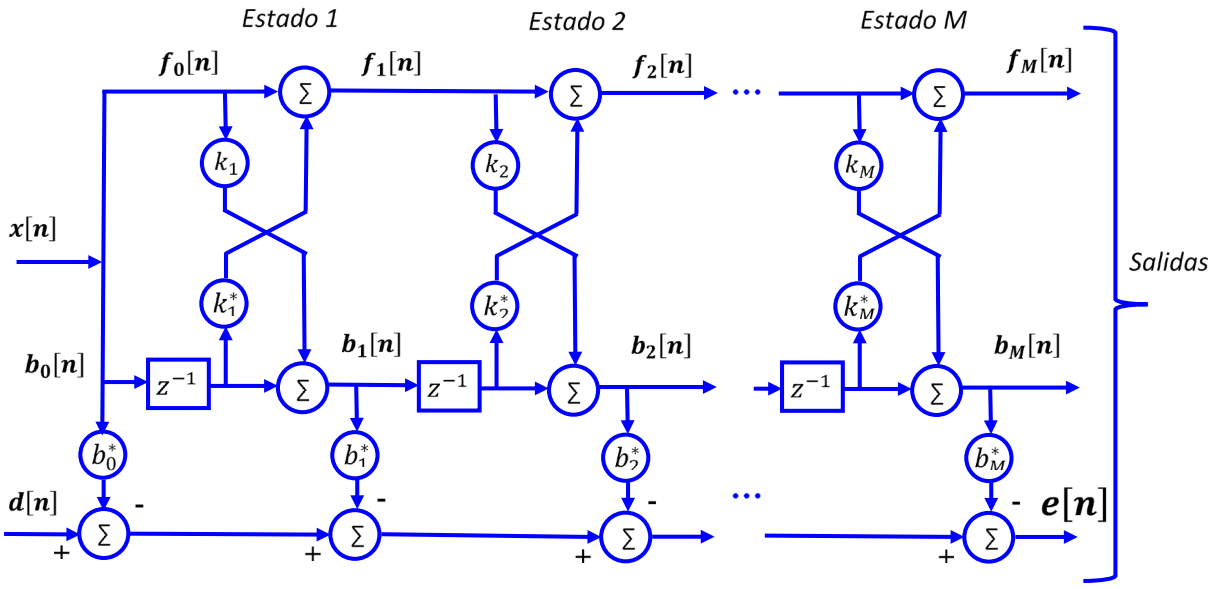


Fig. 62 – Filtro Lattice predictor

Las siguientes ecuaciones corresponden al filtro Lattice predictor:

$$f_m[n] = f_{m-1}[n] + k_m^*.b_{m-1}[n-1]$$

$$b_m[n] = b_{m-1}[n-1] + k_m.f_{m-1}[n]$$

Siendo $m = 0,1,2..M$, con M el orden de predicción final. La variable $f_m[n]$ es el emésimo error de predicción hacia adelante y $b_m[n]$ es el emésimo error de predicción hacia atrás. El coeficiente k_m se llama coeficiente de reflexión m. El error de predicción directa $f_m[n]$ se define como la diferencia o resta entre la entrada $x[n]$ y el valor de predicción de un paso; este último se basa en el conjunto de m entradas pasadas $x[n-1],\ldots,x[n-m]$. En consecuencia, el error de predicción hacia atrás $b_m[n]$ se define como la diferencia entre la entrada $x[n-m]$ y su predicción "hacia atrás" basada en el conjunto de m entradas "futuras" $x[n],\ldots,x[n-m+1]$. Considerando las condiciones en la entrada de la etapa 1 en la figura, tenemos:

$$f_0[n] = b_0[n] = x[n] \qquad (3)$$

Siendo $x[n]$ la entrada del predictor de celosía en el momento n. Por tanto, partiendo de las condiciones iniciales de la (4) y dado el conjunto de coeficientes de reflexión $k_1, k_2, \ldots\ldots k_M$, podemos determinar el par final de salidas $f_M[n]$ y $b_M[n]$ moviéndonos a través del predictor lattice, etapa por etapa. Para una secuencia de entrada correlacionada $x[n],\ldots,x[n-m]$ extraídos de un proceso estacionario, los errores de predicción hacia atrás $b_0[n]$, $b_1[n]$,....$b_M[n]$ forman una secuencia de variables aleatorias no correlacionadas. Además, existe una correspondencia biunívoca entre estas dos secuencias de variables aleatorias en el sentido de que, si se nos da una de ellas, podemos determinar de forma única la otra, y viceversa. En consecuencia, una combinación lineal de los errores de predicción hacia atrás $b_0[n]$, $b_1[n]$,....$b_M[n]$ puede usarse para proporcionar una estimación de alguna respuesta deseada $d[n]$, como se muestra en la mitad inferior de la Fig. 62. La diferencia entre $d[n]$ y la estimación así producida representa el error de estimación $e[n]$. El proceso que se describe en el presente documento se denomina estimación de proceso conjunto. Naturalmente, podemos usar la secuencia de entrada original $x[n], x[n-1]$ ……$, x[n-M]$para producir una estimación de la respuesta deseada $d[n]$ directamente. El método indirecto representado en la figura, sin embargo, se simplifica el cálculo de los pesos de

derivación $h_0, h_1, \ldots\ldots, h_M$ explotando la naturaleza no correlacionada de los correspondientes errores de predicción hacia atrás utilizados en la estimación.

3 - Matriz sistólica (systolic array). Una matriz sistólica representa una red de computación paralela ideal para *mapear* una serie de cálculos grandes de álgebra lineal, tales como multiplicación, triangularización y sustitución inversa de matrices. Se pueden distinguir dos tipos básicos de elementos de procesamiento en una matriz sistólica: células límite y células internas. Sus funciones o bloques se muestran en las Fig. 63 (a) y (b), respectivamente. En cada caso, el parámetro r representa un valor almacenado dentro de la celda. La función de la celda límite es producir una salida igual a la entrada x dividida por el número r almacenado en la celda. La celda interna tiene doble función: (a) multiplicar la entrada s (que viene de arriba) por el número r almacenado en la celda, restar el producto rs de la segunda entrada (que viene de la izquierda) y de ese modo producir la diferencia $x - rs$ como una salida del lado derecho de la celda y (b) para transmitir la primera entrada s hacia abajo sin alteración. Considere, por ejemplo, la matriz triangular de 3 por 3 que se detalla en la Fig. 64. Esta matriz sistólica implica una combinación de células internas y límite. En este caso, la matriz triangular calcula un vector de salida y, relacionado con el vector de entrada u por:

$$y = R^{-T}.x \qquad (7)$$

donde R^{-T} es la inversa de la matriz transpuesta R^T. Los elementos de R^T son el contenido de las respectivas celdas de la matriz triangular. Los ceros agregados a las entradas de la matriz en la figura están destinados a proporcionar los retrasos necesarios para canalizar el cálculo dado por la ecuación (7). Una arquitectura de matriz sistólica, como se describe en el presente documento, ofrece las características deseables de modularidad, interconexiones locales y procesamiento paralelo sincronizado y altamente canalizado; la sincronización se logra mediante un reloj global.

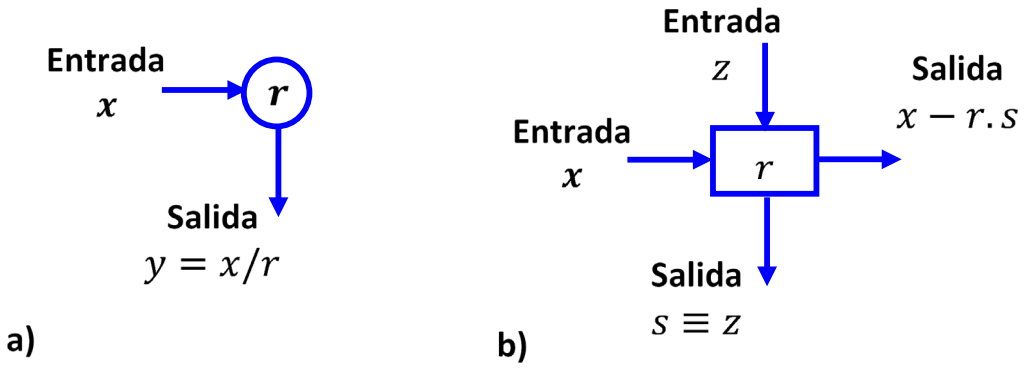


Fig. 63 – Dos celdas básicas de matriz sistólica: (a) celda límite; (b) celda interna.

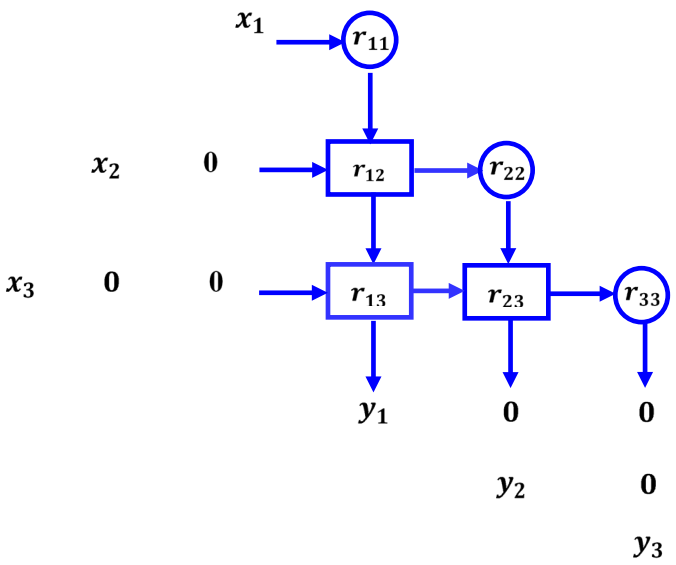


Fig. 64 – Ejemplo matriz sistólica triangular

Filtros lineales con memoria infinita (IIR)

Se observa que las estructuras del filtro clásico FIR, la estructura del predictor de celosía y la matriz sistólica triangular comparten una propiedad común: los tres se caracterizan por una respuesta de impulso de duración finita. Es decir, los filtros FIR tienen estructuras con rutas feedforward (rutas de alimentación hacia adelante).

En la Fig. 65 se muestra un ejemplo de estructura de filtro IIR. La característica que distingue a un filtro IIR de un filtro FIR es la inclusión de rutas feedback (rutas de retroalimentación o alimentación hacia atrás). De hecho, la presencia de retroalimentación hace que la duración de la respuesta impulsional de un filtro IIR sea infinitamente larga. Además, la presencia de retroalimentación introduce un nuevo problema: pueden resultar *inestables*. En particular, es posible que un filtro IIR se vuelva inestable (por ej. entre en oscilación), a menos que se tomen precauciones especiales en la elección de los coeficientes de retroalimentación. Por el contrario, los filtros de respuesta impulsional finita (FIR) siempre son estable. Esto explica el uso popular de filtros FIR, de una forma u otra, como base estructural para el diseño de filtros adaptativos lineales.

Estructura del Filtro IIR (se agrega la realimentación de la salida)

$y[n] = \sum_{k=0}^{N} b_k . x[n-k] - \sum_{i=1}^{M} a_i . y[n-i]$

Tomamos la ecuación anterior y aplicamos «Transformada Z» en ambos miembros:

$f[n \pm a] \longleftrightarrow z^{\pm a} . F(z)$; para CIN

$$Y(z) = \sum_{k=0}^{N} b_k . z^{-k} . X[n] - \sum_{i=1}^{M} a_i . z^{-k} . Y(z)$$

$$Y(z) . \left(1 + \sum_{i=1}^{M} a_i . z^{-k} . Y(z) \right) = \sum_{k=0}^{N} b_k . z^{-k} . X[n]$$

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{k=0}^{N} b_k . z^{-k} . X[n]}{1 + \sum_{i=1}^{M} a_i . z^{-k} . Y(z)}$$

$$\boxed{H(z) = \frac{\sum_{k=0}^{N} b_k . z^{-k}}{1 + \sum_{k=1}^{M} a_k . z^{-k}}}$$

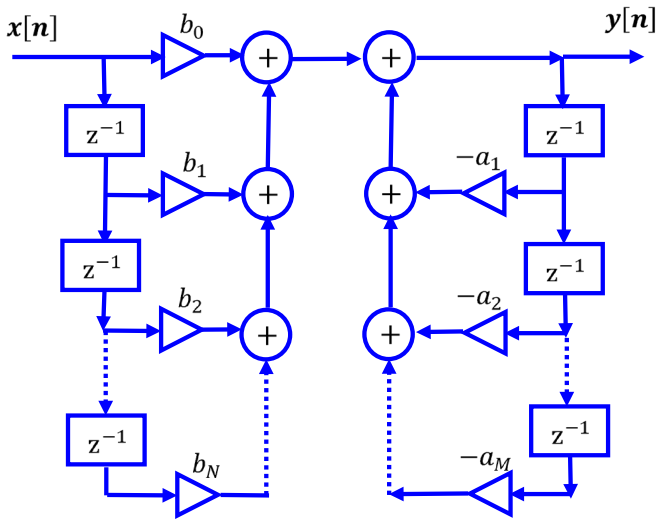


Fig. 65 – Ejemplo de estructura de Filtro IIR

En la Fig. 66 se muestra otra estructura de filtro IIR donde se redujeron la cantidad de desfasadores.

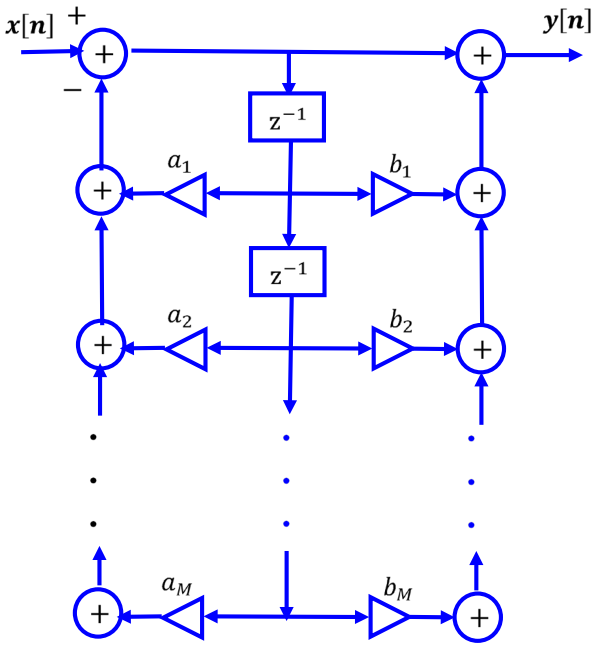


Fig. 66 – Ejemplo de estructura de Filtro IIR, reduciendo cantidad de desfasadores

Introducción al desarrollo de algoritmos de filtros lineales adaptativos

No existe una solución única para el problema del filtrado adaptativo lineal. Más bien, tenemos un "conjunto de herramientas" representado por una variedad de algoritmos recursivos, cada uno de los cuales ofrece ventajas propias. El desafío al que se enfrenta el usuario del filtrado adaptativo es, en primer lugar, comprender las capacidades y limitaciones de varios algoritmos de filtrado adaptativo y, en segundo lugar, utilizar este conocimiento en la selección del algoritmo apropiado para la aplicación en cuestión.

Básicamente, podemos identificar dos enfoques distintos para implementar algoritmos correspondientes a filtros adaptativos lineales: LMS y RLS

En la Fig. 67 se muestra un esquema básico de un sistema adaptativo.

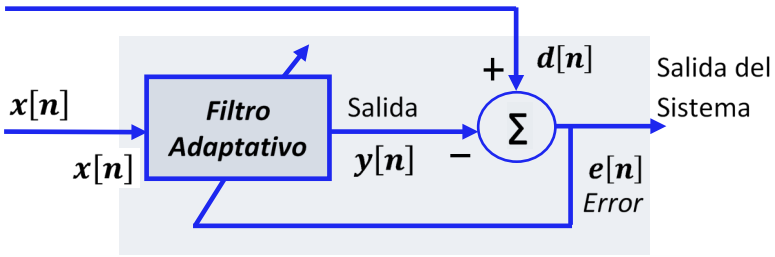


Fig. 67 – Sistema adaptativo

$x[n]$: entrada del filtro

$y[n]$: salida del filtro

$e[n]$: error generado

$d[n]$: función deseada

El filtro, después de un tiempo que realice varias iteraciones, deberá generar una aproximación a la función deseada $d[n]$. Se debe reducir el error $e[n]$. En filtros adaptativos se utiliza mucho minimizar el error cuadrático medio (MSE) mediante la siguiente función de costo:

$$J = E\{e^2[n]\} = E\{d^2[n] - 2.d[n].y[n] + y^2[n]\}$$

El filtro del sistema adaptativo podría ser un filtro IIR, filtro lattice, un combinador lineal de entradas adaptativo u otra estructura. El filtro FIR es el más utilizado, en este libro analizaremos este filtro, escribimos su ecuación característica con coeficientes fijos:

$$y[n] = \sum_{k=0}^{N-1} b_k.x[n-k]$$

Los filtros adaptativos buscan hallar los valores más adecuados de los coeficientes w_k, de manera de minimizar la salida de error $e[n]$ del sistema adaptativo. Estos coeficientes se actualizan a medida que va cambiando la entrada al sistema. En la Fig. 68 se muestra un esquema del Filtro Adaptativo FIR. La salida del filtro $y[n]$ se expresa mediante la siguiente ecuación:

$$y[n] = \sum_{i=0}^{N} w_i[n].x[n-i] = w^T[n].x[n]$$

Siendo $x[n] = \left[x[n],\ x[n-1], \ldots \ldots x[n-N]\right]^T$

$w[n] = \left[w_0[n],\ w_1[n], \ldots \ldots w_N[n]\right]^T$

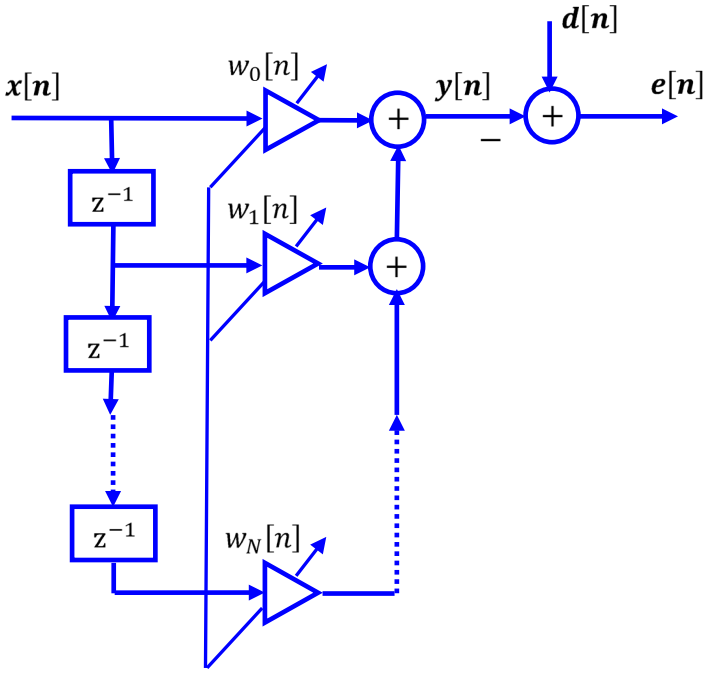


Fig. 68 – Filtro Adaptativo FIR

La señal de entrada es $x[n]$. La señal de error es $e[n]$ resulta de la resta entre la señal deseada $d[n]$ y la señal de salida del filtro $y[n]$.

La salida del filtro se obtiene mediante:

$$y[n] = w^T[n].x[n] \qquad (4)$$

Siendo $w[n]$ los coeficientes del filtro

Se obtiene la Ecuación para el cálculo de la señal de error:

$$e[n] = d[n] - y[n] \qquad (5)$$

Generalmente se trata de que la señal $e[n]$ sea cero, para esto a partir de la señal $x[n]$ se genera $y[n]$ y se van ajustando los coeficientes del filtro para que $y[n]$ sea igual a la señal esperada $d[n]$.

Se disponen de distintos métodos pata minimizar la señal de error $e[n]$. Por ejemplo, en caso de conocer la estadística de las señales, se puede minimizar la siguiente función: $J = E\{|e[n]|^2\}$. O en caso de no conocer la estadística, se puede simplificar minimizando la función: $J_s = |e[n]|^2$. Mediante el algoritmo de gradiente descendente, se obtiene:

$$w[n+1] = w[n] - \frac{\mu}{2}.\nabla J \qquad (6)$$

Siendo μ una constante que modifica la velocidad de convergencia del algoritmo.

Mediante el método LMS (Least Mean Square), se tiene la siguiente ecuación:

$$w[n+1] = w[n] + \mu.e[n].x[n] \qquad (7)$$

A modo de ejemplo, para $\mu = 0{,}01$ y filtro de un solo coeficiente se tiene:

$$w[n+1] = w[n] + 0{,}01.x[n].e[n] \ ;$$

Siendo $y[n] = w[n].x[n] \ ; \ \ e[n] = d[n] - y[n]$

Método de gradiente estocástico descendente

El enfoque de gradiente estocástico utiliza una línea de retardo con derivación, o filtro FIR, como base estructural para implementar el filtro adaptativo lineal. Para el caso de señales estacionarias, la función de costo J, también conocida como índice de desempeño, se define como el error cuadrático medio (es decir, se toma el valor cuadrático medio de la diferencia o resta entre la señal deseada y la señal de salida del filtro FIR). Esta función de costo es precisamente una función de segundo orden de los pesos de los bloques del filtro FIR. La dependencia del error cuadrático medio de los pesos desconocidos de los bloques puede verse en forma de un paraboloide multidimensional (es decir, un "tazón de ponche" punch bowl) con un fondo definido de forma única o un punto mínimo. Como se mencionó anteriormente, nos referimos a este paraboloide como la superficie de comportamiento del error; los pesos de los bloques correspondientes al punto mínimo de la superficie definen la solución óptima de Wiener.

Para desarrollar un algoritmo recursivo con el fin de actualizar los pesos del filtro FIR adaptativo utilizando el enfoque de gradiente estocástico, como su nombre lo implica, debemos comenzar con una función de costo estocástico. Para tal función, por ejemplo, podemos usar el valor instantáneo al cuadrado del error (señal de error), el error es la diferencia entre la respuesta o señal deseada suministrada externamente y la respuesta real del filtro FIR a la señal de entrada.

$$J_s = |e[n]|^2$$

Luego, al derivar esta función de costo estocástico con respecto al vector de pesos de del filtro, obtenemos un vector de gradiente que es naturalmente estocástico. Con el deseo de avanzar hacia la optimización, la adaptación se realiza a lo largo de la dirección "negativa" del vector de gradiente.

En la Fig. 69 se muestra la función de error cuadrático medio J en función de w para el caso de uno y 2 coeficientes.

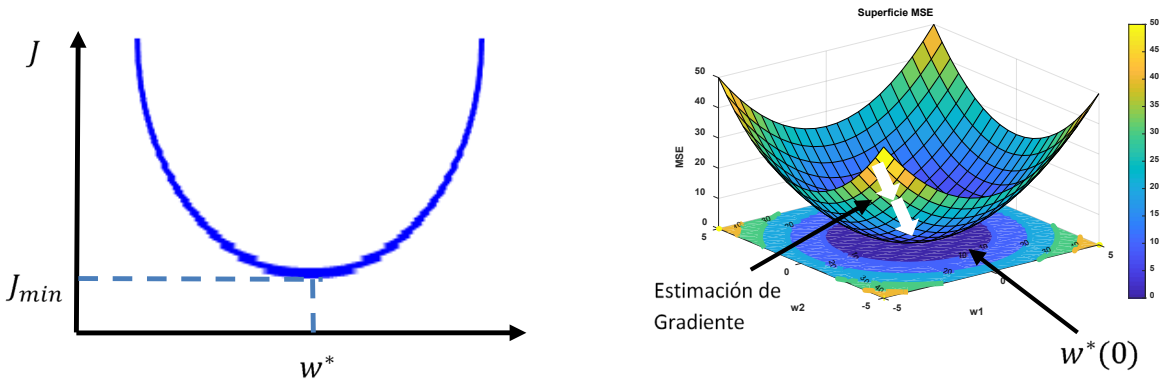


Fig. 69 – Función de error cuadrático medio J en función de w. Izquierda) 1 coeficiente derecha) 2 coeficientes

LMS (Least Mean Squares)

El algoritmo LMS se desarrolló por Widrow y Hoff (Widrow &Hoff, 1960). Este algoritmo de filtrado adaptativo resulta del enfoque de pasos descendentes, puede expresarse en palabras de la siguiente manera:

[Vector de pesos actualizados] = [Vector de pesos anterior] + [Parámetro de tasa de aprendizaje] x [Vector de entrada] x [señal de error]

El parámetro de tasa de aprendizaje determina la tasa a la que se realiza la adaptación. El algoritmo recursivo así descrito se denomina algoritmo de mínimos cuadrados medios (LMS: Least Mean Squares), que tiene baja carga computacional y es eficaz en rendimiento. Sin embargo, su comportamiento de convergencia es lento y difícil de estudiar matemáticamente.

A continuación, se muestra el cálculo del MSE

$$y[n] = w^T[n].x[n] \qquad (8)$$

$$e[n] = d[n] - y[n] = d[n] - w^T[n].x[n] \qquad (9)$$

$$e^2[n] = (d[n] - w^T[n].x[n])^2 \qquad (10)$$

Teniendo en cuenta:

$J = E\{e^2[n]\}$: MSE, es el error cuadrático medio

$\sigma^2 = E\{d^2[n]\}$: autocorrelación de señal deseada

$p = E\{d[n].x[n]\}$: correlación cruzada entre $d[n]$ y $x[n]$

$R = E\{x[n].x^T[n]\}$: autocorrelación de la entrada $x[n]$

Desarrollando obtenemos:

$$\boxed{J = \sigma^2 - 2.w^T.p + w^T.R.w} \qquad (11)$$

Se desea obtener el valor del peso ideal, es decir el valor de w que minimice el error J. Para el caso de un coeficiente, se calcula la derivada de J y se iguala a 0.

$$\nabla J_w = \frac{\partial J}{dw} = -2.p + 2.R.w \qquad (12)$$

$$\frac{\partial J}{dw} = -2.p + 2.R.w = 0 \qquad (13)$$

$$R.w = p \qquad (14)$$

$$w_o = w_{optima} = R^{-1}.p \qquad (15)$$

Se obtuvo w_{optima} que es la solución óptima de Wiener

Los pesos se actualizan en cada iteración, el nuevo w, es decir $w[n+1]$ debe ser el w anterior: $w[n]$ al cual se le resta/suma un porcentaje de $\hat{g}_w(n)$ que lleve a la función al valor mínimo de J. Siendo μ una constante que modifica la velocidad de convergencia que tiene el algoritmo

$$w[n+1] = w[n] - \mu.\hat{g}_w[n] \qquad (16)$$

$$\hat{g}_w[n] = \hat{\nabla} J_w$$

Desarrollando obtenemos la siguiente solución conocida como LMS (Least Mean Square):

$$\boxed{w[n+1] = w[n] + \mu.2.e[n].x[n]} \qquad (17)$$

Otra nomenclatura

Partiendo de la ecuación utilizada en LMS para un filtro FIR se puede escribir la ecuación de la siguiente forma:

$w_{n+1} = w_n + \mu.2.e[n].x[n]$

Para un filtro FIR de orden n, se extiende el algoritmo LMS de la siguiente forma:

$w_{n+1}[i] = w_n[i] + \mu.2.e[n].x[n-i]$

Siendo $i = 0,1,\dots\dots,N$; varios pesos

$$y[n] = w_n[0].x[n] + w_n[1].x[n-1] + \cdots \dots . + w_n[N-1].x[n-N+1]$$

Nota: Para hallar la última ecuación se tiene en cuenta la respuesta del filtro FIR: $y[n] = \sum_{k=0}^{N} b_k.x[n-k]$, pero para sistemas adaptativos se utilizan los coeficientes $w_n[i]$ en vez de b_k. Así se expresa que w_n varía en función de las iteraciones del algoritmo.

Método de Mínimos Cuadrados Recursivos (RLS: Recursive Least Squares)

El segundo enfoque para desarrollo de algoritmos de filtrado adaptativo lineal se basa en el método de cuadrados mínimos. De acuerdo con este método, se minimizan la suma de los errores cuadráticos correspondientes a las diferencias entre la respuesta deseada y la señal de salida real del filtro.

Resulta diferente al método del gradiente estocástico, esta minimización se logra mediante manipulaciones de matrices algebraicas, lo que resulta en una regla de actualización que puede expresarse en palabras, de la siguiente manera:

[Vector de pesos actualizados] = [Vector de pesos anterior] + [Vector de ganancia] x [Innovación]

Donde la innovación es información "nueva" puesta en el proceso de filtrado en el momento de su actualización. El algoritmo de filtrado adaptativo descripto de esta forma se llama algoritmo de *mínimos cuadrados recursivos (RLS)*. Una propiedad distintiva de este segundo algoritmo es su rápida tasa de convergencia, que se logra a expensas de una mayor complejidad computacional.

Cuando se reciben muestras nuevas de las señales entrantes en cada iteración, la solución para de mínimos cuadrados se puede calcular en forma recursiva, lo que da como resultado los algoritmos de mínimos cuadrados recursivos (RLS). Se sabe que los algoritmos RLS persiguen una convergencia rápida incluso cuando la dispersión del valor propio que tiene la matriz de correlación de la señal de entrada es grande. Estos algoritmos pueden tener un rendimiento competitivo cuando se trabaja en entornos que varían en el tiempo, según modelos de parámetros desconocidos. Todas estas ventajas vienen con el costo de una mayor complejidad computacional y algunos problemas de estabilidad, que no son tan críticos en los algoritmos basados en LMS.

En el algoritmo RLS, los coeficientes se actualizan a medida que va cambiando la entrada al sistema. El proceso de minimización de error requiere la información de la señal de entrada disponible hasta el momento. Además, la función objetivo que buscamos minimizar es determinística: $\varepsilon^d[k]$

$$\varepsilon^d[k] = \sum_{i=0}^{k} \lambda^{k-i}.\varepsilon^2[i]$$

$$\varepsilon^d[k] = \sum_{i=0}^{k} \lambda^{k-i}.\ \left[d[i] - x^T[i].w[k]\right]^2$$

$w[k]$ son los coeficientes del filtro adaptativo

$\varepsilon[i]$ es el error «a posteriori» en instante i

λ es un factor de pesos exponencial: $0 \ll \lambda < 1$. También llamado forgetting factor (factor de olvido)

λ también se llama factor de olvido

$\varepsilon[k]$ es el error «a posteriori»

Al minimizar el error determinístico, obtenemos:

$$\boxed{p_D[k] = \sum_{i=0}^{k} \lambda^{k-i}.x[i].d[i]}$$

Se utiliza la información previa, teniendo una matriz de correlación determinística $R_D[k]$

${R_D}^{-1}[k]$: Matriz Inversa de correlación determinística

$p_D[k]$: matriz de correlación cruzada entre x y d

Mediante el "Lema de Inversión de Matrices" se estima ${\mathrm{R_D}}^{-1}[\mathrm{k}]$ de la siguiente forma:

$$\boxed{S_D[k] = {R_D}^{-1}[k] = \frac{1}{\lambda}.\left[S_D[k-1] - \frac{S_D[k-1].x[k].x^T[k].S_D[k-1]}{\lambda + x^T(k).S_D[k-1].x[k]}\right]}$$

La Matriz $S_D[k]$ se inicializa como matriz diagonal. En varias iteraciones se va modificando dicha matriz para reducir el error

$R_D{}^{-1}[k]$ resulta menos compleja de calcular (de orden N^2 multiplicaciones) que la inversión directa de $\hat{R}[k]$ (de orden N^3 multiplicaciones)

$$\boxed{w[k] = R_D{}^{-1}[k].p_D[k]}$$

En los resultados anteriores se utiliza error a posteriori

También se puede utilizar el error a priori, que corresponde al algoritmo RLS Alternativo

$$e[k] = d[k] - x^T[k].w[k-1]$$

Manipulando matrices se obtiene la siguiente expresión:

$$w[k] = w[k-1] + e[k].S_D[k].x[k]$$

En el método RLS la Innovación es información "nueva" puesta en el proceso de filtrado en el momento de su actualización de coeficientes.

RLS presenta las siguientes ventajas: Rápida tasa de convergencia. Incluso con dispersiones grandes de matriz de correlación. Buen rendimiento en entornos que varían con el tiempo con parámetros desconocidos

Desventajas de RLS: Mayor complejidad computacional. Algunos problemas de estabilidad (LMS no es tan crítico). Es importante tener muestras nuevas de las señales entrantes en cada iteración

Filtros no lineales adaptativos

Los filtros lineales están basados en el criterio de error cuadrado mínimo. El filtro de Wiener resulta en la minimización de este criterio y plantea como objetivo resolver el filtrado lineal adaptativo en un ambiente estacionario, solo para estadísticas de segundo orden de la entrada, y no superiores. El estadístico para un orden kº es igual al valor k-ésimo menor de una muestra estadística. Esta condición de estadístico de orden 2, limita la habilidad de los filtros lineales adaptativos de extraer información desde una entrada de datos no Gaussiana. A pesar de la importancia teórica, se cuestiona la existencia de ruido Gaussiano (Johnson y Rao, 1990). Muchos procesos No Gaussianos son bastante comunes en el procesamiento de señales. La utilización de filtros de Wiener y de filtros adaptativos lineales en presencia de ruido no Gaussiano puede resultar ser una solución no óptima. Debido a estas limitaciones, se incorporan estructuras no lineales en los filtros adaptativos, para tener en cuenta órdenes superiores estadísticos. Así también, al no tener un filtro de Wiener de referencia, se complica el análisis matemático, donde se espera un aprendizaje eficiente en los algoritmos y tener amplio uso en diferentes áreas de aplicación.

Los sistemas lineales que trabajan con filtrado adaptativo consisten en calcular y adaptar los coeficientes de los filtros lineales, generalmente trabajan en tiempo real. Estos algoritmos se utilizan en muchas aplicaciones en donde las señales medidas pueden modelarse como ruidos gaussianos aplicados a sistemas lineales, y sus combinaciones son de tipo aditivo. En las comunicaciones digitales los sistemas adaptativos lineales modelan correctamente las señales. Por ejemplo, el ruido de canal se considera ruido gaussiano aditivo, las interferencias entre símbolos también se consideran de tipo aditivo, y se supone que los modelos son filtros lineales selectivos en frecuencia. Estos modelos son precisos, y a velocidades medias o bajas filtran correctamente las señales.

Sin embargo, para velocidades altas de comunicación los modelos lineales no son válidos. La compresión de señales, la saturación del amplificador, la interacción multiplicativa entre las señales gaussianas y el filtrado no lineal de las señales gaussianas son fenómenos típicos que se producen en los sistemas de comunicación que no pueden modelarse bien con los sistemas adaptativos lineales. Además, si la función de transferencia de canal no tiene una fase mínima y/o la relación señal/ruido no es lo suficientemente alta, colocar un ecualizador con filtrado adaptativo lineal tendría un rendimiento muy bajo teniendo en cuenta la de tasa de error de bits. Un gran inconveniente de tratar con modelos no lineales es la falta de herramientas matemáticas que, por otro lado, están ampliamente disponibles para los modelos lineales. La falta de herramientas analíticas se origina en los altos grados y dimensionalidad del comportamiento no lineal. El rendimiento mejorado del ecualizador no lineal se justifica en muchas simulaciones disponibles en la bibliografía, donde la tasa de error se utiliza como medida de rendimiento.

En la Fig. 70 se muestra un esquema básico de filtro no lineal. Existen muchas técnicas para de filtros adaptativos no lineales. A continuación, enunciamos solamente algunas técnicas importantes:

a) Modelo polinomial no recursivo basado en la expansión de la serie Volterra.
b) Modelo polinomial recursivo basado en ecuaciones en diferencias no lineales.
c) Red neuronal perceptrón multicapa (MLP).
d) Red neuronal con función de base radial (RBF).

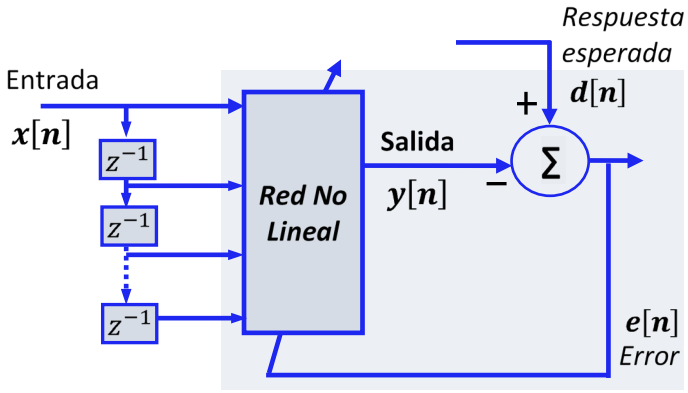


Fig. 70 – Filtro Adaptativo No lineal

Filtros Adaptativos No Lineales basados en Volterra

En este tipo de filtros la no linealidad se encuentra en el extremo final del filtro. Se utiliza la serie de Volterra que provee un método atractivo para describir la relación entre la entrada-salida de un dispositivo no lineal con memoria. El nombre surge debido a que fue estudiado por primera vez por Vito Volterra alrededor del año 1880, es una generalización de la Serie de Taylor de una función. Sin embargo, Norbert Wiener (1958) fue el primero en utilizar la serie de Volterra para modelar la relación que tiene la entrada y salida de un sistema no lineal.

Para una serie temporal x_n denota la entrada de un sistema no lineal de tiempo discreto. Se puede utilizar un conjunto de estas entradas para definir un conjunto de núcleos discretos de Volterra de la siguiente manera:

H_0: término de orden 0 (dc)

$H_1[n] = \sum_i h_i . x_i$; $H_1[n]$: término de primer orden (lineal)

$H_2[n] = \sum_i \sum_j h_{ij}.x_i.x_j$; $H_2[n]$: término de segundo orden (cuadrático)

$H_3[n] = \sum_i \sum_j \sum_k h_{ijk}.x_i.x_j.x_k$; $H_3[n]$: término de tercer orden (cúbico)

Y así continua para términos de mayor orden. Generalmente, el coeficiente $h's$, se fija por métodos analíticos.

Un filtro no lineal compuesto por los siguientes bloques:

- Un expansor de estados no lineal Volterra que combina un conjunto de valores de entradas $x_0, x_1, \ldots . x_n$ que produce un conjunto de salidas $u_0, u_1, \ldots . u_q$, donde q es mayor a n. Por ejemplo, la extensión del vector de un sistema puede tener la siguiente forma:

  $u = [1,\ x_0, x_1, x_2,\ {x_0}^2, x_0.x_1,\ x_0.x_2,\ {x_1}^2,\ x_1.x_2,\ {x_2}^2]^T$

- Un filtro lineal adaptativo tipo FIR que utiliza como entradas u_k (elementos de u) para generar una estimación $\widehat{d_n}$ de la respuesta deseada d_n
- Un sumador

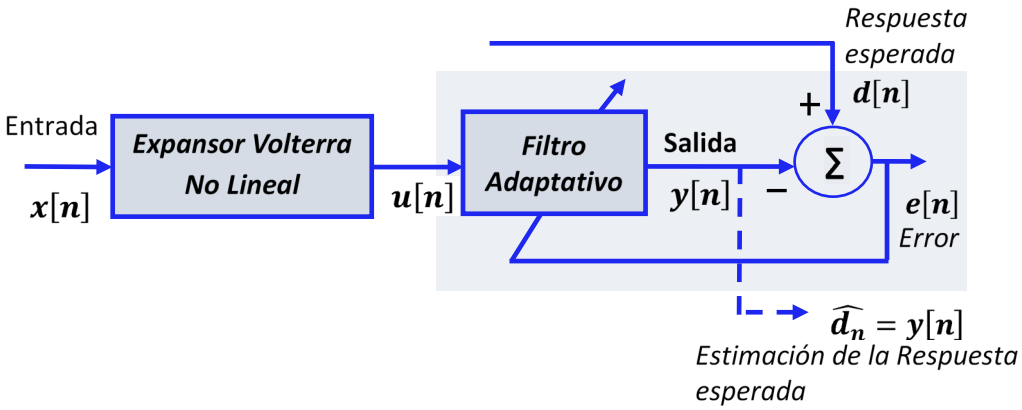


Fig. 71 – Filtro Adaptativo no lineal basado en Volterra

La idea importante para resaltar es que mediante el uso de un esquema similar al de la figura anterior, podemos extender el uso de filtros lineales adaptativos incluyendo los filtros de Volterra

Redes Neuronales

Los sistemas neuronales artificiales (ANS) imitan la estructura del sistema nervioso, construyendo sistemas de procesamiento de información que trabajan en paralelo, distribuidos y adaptativos, con cierto comportamiento "inteligente". El cerebro y la computadora resultan mucho más diferentes de lo que suele suponerse cuando se habla de cerebros electrónicos. El elemento esencial de los ANS es la neurona artificial, organizada en capas; la red neuronal está formada por varias capas. Una red neuronal o un conjunto de redes, junto con las interfaces necesarias, forman el sistema global de procesamiento.

En la Fig. 72 se tiene un modelo estándar de neurona artificial, se indica su analogía con una neurona biológica. Consiste en las siguientes partes:

- Un conjunto de entradas $x_j(t)$ y pesos sinápticos w_{ij}
- Una regla de propagación $h_i(t) = \sum w_{ij} x_j$ (la más común)
- Una función de activación $y_i(t) = f_i(h_i(t))$, que representa la salida de la neurona

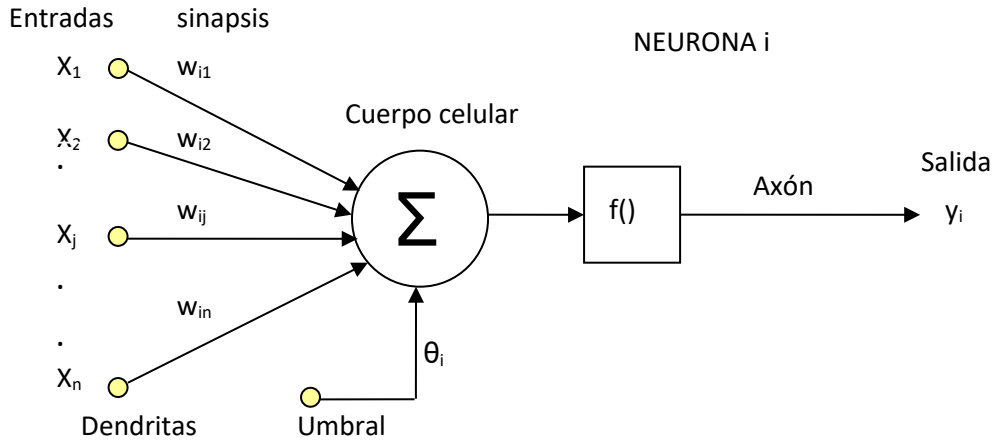


Fig. 72 – Modelo de neurona estándar

La red neuronal artificial, o también llamada red neuronal se diseña con la interconexión de muchas unidades de procesamiento con comportamiento no lineal llamadas neuronas, obteniendo bloques no lineales distribuidos a través de toda la red. El diseño de redes neuronales desde el inicio está motivado por la forma en que trabaja el cerebro humano, por este motivo se llaman neuronales. La sinapsis, las dendritas y el axón representan la analogía que se tiene con la neurona biológica.

Se denomina arquitectura de red neuronal a la estructura de conexionado que tiene la red. En una red neuronal los nodos se conectan entre sí por medio de sinapsis, con conexiones sinápticas unidireccionales, es decir, la información solo se propaga en un sentido. Las neuronas se suelen agrupar en capas o unidades estructurales.

En las redes neuronales existen tres tipos de capas: capa de entrada, oculta y de salida. La capa sensorial o de entrada está formada por neuronas encargadas de recibir datos o señales (por ejemplo, provenientes de sensores o de realimentación de las salidas). La capa de salida contiene neuronas que entregan la respuesta de toda la red neuronal. En cambio, la capa oculta no se conecta directamente con el entorno. Estas capas agregan a la red neuronal mayor cantidad de grados de libertad, flexibilizando las respuestas que pude brindar la red.

Aplicaciones de filtros adaptativos

A continuación, se enuncian 4 clases de aplicaciones de Filtros adaptativos:

- **Sistema de identificación**
- **Sistema de identificación con Modelado Inverso**
- **Predicción**
- **Cancelación de Ruidos, Interferencias y ecos**

La capacidad de un filtro adaptativo para operar satisfactoriamente en un entorno desconocido y rastrear las variaciones temporales de las estadísticas de entrada convierte al filtro adaptativo en una herramienta poderosa para aplicaciones de control y para el procesamiento avanzado de señales. De hecho, los filtros adaptativos se han aplicado con éxito en campos tan diversos como comunicaciones, control, radar, sonar, sismología e ingeniería biomédica. Aunque estas aplicaciones son de naturaleza bastante diferente, tienen una característica básica en común: un vector de entrada y una respuesta deseada que se utilizan para calcular un error de estimación, que a su vez se utiliza para controlar los valores de un conjunto de coeficientes de filtro ajustables. Los coeficientes ajustables pueden tomar la forma de pesos, coeficientes de reflexión o parámetros de rotación, dependiendo de la estructura de filtro empleada. Sin embargo, la diferencia esencial entre las diversas aplicaciones del filtrado adaptativo surge en la forma en que se extrae la respuesta deseada. En este contexto, podemos distinguir cuatro clases básicas de aplicaciones de filtrado adaptativo. En la Fig. 73 se muestra un esquema básico de filtro adaptativo. Se utiliza la siguiente notación:

x: entrada aplicada al filtro adaptativo;

y: salida del filtro adaptativo;

d: respuesta deseada;

$e = d - y$: error de estimación.

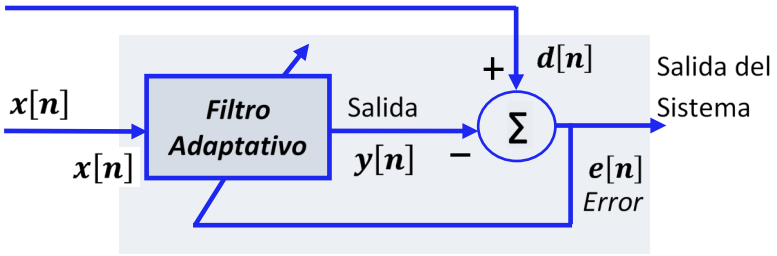


Fig. 73 – Sistema adaptativo

Existen cuatro clases de básicas de aplicaciones en el filtrado adaptativo, se identifican con las siguientes funciones:

I. Identificación.

El filtro adaptativo se puede utilizar para modelado o identificación de sistemas. El filtro rastrea el comportamiento de un sistema desconocido utilizando la entrada y la salida del sistema desconocido. En las Fig. 74 se muestra un esquema adaptativo de identificación de sistemas y en la Fig. 75 se muestra mismo esquema con menos detalles. La identificación de sistemas y la obtención de un modelo matemático es fundamental para la ciencia y la ingeniería. En la clase de aplicaciones que se ocupan de la identificación, se utiliza un filtro adaptativo para proporcionar un modelo lineal que representa el mejor ajuste (en cierto sentido) a una planta desconocida. La planta (sistema

desconocido) y el filtro adaptativo son impulsados por la misma entrada. La salida de la planta proporciona la respuesta deseada para el filtro adaptativo. Si la planta es de naturaleza dinámica, el modelo variará en el tiempo.

En la aplicación de identificación del sistema, la señal deseada es la salida del sistema desconocido cuando es excitada por una señal de banda ancha, en la mayoría de los casos una señal de ruido blanco. La señal de banda ancha también se utiliza como entrada para la adaptación.

Luego de varias iteraciones la salida $y[n]$ va a ser similar a la salida del sistema desconocido $y[n]$. Teniendo en cuenta que el filtro adaptativo y el sistema desconocido tienen la misma entrada, la función de transferencia que presenta el filtro adaptativo será similar a la del sistema desconocido.

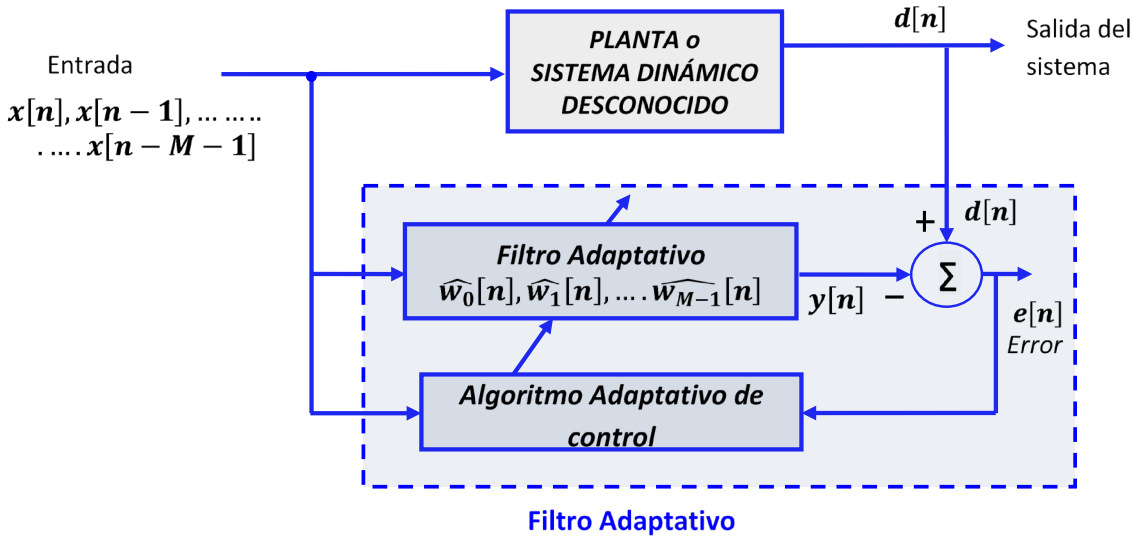


Fig. 74 – Identificación de Sistemas, esquema detallado

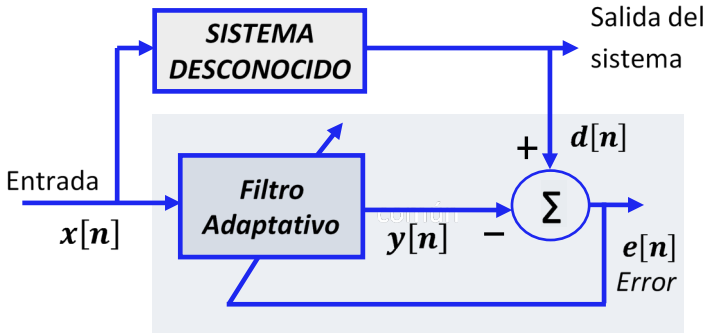


Fig. 75 – Identificación, modelo I

Otro esquema de Identificación se tiene en la Fig. 76. En este caso se toma como señal deseada la salida del sistema desconocido $M(z)$. Se aplica ruido blanco en la entrada, después que el filtro adaptativo minimice el error, la salida de todo el sistema será de nuevo ruido blanco. El bloque $D(z)$ se comporta como inverso del sistema desconocido, entonces los polos de $D(z)$ serán los ceros de $M(z)$ y viceversa.

Se tienen buenos resultados cuando la serie temporal tiene un espectro con picos altos y angostos.

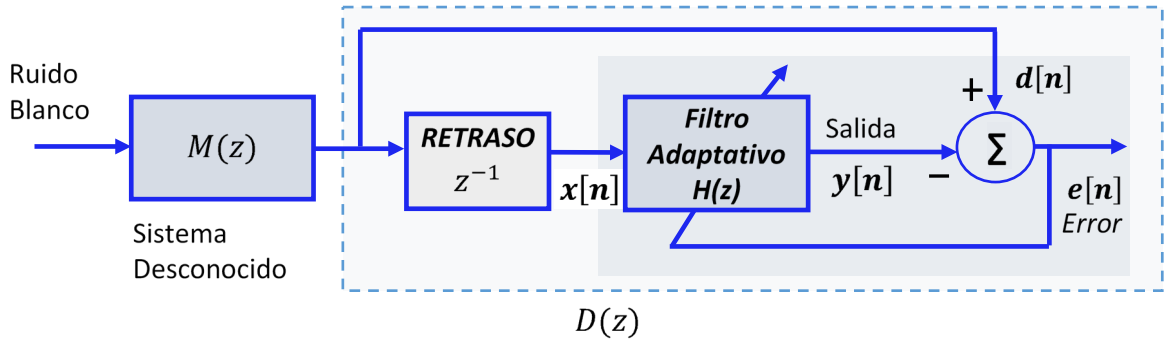


Fig. 76 – Identificación, modelo II

II. Modelado inverso.

En la Fig. 77 se muestra un esquema básico de esta clase de aplicaciones, la función del filtro adaptativo es proporcionar un modelo inverso que represente el mejor ajuste (en cierto sentido) a una planta ruidosa desconocida. Idealmente, para el caso de un sistema lineal, el modelo inverso presenta una función de transferencia que es igual a la recíproca (inversa) de la función de transferencia de la planta, de manera que la combinación de los dos se comporta como un medio de transmisión ideal. Una versión retrasada de la entrada de la planta (sistema) constituye la respuesta deseada para el filtro adaptativo. En algunas aplicaciones, la entrada de la planta se utiliza sin demora como la respuesta deseada.

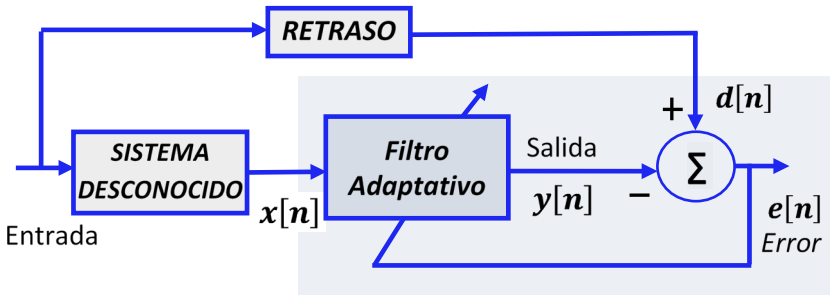


Fig. 77 – Identificación mediante Modelado Inverso

También se puede tener el esquema anterior, pero sin el bloque de retraso, como se muestra en la Fig. 78. La salida del sistema desconocido $P(z)$, o planta, ingresa al bloque del modelo $H(z)$. En este caso, al minimizar la señal de error el sistema desconocido $P(z)$ y el modelo se cancelan uno con otro. Es decir que $H(z)$ actua como modelo inverso. Este modelo solo funciona si el sistema desconocido P(z) solo tiene polos en su función de transferencia, y el modelo $H(z)$ tiene el mismo tamaño que $P(z)$. Se recomienda agregar ruido a la señal de entrada, de manera que se ajuste correctamente el modelo $H(z)$

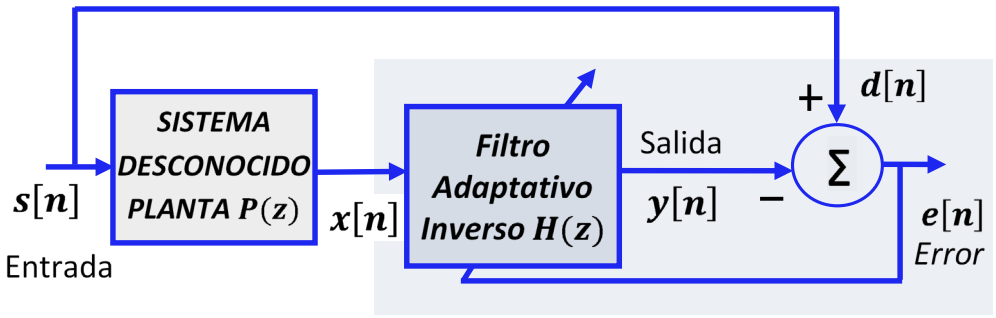


Fig. 78 – Identificación mediante Modelado Inverso

III. Predicción

Aquí, la función del filtro adaptativo es proporcionar la mejor predicción (en cierto sentido) del valor actual de una señal aleatoria. En la Fig. 79 se muestra un esquema básico de predicción. El valor actual de la señal sirve así para una respuesta deseada para el filtro adaptativo. Los valores pasados de la señal son las entrada aplicadas al filtro. Dependiendo de la aplicación de interés, se puede utilizar como salida del sistema la salida del filtro adaptativo $y[n]$) o el error de estimación $e[n]$. En el primer caso, el sistema funciona como predictor; en el segundo caso, funciona como filtro de error de predicción.

En el caso de predicción, la señal deseada es una versión hacia adelante de la señal de entrada del filtro adaptativo como se detalla en la Fig. 79. Después de la convergencia, el filtro adaptativo representa un modelo para la señal de entrada y se puede utilizar como modelo predictor de la señal de entrada.

En el caso de predicción, se supone que la estructura de la serie temporal de entrada no se modifica, o lo hace muy lentamente, de manera que el sistema pueda adaptarse a los cambios.

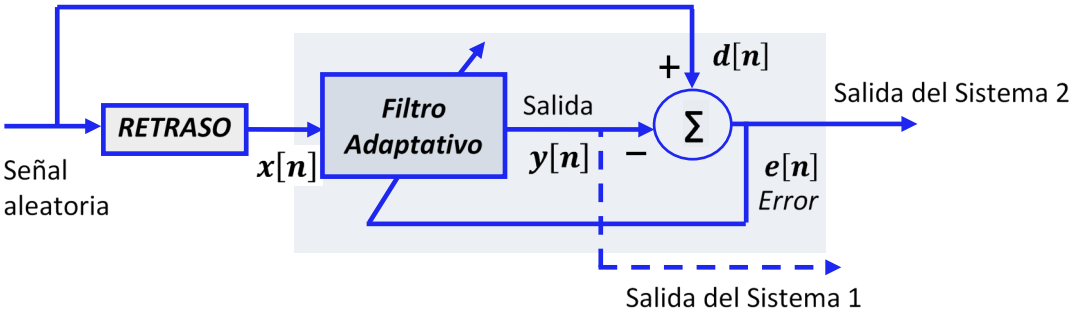


Fig. 79 – Predicción

En sistemas de transmisión se suele utilizar un impulso para analizar la respuesta del canal, en la Fig. 80 se muestra un diagrama en bloques

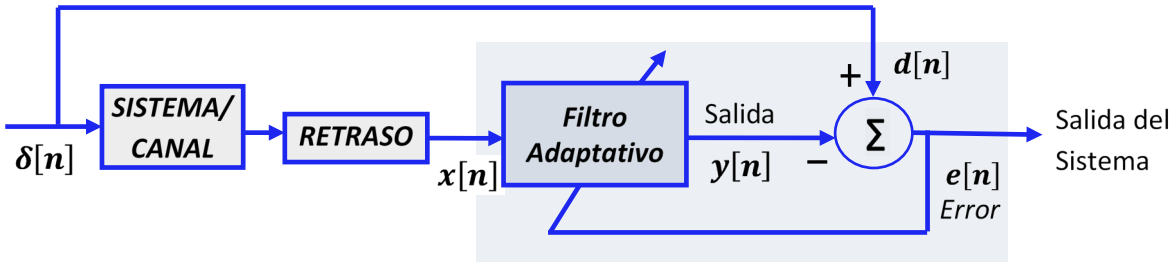


Fig. 80 – Filtro adaptativo para estudios de sistemas de transmisión. Se estudia la respuesta impulsional

IV. Cancelación de interferencias y ruidos.

Los filtros adaptativos se utilizan mucho para eliminar interferencias y ruido en sistemas eléctricos y mecánicos. Tiene aplicaciones interesantes cuando la señal de entrada esta perturbada por una interferencia de banda estrecha, como por ejemplo ruido de 50 Hz en la línea eléctrica. Resulta muy útil cancelar interferencia en aplicaciones con señales biológicas, estas señales son muy débiles, y el cuerpo humano y los equipos de medición actúan como antena de las interferencias

En la Fig. 81 se muestra un esquema básico de cancelación de ruidos e interferencias. Se utiliza para cancelar la interferencia desconocida contenida (junto con un componente de señal portadora de información) en una señal primaria, optimizándose la cancelación en algún sentido. La señal primaria se utiliza como la respuesta deseada para el filtro adaptativo. Como entrada el filtro se utiliza ***una señal de referencia (auxiliar)***. La señal de referencia se deriva de un sensor o de un conjunto de sensores ubicados de tal manera que suministran la señal de ruido, de tal manera que la señal principal de información es débil o esencialmente indetectable.

Se cancela ruido a la entrada de un sistema, como se observa en la Fig. 81. La señal s esta contaminada con ruido r. En el filtro se ingresa solamente el ruido r', este ruido esta correlacionado con el ruido r que tiene tiene la señal s. Cuando el sistema ajusta sus parámetros, la salida y se asemeja al ruido aditivo de s , y **la señal de error se asemeja a la entrada s**. Este sistema funciona bien cuando el ruido no está correlacionado con la señal de entrada s, obteniendo una cancelación de ruido en dicha señal.

Es decir que en la salida del filtro se desea estimar el ruido real $r[n]$, para luego restarlo a la señal $d[n]$ que en este caso es la señal de información con ruido: $s[n] + r[n]$. De este modo se tiene una aproximación de la señal $s[n]$ sin ruido. Para un filtro FIR de orden 1 (un solo retraso) se tiene: $y[n] = w_0[n].x[n]$

Cuando $y[n] \cong \hat{r}[n]$, la señal de error resulta $e[n] = s[n] + r[n] - \hat{r}[n] \cong \hat{s}[n]$

Es decir, se obtiene una señal $\hat{s}[n]$ limpia sin ruido, que resulta similar a $s[n]$

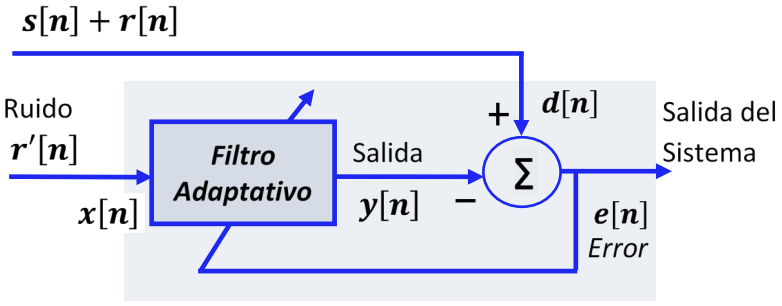


Fig. 81 – Cancelación de interferencias y ruido a la entrada

Como ejemplo de cancelación de ruido y mejora de la señal se toma una señal triangular con ruido blanco. En la entrada del filtro adaptativo se ingresa ruido blanco correlacionado con el ruido que contiene la señal triangular. Mediante varias iteraciones, los coeficientes del filtro se ajustan de manera de minimizar el error cuadrático medio. Se observa que a medida que aumenta la cantidad de iteraciones, la señal de error se asemeja más a la señal triangular, ver Fig. 82

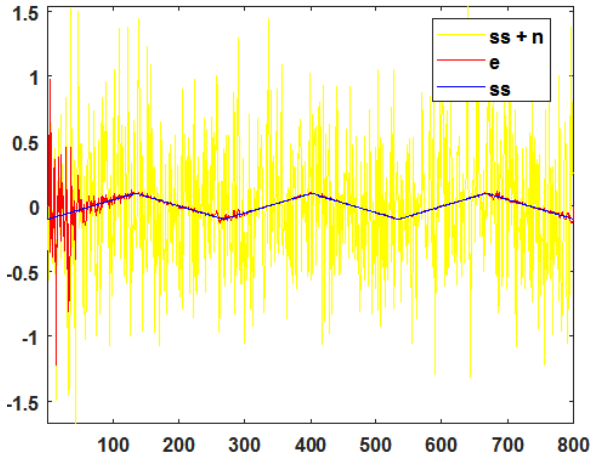


Fig. 82 – Ejemplo de cancelación de ruido. Señal de error (línea roja) y señal triangular (línea azul)

En la Fig. 83 se muestra un ejemplo de cancelación de ruido, similar a la Fig. 81, pero con el agregado de sensores. El ruido captado en ambos sensores tiene que provenir de la misma fuente, debe estar correlacionado. En este caso se desea mejorar una señal $s[n]$ corrompida por ruido, esta señal ingresa al sensor primario o principal, obteniendo $s[n] + r[n]$. También se dispone de un sensor secundario de referencia, donde se mide una señal correlacionada con el ruido $r'[n]$. Mediante el filtro adaptativo se busca modelar la señal de ruido para esta distorsión a la señal del sensor primario. Después de la convergencia, el error de salida será una señal mejorada respectos de la señal de entrada. La figura ilustra una configuración típica de mejora de señal.

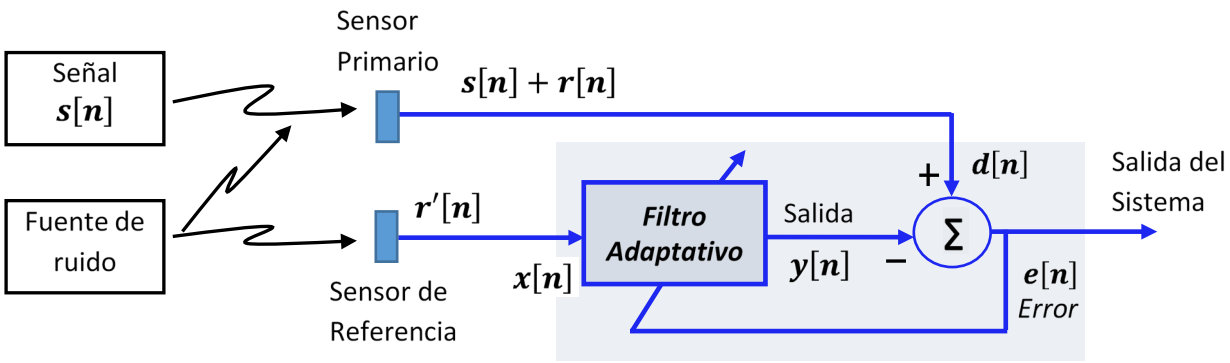


Fig. 83 – Ejemplo de Cancelación de ruido con filtro adaptativo con sensores

En la Fig. 84 se muestra un ejemplo de cancelación de ruido a la salida. La salida del sistema o canal de comunicación se le suma ruido, y este resultado ingresa al filtro adaptativo. Esta señal se filtra para producir una salida sin ruido en y. La salida en y es comparada con la señal deseada, que en este caso

es la señal de entrada x pero retrasada, se adaptan los parámetros para compensar los retardos propios del sistema y del filtro adaptativo. Después de que el filtro se ha adaptado y iguala a d, donde el error e se minimiza, y tiende a anularse.

Mediante el esquema de la Fig. 84 se puede implementar la ecualización de un canal de transmisión, por ejemplo, en telecomunicaciones. Los ecualizadores se utilizan para modificar la respuesta en frecuencia de un sistema. Esto se utiliza, por ejemplo, en una línea telefónica donde se requiere una respuesta plana entre un extremo a otro. En transmisión si se tiene un canal ecualizado, se reproduce fielmente en la salida. Los teléfonos, las líneas de transmisión y los cables de video utilizan ecualizadores que preparan las señales que serán transmitidas.

El esquema de ecualización de un canal consiste en aplicar la señal transmitida originalmente distorsionada por el canal más ruido ambiental como señal de entrada a un filtro adaptativo, mientras que la señal deseada es una versión retardada de la señal original como se observa en la Fig. 84. Esta versión retardada de la señal de entrada está disponible en general en el receptor en forma de señal de entrenamiento estándar. En el caso de no sumar ruido, con la minimización del MSE, el filtro adaptativo representa un modelo inverso (ecualizador) del canal.

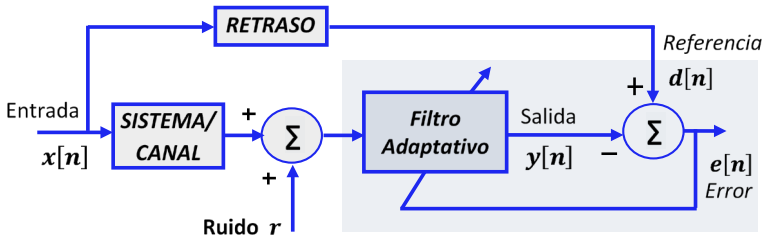


Fig. 84 - Cancelación de ruido a la salida. Ecualización de un canal de transmisión

En la Fig. 85 se muestra un resumen de aplicaciones básicas de filtrado adaptativo

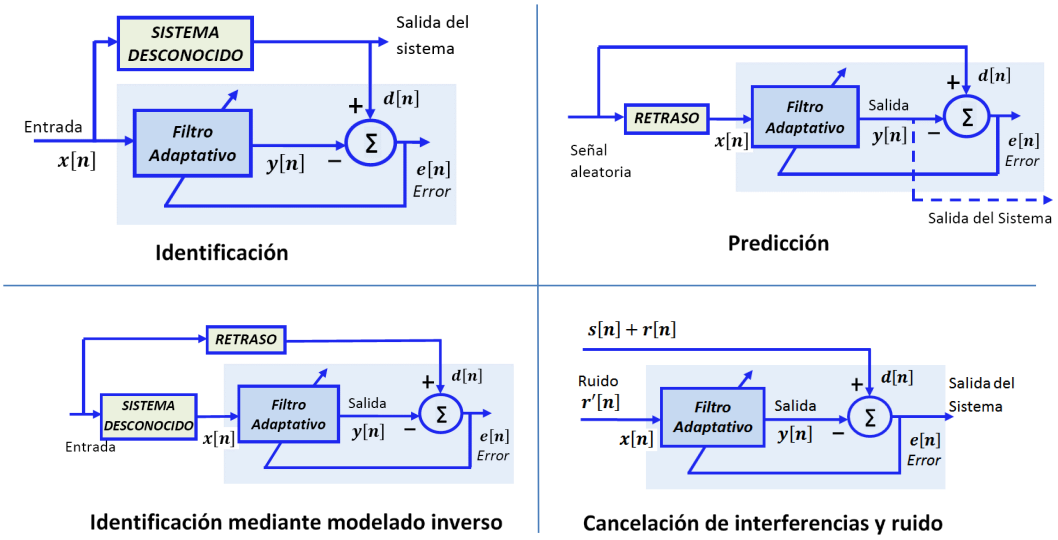


Fig. 85 – Resumen de aplicaciones básicas de filtrado adaptativo

En la Tabla I se enumeran aplicaciones específicas correspondientes a las cuatro clases básicas de aplicaciones de filtrado adaptativo. Las aplicaciones de sistemas de control de las áreas de sismología, electrocardiografía, comunicaciones y radar. Sus propósitos se describen brevemente en la última columna de la tabla.

Tabla I – Ejemplos de aplicaciones de filtros adaptativos separadas en las 4 clases básicas

Clase de Filtro Adaptativo	Aplicación	Propósito
I. Identificación	Identificación de Sistemas	Dado un sistema dinámico desconocido, el propósito del sistema de identificación es diseñar un filtro adaptativo que proporcione una aproximación al sistema.
	Modelado de capas de la superficie terrestre	En la sismología de exploración, se desarrolla un modelo en capas de la tierra para desentrañar las complejidades de la superficie terrestre.
II: Modelado inverso	Ecualización	Dado un canal de respuesta impulsiva desconocida, el propósito de un ecualizador adaptativo es operar en la salida del canal de manera que la conexión en cascada del canal y el ecualizador proporcione una aproximación a un medio de transmisión ideal.
III: predicción	Codificación predictiva	La predicción adaptativa se usa para desarrollar un modelo de una señal de interés (por ejemplo, una señal de voz); en lugar de codificar la señal directamente, en la codificación predictiva, el error de predicción se codifica para su transmisión o almacenamiento.
	Análisis espectral	En esta aplicación, el modelado predictivo se utiliza para estimar el espectro de potencia de una señal de interés
IV: Cancelación de interferencias	Cancelación de Ruido	El propósito de un cancelador de ruido adaptativo es restar ruido de una señal recibida de una manera controlada de forma adaptativa para mejorar la relación señal-ruido. La cancelación de eco, experimentada en circuitos telefónicos, es una forma especial de cancelación de ruido. En electrocardiografía (ECG) también se utiliza cancelación de ruido.
	Formador de haz	Un formador de haz es un filtro espacial que consta de una serie de elementos de antena con pesos ajustables (coeficientes). El doble propósito de un formador de haz adaptativo es controlar de manera adaptativa los pesos para cancelar las señales interferentes que inciden en la matriz desde direcciones desconocidas y, al mismo tiempo, proporcionar protección a una señal objetivo de interés.

Formación de haces Adaptativo (Adaptive Beamforming)

Los métodos y estructuras de filtrado adaptativo discutidos hasta ahora son todos de tipo temporal, ya que la operación de filtrado se realiza en el dominio del tiempo. Naturalmente, el filtrado adaptativo también puede ser de tipo espacial, en el que se coloca una serie de sensores independientes en diferentes puntos del espacio para "escuchar" una señal que se origina en una fuente distante. En efecto, los sensores proporcionan un medio para muestrear la señal recibida en el espacio. El conjunto de salidas de los sensores recopiladas en un instante particular de tiempo constituye un valor instantáneo de la fuente de datos (snapshot). Cuando los sensores se encuentran uniformemente en línea recta, la instantánea de datos en un filtro espacial juega un papel análogo al de un conjunto de entradas de consecutivas que existen en un filtro FIR en un instante de tiempo particular.

Las aplicaciones relevantes que implican el uso de procesamiento de matriz de señales son las siguientes:

- Radar, en el que los sensores consisten en elementos de antena (por ejemplo, dipolos, bocinas o guías de ondas) que responden a las ondas electromagnéticas incidentes. El requisito aquí es detectar la fuente responsable de la radiación de las ondas electromagnéticas, estimar el ángulo de llegada de las ondas y extraer información sobre la fuente.
- Sonar, en el que los sensores consisten en hidrófonos (el hidrófono consiste en un transductor que convierte sonido a electricidad, se utiliza sumergido en agua u otro líquido) diseñados para responder a ondas acústicas incidentes y los requisitos de procesamiento de señales son de naturaleza similar a los del radar.
- Mejora del habla, en la que los sensores constan de micrófonos y el requisito es, por ejemplo, escuchar la voz del hablante deseado en presencia de ruido de fondo. Se trabaja con una Matriz de señales desfasadas

En estas aplicaciones, hablamos de formación de haces, cuyo propósito es diferenciar las propiedades espaciales de la señal y el ruido. El sistema utilizado para realizar la formación de haces se denomina formador de haces. El término "formador de haz" se deriva del hecho de que las primeras antenas se diseñaron para formar haces a partir de recibir una señal de fuente que irradia desde una dirección específica y para atenuar las señales que se originan en otras direcciones que no tienen interés. Tener en cuenta que la formación de haces se aplica tanto a la radiación (es decir, la transmisión) como a la recepción de energía.

En la Fig. 86 se detalla el diagrama de bloques de un formador de haz adaptativo que usa una matriz lineal de sensores idénticos. Las salidas del sensor, que se supone que tienen banda base, se ponderan individualmente y luego se suman para producir la salida general del formador de haz.

El término "banda base" indica la banda original de frecuencias que opera la fuente. El formador de haz debe cumplir dos requisitos:

• ***Capacidad de dirección*** (***Steering capability)***, mediante la cual la señal de destino (fuente) se encuentra protegida.

• ***Cancelación de interferencias***, de modo que se maximice la relación de señal / ruido de la salida.

Un método para satisfacer estos requisitos es minimizar la varianza (es decir, la potencia promedio) de la salida del formador de haz, sujeto a la restricción de que, durante el proceso de adaptación, el vector de pesos $w[n]$ de tamaño M-por-1 satisfaga la condición:

$$w^H[n].s[\theta] = 1 \;\; ; \text{ para todo } n \text{ y } \;\; \theta = \theta_t, \qquad (18)$$

donde $s[\theta]$ es un vector de dirección de tamaño M-por-1. El superíndice H denota transposición hermitiana (es decir, transposición combinada con conjugación compleja). Se supone que los datos de la banda base tienen un valor complejo, de ahí la necesidad de una conjugación compleja. El valor del ángulo eléctrico $\theta = \theta_t$ está determinado por la dirección del objetivo. El ángulo θ se mide con respecto al sensor 0 tratado como punto de referencia. La relación del vector de dirección respecto de θ se define mediante:

$$s[\theta] = \left[1, e^{-j.\theta}, \ldots\ldots, e^{-j.(M-1).\theta}\right]^T \qquad (19)$$

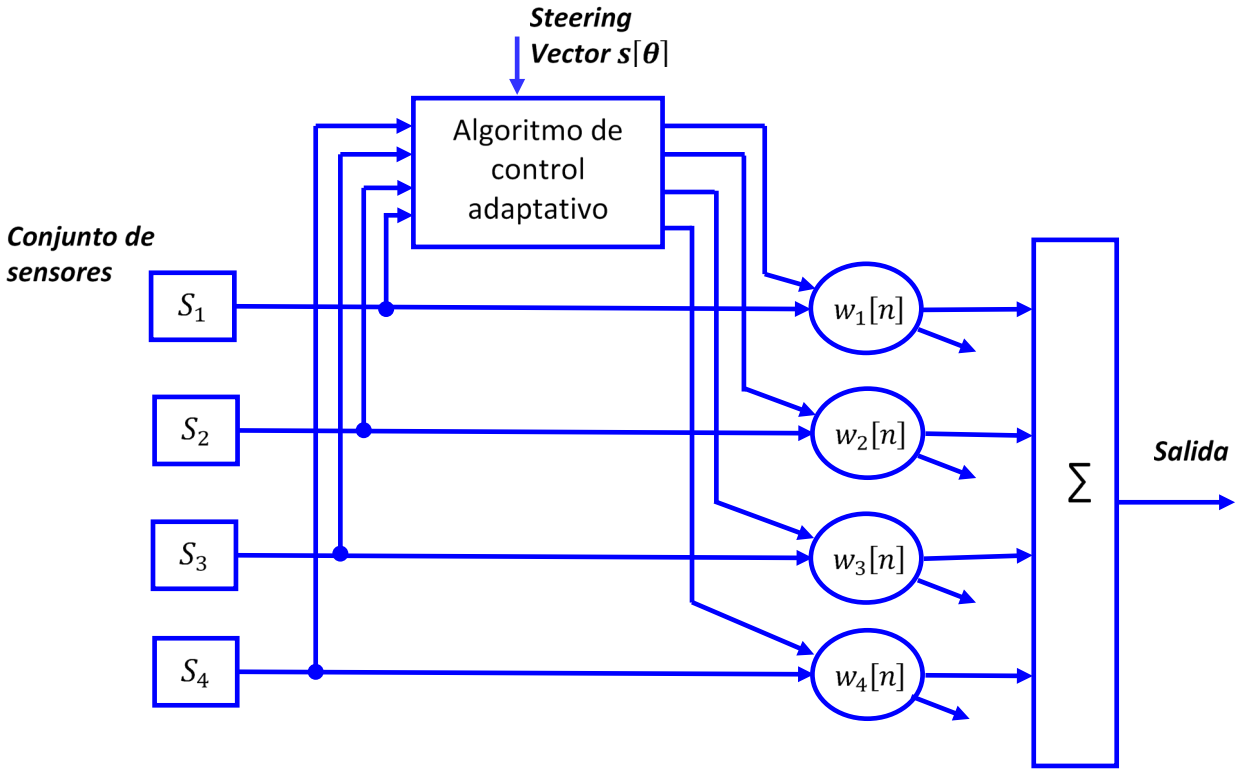


Fig. 86 – Formador de haz adaptable para una matriz de cinco sensores. Las salidas del sensor (en banda base) es de valor complejo; por tanto, las ponderaciones o pesos se valoran de forma compleja.

Sea Φ el ángulo de incidencia actual de una onda plana, medido con respecto a la normal de la matriz lineal. Entonces, de la Fig.10, vemos fácilmente que:

$$\theta = \frac{2.\pi.d}{\lambda}.\sin(\Phi) \quad ; \quad -\frac{\pi}{2} \leq \Phi \leq \frac{\pi}{2} \qquad (20)$$

Donde d es el espacio entre sensores adyacentes de la matriz y λ es la longitud de onda correspondiente a la onda incidente. Con Φ restringido dentro del rango $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$ y los valores permisibles de θ dentro del rango $[-\pi, \pi]$, encontramos, de la Ecuación (20) , que d debe ser menor que $\lambda/2$, de modo que exista una correspondencia biunívoca entre los valores de θ y Φ sin ambigüedad. Por tanto, el requisito $d < \lambda/2$ puede considerarse como el ***análogo espacial del teorema de muestreo***. La imposición de la restricción de protección de señal en la Ecuación (18)

asegura que, para la dirección del ángulo $\theta = \theta_t$ la respuesta de la matriz se mantiene constante (igual a la unidad), sin importar qué valores se asignen a los elementos del vector de peso w. Un algoritmo que minimiza la variación de la salida del formador de haz, sujeto a esta restricción, se denomina algoritmo de formación de haz de respuesta sin distorsión de variación mínima (MVDR: minimum-variance distortionless response). Tener en cuenta que la imposición de la restricción de protección de señal reduce el número disponible de grados de libertad a $M - 2$, donde M es el número de sensores en la matriz. En consecuencia, el número de nulos independientes producidos por el algoritmo MVDR (es decir, el número de interferencias independientes que pueden cancelarse) es $M - 2$.

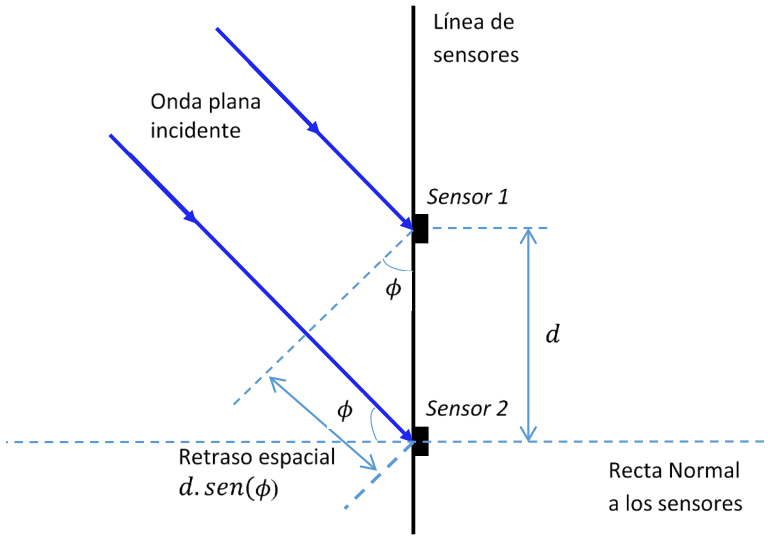


Fig. 87 – Retraso espacial cuando una onda plana incide en una matriz lineal.

Aplicaciones de audio

En la Fig. 88 se muestra un ejemplo de cancelación de ruido en una sala de conferencias o en una fábrica con máquinas que generan ruidos molestos. Mediante un micrófono principal, se recoge la voz con ruido del orador principal y además mediante micrófonos secundarios se recoge el ruido de ambiente sólo (se colocan estos micrófonos alejados del orador principal). En estos casos más generales los filtros tienen que ser grandes, ya que el ruido se produce en varias frecuencias y no es una señal de banda estrecha. Este método resulta eficaz cuando el ruido es regular y contiene muchos armónicos como por ejemplo una máquina que gira. Si se tiene ruido de banda ancha sólo funciona bien en las frecuencias bajas.

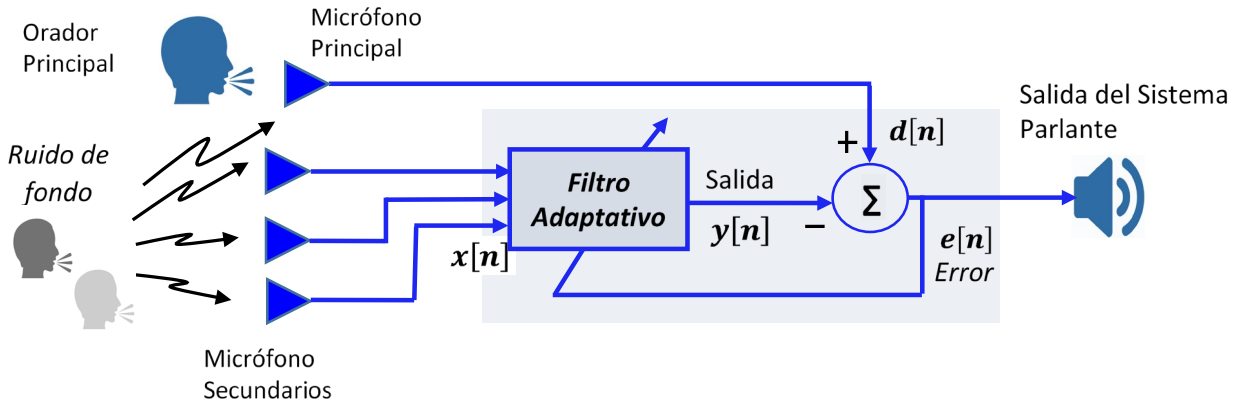


Fig. 88 – Ejemplo de cancelación de ruido en voz mediante filtro adaptativo

Ejercicios de filtros adaptativos en Matlab®

Ejercicio 1.1

Ejemplo simple LMS con filtro de 1 solo coeficiente

Se desea tener un sistema Simple adaptativo, donde la salida valga 10, independientemente de los valores que tome la entrada.

Suponer que la entrada primero toma estos valores: $x_1[n] = 2.\sum_{k=0}^{14} \delta[n-k] = \{2, 2, 2, \ldots..\}$, luego toma estos valores: $x_2[n] = \{3, 3, 3, \ldots..\}$. Y por último $x_3[n] = \{a, a, a, \ldots..\}$, siendo $1 \leq a \leq 3$

Se pide:

a) Resolver el sistema con planilla Excel. Por ejemplo, elegir $a = 2{,}8$ y $a = 20$ (que no cumpla restricción). Asignar los largos adecuados de $x_2[n]$ y $x_3[n]$ de manera de poder sacar conclusiones
b) Para un factor de convergencia de $\mu = 0{,}1$, indicar cuantas iteraciones se necesitan para que el algoritmo converja para el caso de $x_1[n]$ y para el caso de $x_2[n]$
c) Probar distintos valores de μ ¿Qué pasa con la velocidad de convergencia y el error?
d) ¿Qué pasa para $\mu = 0{,}2$?

Resolución:

a) b) c)

Valor deseado de la salida: $d[n] = \{10, 10, 10, \ldots..\}$

Para $x_1[n]$, calculamos el coeficiente $w[n]$ que aproxime la salida $y[n]$ al valor esperado: 10

$y[n] = w[n].x[n]$

$w[n] = \frac{10}{2} = 5$, este es el valore que debería converger

Estimamos $w[n]$ mediante LMS. En la Fig. 89 se muestra una resolución simple, donde se detallan las fórmulas utilizadas, la convergencia de w y el error.

Pendiente para el lector completar análisis y conclusiones

d) Si la señal de entrada varía entre 1 y 3, su valor máximo es 3. Su potencia máxima resulta: $P_{x\,máx} = 3^2 = 9$

Más adelante veremos que debe cumplir: $0 < \mu < \frac{1}{N.P_x}$, siendo N la cantidad de coeficientes que contiene el filtro. En este caso: $N = 1$. Nos queda:

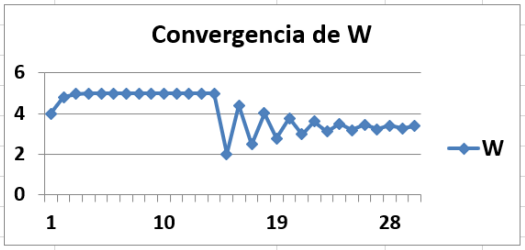
$$0 < \mu < \frac{1}{9} = 0{,}111$$

Para $\mu = 0{,}2$ No cumple. Verificar con Planilla Excel.

$y = w.x$ $\quad e = d - y$ $\quad w[n+1] = w[n] + 2.\mu.e.x$

n	w(n)	Entrada: x	Salida: y	Error: e	w(n+1)
0	0	2	0	10	4
1	4	2	8	2	4,8
2	4,8	2	9,6	0,4	4,96
3	4,96	2	9,92	0,08	4,992
4	4,992	2	9,984	0,016	4,9984
5	4,9984	2	9,9968	0,0032	4,99968
6	4,9997	2	9,99936	0,0006	4,99994
7	4,9999	2	9,99987	0,0001	4,99999
8	5	2	9,99997	3E-05	5
9	5	2	9,99999	5E-06	5
10	5	2	10	1E-06	5
11	5	2	10	2E-07	5
12	5	2	10	4E-08	5
14	5	2	10	8E-09	5
15	5	3	15	-5	2
16	2	3	6	4	4,4
17	4,4	3	13,2	-3,2	2,48
18	2,48	3	7,44	2,56	4,016
19	4,016	3	12,048	-2,048	2,7872
20	2,7872	3	8,3616	1,6384	3,77024
21	3,7702	3	11,3107	-1,3107	2,98381
22	2,9838	3	8,95142	1,0486	3,61295

Valor esperado: d	10
w0	0
mu	0,1



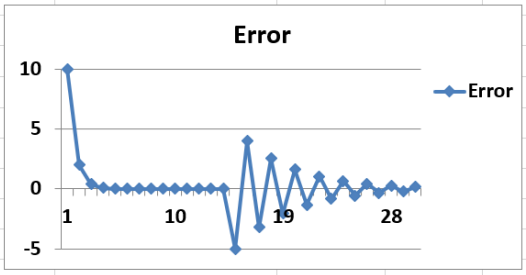


Fig. 89 – Ejemplo simple de LMS con planilla Excel para un filtro de 1 coeficiente

Ejercicio 1.2

Ejemplo Simple LMS para «Identificación»» de Sistemas

Mediante un sistema adaptativo, se desea tener una señal de salida fija donde su amplitud valga 10.

Entrada en formato Matlab®: x=[2*ones(1,30) 3*ones(1,30) 1.6*ones(1,30)] ;

a) Utilizar sistema con filtro de 1 coeficiente, $\mu = 0{,}2$ y la librería LMS.m

b) Modificar secuencia de entrada

Comparar respuesta para: $\mu = 0{,}1 * 2$; $\mu = 0{,}01 * 2$; $\mu = 0{,}25 * 2$

Escribir conclusiones: …que pasa con el error y la velocidad de convergencia

Resolución

```
x=[2*ones(1,30)  3*ones(1,30)  1.6*ones(1,30)] ;
largo = length(x) ;
d=10*ones(1,largo) ;
mu=0.1  *2; N= 1 ; % Coefic del filtro
W0=0 ;  % Coeficiente Inicial
S  = struct('step',mu,'filterOrderNo',(N-1),'initialCoefficients',W0);
  %[y,e,W]  =   LMS(d,transpose(x),S);
[y,e,W]  =   LMS(d,x,S);  % Solicitar librería al autor de este libro

%   Graficamos error MSE
figure,
plot(1:largo, e,'-k', 'linewidth',3); grid on
title('Curva de aprendizaje del error'); axis tight;
xlabel('Número de iteraciones, k'); ylabel('Error');
%   Graficamos salida y
figure,
plot(1:largo, y,'-k', 'linewidth',3); grid on
title('Salida del Sistema:  y');
xlabel('Número de iteraciones, k'); ylabel('Amplitud');
%   Graficamos W
figure;
subplot (211), plot(real(W(1,:)), 'linewidth',3); axis tight; grid on
title('Evolución del primer coeficiente (parte real)');
xlabel('Número de iteraciones, k'); ylabel('Coeficientes');
subplot (212), plot(imag(W(1,:)), 'linewidth',3, 'color','r');axis tight;
title('Evolución del primer coeficiente (parte imaginaria)');
xlabel('Número de iteraciones, k'); ylabel(' Coeficientes ');
```

En las Fig. 90 y Fig. 91 se muestran los resultados Matlab® del punto a).

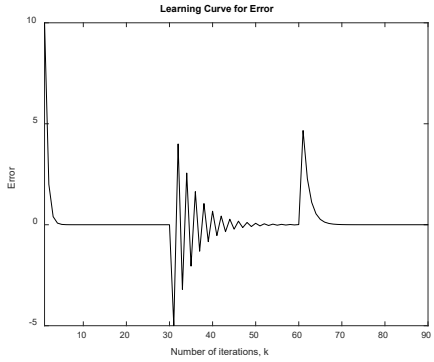
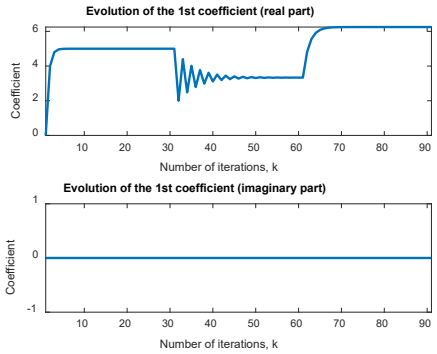
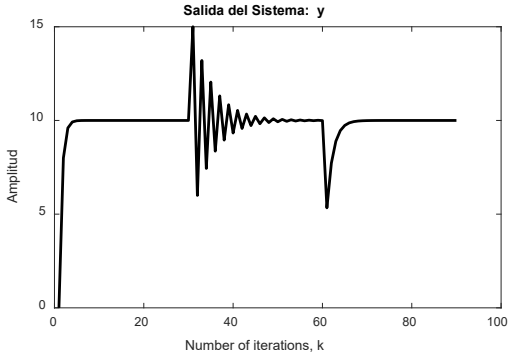


Gráfico de señal de Error



Evolución del coeficiente del filtro

Fig. 90 – Gráfico de error y evolución del coeficiente del filtro adaptativo LMS



Salida del Sistema

Fig. 91 – Salida del sistema LMS

Ejercicio 1.3

Algoritmo LMS

Se pide analizar el siguiente código

```
% Debemos generar las señales ss  (señal original), n (ruido 1) y x (ruido 2)
%% Filtro Adaptativo LMS
  % Inicializamos
  d=ss + n; % Señal d= ss (audio) + ruido
  mu=0.01; % Parámetro mu
  n_coeficientes = 19 ;
  w=zeros(1, n_coeficientes); % Inicializamos los coeficientes del filtro Adaptativo
  y=zeros(1,length(t)); % Inicializamos en cero el vector de salida
  error=y; % Inicializamos vector error (ceros)
  % Filtro Adaptativo con Algoritmo LMS
  inicio = n_coeficientes+1 ;
```

```
 for m=  inicio : (length(t)-1)
   sum=0;
   for i=1: n_coeficientes
    sum =sum +  w(i)* x(m - i);
   end
   y(m)=sum;
   error(m)=d(m)-y(m);
   for i=1: n_coeficientes
    w(i) = 2*mu*error(m)*x(m - i)  + w(i) ;
   end
   % Guardamos histórico de los coeficientes, "solo" para analizar convergencia
   W_matriz_hist(m-n_coeficientes, :)=w ; % Opcional, funciona más lento
 end
% Calculamos Transformada de Fourier.
% Graficamos señales Temporales, transformadas, correlaciones, etc.
```

Ejercicio 1.4

I - Cancelación de Ruido de Audio

Se pide filtrar audio con el sistema adaptativo de la Fig. 92

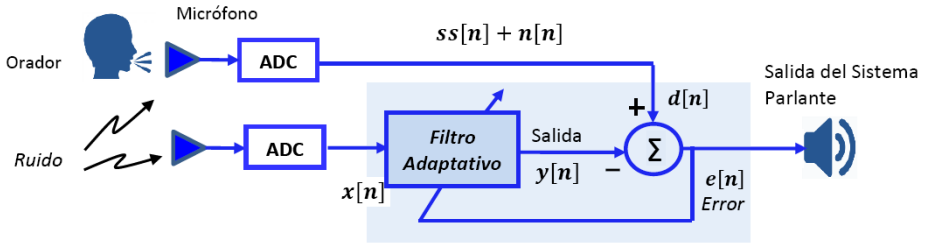


Fig. 92 – Esquema del sistema adaptativo para cancelación de ruido de voz

Se pide implementar dicho sistema en Matlab®

Ver resolución en el capítulo IV del presente libro.

Se muestran los resultados Matlab® en la Fig. 93.

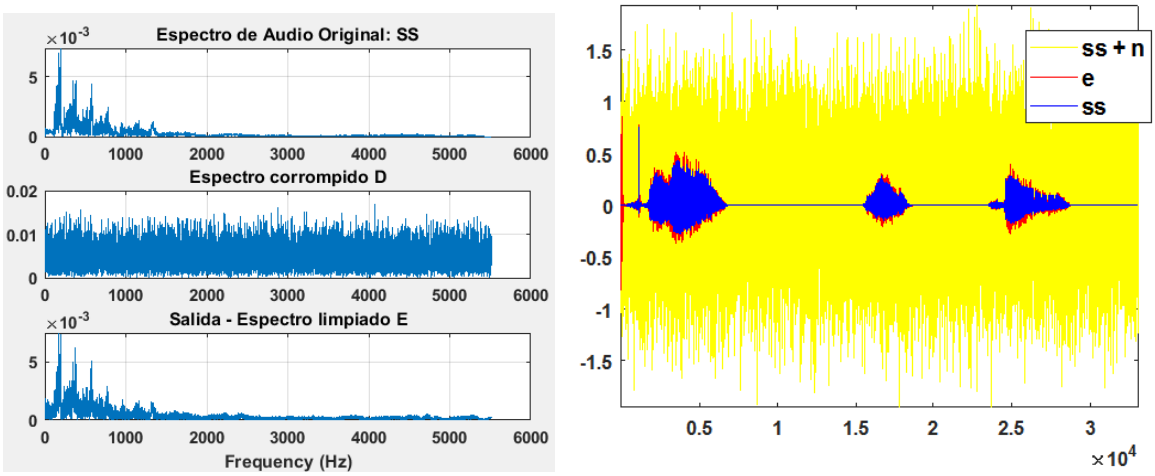


Fig. 93 – Espectros de las señales y señales temporales

Descarga de los códigos de los ejercicios

II - Procesos estacionarios y modelos

Introducción a Procesos Estocásticos

Variable aleatoria

Definimos el siguiente espacio muestral y variable aleatoria

- Espacio muestral:
  $S = \{s_{1,}\ s_2\ , \ldots . , s_n\}$ donde s_i son los resultados experimentales
- Variable Aleatoria. Es una función de los datos experimentales
  $X = X(s_i)$

Notación utilizada:

$\{X \leq x\}$ es el subconjunto de resultados de S que consiste en todos los eventos s_j tales que $X = X(s_j)$ sea menor que x

Función de distribución acumulativa de probabilidad:

$$F_X(x) = P\{X \leq x\}$$

Ejemplo de tiro de un dado

Suponemos un dado que se arroja 60 veces, se obtienen los siguientes conjuntos

$S = \{C_{uno,}, C_{dos}, C_{tres,}, C_{cuatro}, C_{cinco}, C_{seis}\}$

Procesos estocásticos

Un proceso estocástico es un conjunto (ensamble) de funciones o secuencias reales o complejas definidas en un espacio de probabilidad. Para tiempos continuos utilizamos $t_1, t_2, \ldots .$, para tiempos discretos $n_1, n_2, \ldots .$ Para el caso de tiempo continuo, los valores de $x(t_1), x(t_2), \ldots$ son variables aleatorias. La señal $x_k(t)$ corresponde a una de las muchas realizaciones del proceso que hubieran podido tener lugar. El conjunto de todos estos posibles registros $x_k(t)$, para $k = 1,2,..$ caracterizan el proceso estocástico. El valor de la variable x en un determinado instante t_0 depende de cuál de las realizaciones se considere. En la Fig. 94 se muestra un ejemplo de un proceso estocástico.

Los procesos estocásticos son aleatorios y no tienen un comportamiento determinista. Se utiliza el término estocástico en procesos, modelos y algoritmos que tienen una secuencia variable de eventos

que se pueden analizar con probabilidades en el transcurso de la variable tiempo. El término proceso estocástico, también conocido como proceso aleatorio, describe la evolución temporal de un fenómeno estadístico de acuerdo con leyes probabilísticas. Esta evolución temporal del fenómeno quiere decir que el proceso estocástico depende del tiempo, definido en algún intervalo de observación. Al tratarse de fenómenos estadísticos hay que realizar experimentos para conocer correctamente la forma que evolucionan en el tiempo.

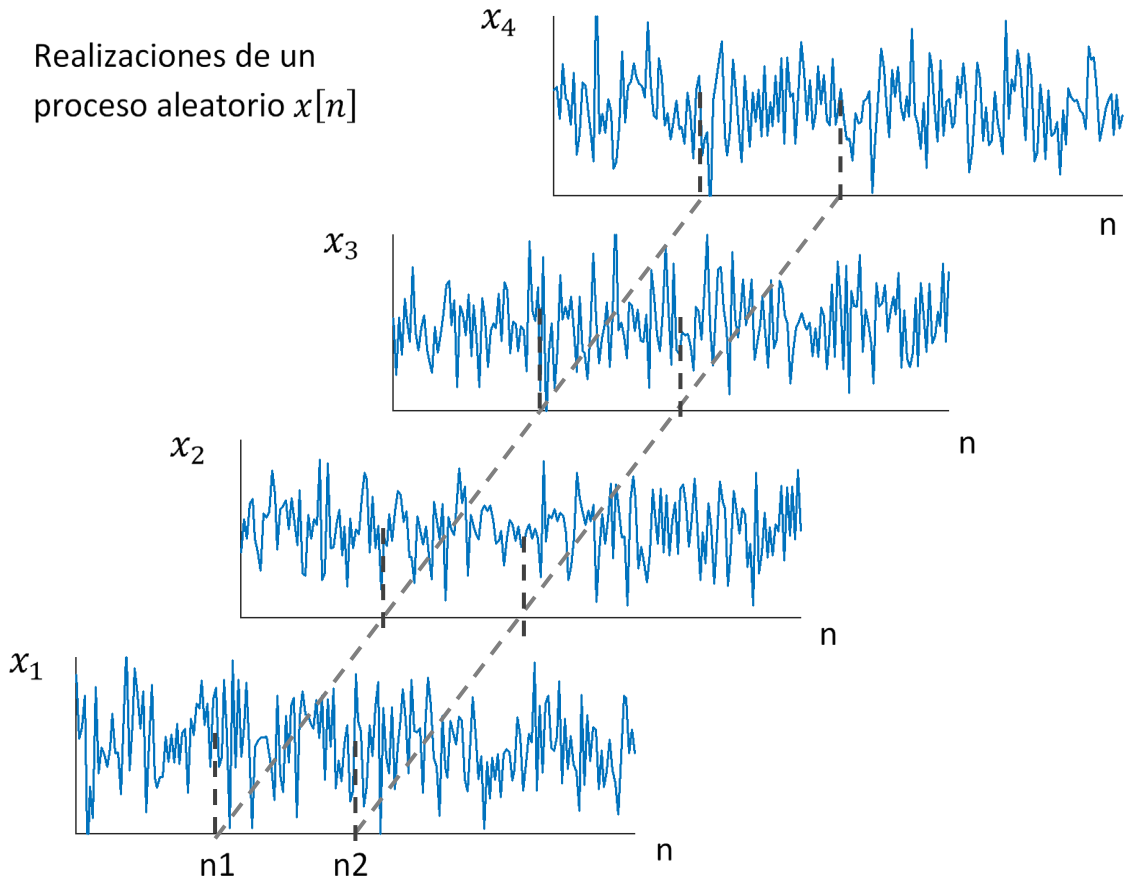


Fig. 94 - Proceso Estocástico para señal de tiempo discreto $\boldsymbol{n}$

En sistemas Adaptativos, estudiaremos los procesos estocásticos correspondientes a señales de tiempo discreto, con muestreo a intervalos constantes y que cumplen con el Teorema de Muestreo. Este teorema dice que la frecuencia de muestreo, conocida como f_s, debe ser mayor al doble de la frecuencia máxima (f_{max}) de la señal analizada

$$f_s > 2.f_{max}$$

Las señales determinísticas son aquellas en las que se puede calcular su valor de forma inequívoca.

En cambio, las señales estocásticas tienen las siguientes características:

- Son aquellas que no pueden definirse de manera determinística.
- Es una variable aleatoria que varía en el tiempo
  $X_1(t) = X(s,t)$
- Se conocen también como señales aleatorias o no determinísticas.

- No se puede estar seguro de que converjan las Transformadas de Laplace y la de Fourier.
- Procesar las señales de forma individual es poco interesante, solo resuelve un caso particular.

Ejemplos de señales estocásticas:

- Ruido electrónico en amplificadores
- Interferencia electromagnética en antenas
- Ruido térmico en resistores
- Voz humana

Definimos el proceso que las genera como un Proceso Estocástico (o proceso aleatorio)

– Este produce un ensamble de señales aleatorias:

$$X_1(t), X_2(t), \dots., X_i(t)$$

– Cada una de ellas es una realización

Valor esperado

Para caracterizar el proceso estocástico utilizaremos el valor esperado (promedio del ensamble) y promedios temporales. Estudiaremos el tratamiento de estas señales en sistemas LTI y posteriormente en sistemas que no cumplen invarianza temporal. En la Fig. 95 se muestra un ejemplo de realizaciones (funciones muestra) de un proceso estocástico para señal de tiempo continuo t.

Tomando la función promedio, el valor esperado es el promedio del ensamble para cada instante de tiempo fijo. Entonces el valor esperado del ensamble resulta:

$$E\{x(t_1)\} = \lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} x_i(t_1)$$

Siendo t_1 un instante de tiempo determinado

Teniendo en cuenta que $N \to \infty$, se puede expresar con la siguiente notación, para funciones continuas:

$$E\{X(t)\} = \int_{-\infty}^{\infty} x.f_x(x,t).dx$$

El valor esperado ppuede ser variable en el tiempo, es decir:

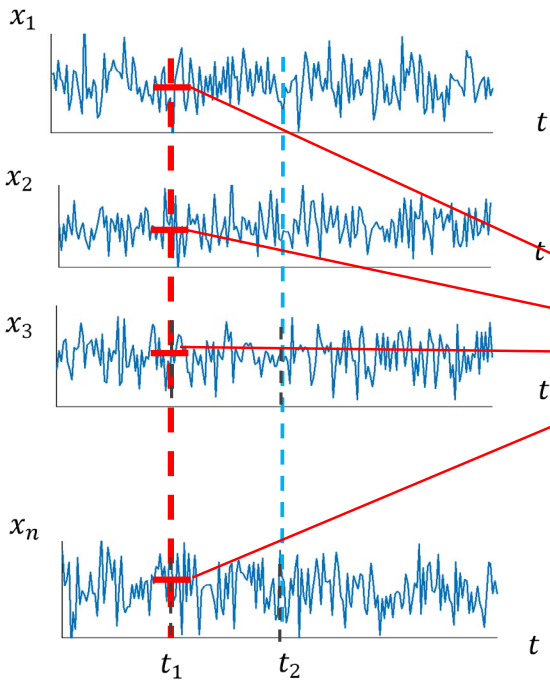
$$E\{x(t_1)\} \neq E\{x(t_2)\}$$

El valor esperado cumple la siguiente propiedad:

$$E\{a.x(t) + b.y(t)\} = a.E\{x(t)\} + b.E\{y(t)\}$$

Para el caso determinístico se cumple:

$$E\{d(t)\} = d(t)$$



Realizaciones de un proceso aleatorio $x(t)$

Valor esperado (media del ensamble):

Es el valor medio obtenido a lo largo del ensamble considerando ***tiempo constante*** $(t = t_1)$

Momento de 1er orden

$$E\{x(t_1)\} = \mu_x = \lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} x_i(t_1)$$

$$E\{x(t_1)\} \neq E\{x(t_2)\}$$

En la práctica no se puede calcular ya que se desconoce la totalidad de las muestras del ensamble.

Selo aproxima a partir de un conjunto finito de muestras

Fig. 95 - Proceso Estocástico y Valor esperado. Ejemplo de realizaciones (funciones muestra) de un proceso estocástico para señal de tiempo continuo $\boldsymbol{t}$

Tomamos como ejemplo el valor esperado de los procesos aleatorios A y B graficados en la Fig. 96. En las realizaciones del proceso A se observa un comportamiento senoidal con ruido, en el proceso B no se observa este comportamiento. Sin embargo, ambos procesos tienen valores esperados similares: $E\{x(t)\} \cong E\{y(t)\}$

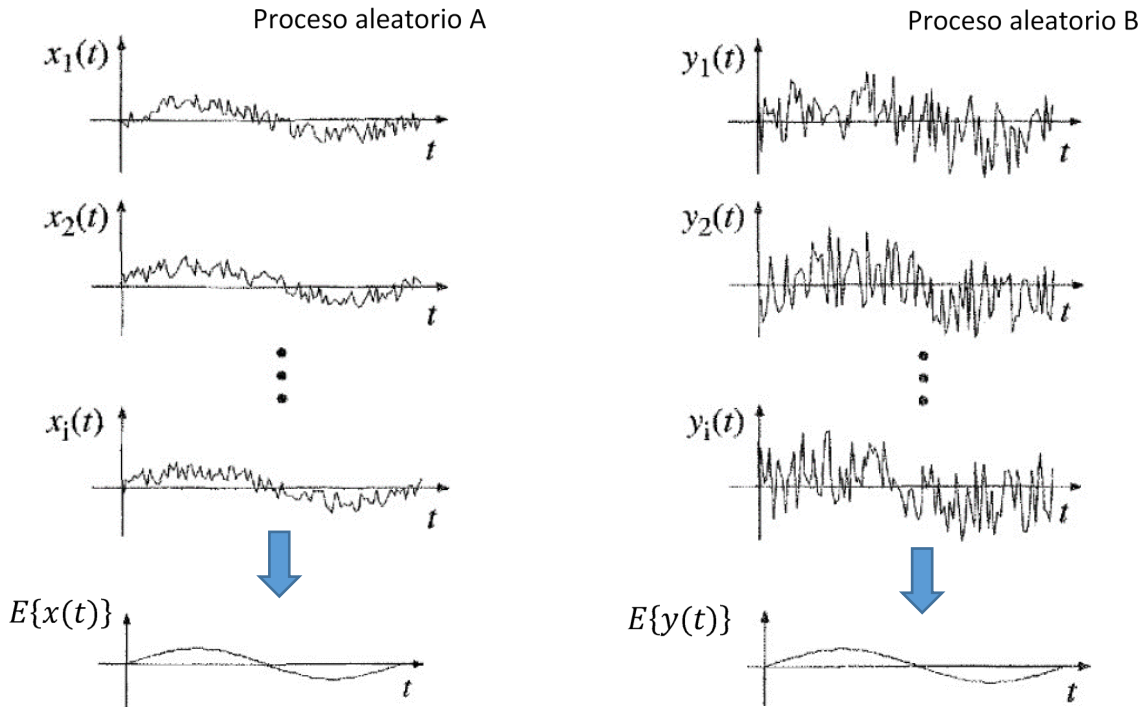


Fig. 96 – Valor esperado para 2 procesos aleatorios distintos

Valores esperados de primer orden

La forma general tiene la siguiente forma:

$$E\{f(x(t))\} = \lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} f(x_i(t))$$

Se denomina primer orden porque depende de un solo valor de tiempo.

Se pueden tener distintas funciones $f(x(t))$

● Promedio

Si la función $f(x(t)) = x(t)$ se obtiene el promedio:

$$E\{f(x(t))\} = E\{x(t)\} = \lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} x_i(t) = \mu_x(t)$$

● Promedio cuadrático

Si la función $f(x(t)) = x^2(t)$ se obtiene el siguiente valor esperado:

$$E\{f(x(t))\} = E\{x^2(t)\} = \lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} {x_i}^2(t)$$

Este valor corresponde a la Energía de la señal

Varianza temporal: Cuadrado de la desviación del promedio lineal. Mide las ***dispersiones de los datos en torno al promedio***

Si la función $f(x(t)) = (x(t) - \mu_x(t))^2$ se obtiene el siguiente valor esperado:

$$\boxed{{\sigma_x}^2(t) = E\{f(x(t))\} = E\{(x(t) - \mu_x(t))^2\}}$$

${\sigma_x}^2(t) = E\{(x(t) - \mu_x(t))^2\} = E\{x^2(t) - 2.x(t).\mu_x(t) + {\mu_x}^2(t)\} = E\{x^2(t)\} - 2.\mu_x(t).E\{x(t)\} + {\mu_x}^2(t)$

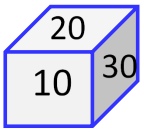
$= E\{x^2(t)\} - 2.\mu_x(t).\mu_x(t) + {\mu_x}^2(t)$

$$\boxed{{\sigma_x}^2(t) = E\{x^2(t)\} - {\mu_x}^2(t)}$$

Ejemplo:

Se tiene un dado que tiene los siguientes valores en sus 6 caras:

$$x_i = \{10\,,20\,,30\,,40\,,50\,,60\}$$



Se pide:

a) Simular con Matlab® el tiro del dado 1000 veces, cada 100 milisegundos, a este evento lo llamamos realización. Repetir este evento 100000 veces.
b) Calcular el valor medio del proceso para distintos valores de *"iteraciones n"* y su desviación estándar
c) Calcular el valor medio de cada realización y su desviación estándar $\sigma = \sqrt{\sigma^2}$
d) Analizar los resultados obtenidos

Resolución:

a) b) c)
Código Matlab®:

```
% Punto a) Generamos una matriz de 100000 filas y 1000 columnas
% Función randi genera números aleatorios Enteros, en este caso de 1 a 6
   matriz = randi([1 6],100000,1000);
% Modificamos para que tengas estos valores: 10,20,30,40,50,60
   matriz = matriz  *10;
% Punto b)
   valor_medio_columnas = mean(matriz,1) ;
   std_columnas = std(valor_medio_columnas) ;
% Punto c)
   valor_medio_filas = mean(matriz,2) ;
   std_filas = std(valor_medio_filas) ;
% Punto d)
   A= mean(valor_medio_filas) ;
```

```
display(['Valor medio A:  ' num2str(A)])
display(['Desviación estándar filas:  ' num2str(std_filas)])
B= mean(valor_medio_columnas) ;
display(['Valor medio B:  ' num2str(B)])
display(['Desviación estándar columnas:  ' num2str(std_columnas)])
```

Resultados:

Valor medio A: 35.0003

Desviación estándar filas: 0.54127

Valor medio B: 35.0003

Desviación estándar columnas: 0.053444

En la Fig. 97 se muestra la matriz de datos obtenidos y los valores medios

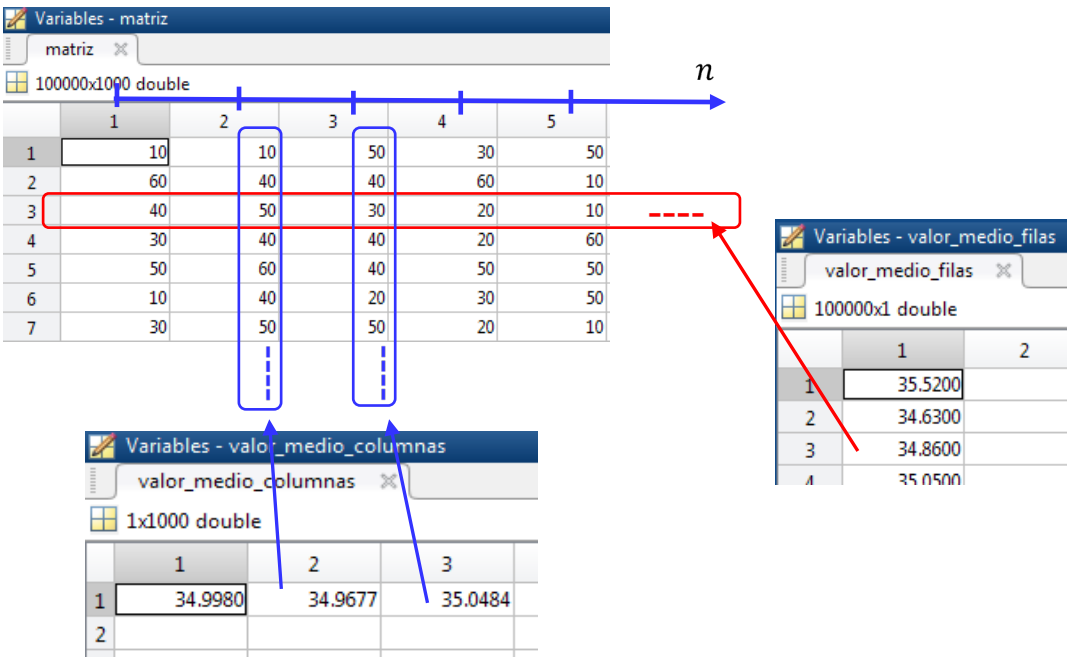


Fig. 97 –Resultados Matlab® del valor esperado

Siendo $x_i = \{10\,,20\,,30\,,40\,,50\,,60\}$, entonces la probabilidad para cada elemento resulta: $p_x(x_i) = 1/6$

En este caso se específica la probabilidad de cada muestra $p_x(x_i)$, habiendo solamente 6 casos posibles

$$E\{x(t)\} = \sum_{i=1}^{6} x_i(t).\,p_x(x_i(t))$$

Podemos desplazarnos en t, y se obtienen los mismos resultados. Entonces, en este caso la expresión anterior se puede escribir de la siguiente manera:

$$\sum_{i=1}^{6} x_i . p_x(x_i) = 35$$

Si calculamos el promedio temporal llegamos al mismo resultado que promedio del ensamble.

En este caso particular $E\{x(t_1)\} = E\{x(t_2)\}$

Valores esperados de segundo orden

En los valores esperados de segundo orden, las funciones dependen de 2 instantes de tiempo

$$E\{f(x(t_1), x(t_2))\} = \lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} f(x_i(t_1), x_i(t_2))$$

Si $f(u, v) = u.v$, es decir producto, se obtiene la función de correlación

$$\varphi_{xx}(t_1, t_2) = E\{x(t_1).x(t_2)\}$$

Al evaluar en un mismo instante, se obtiene la Energía

$$\varphi_{xx}(t_1, t_1) = E\{x^2(t_1)\}$$

Los procesos aleatorios de la Fig. 98 tienen la misma media, es decir que para este caso con los valores de media no se pueden distinguir. Sin embargo, tienen diferentes valores de autocorrelación, ver Fig. 99

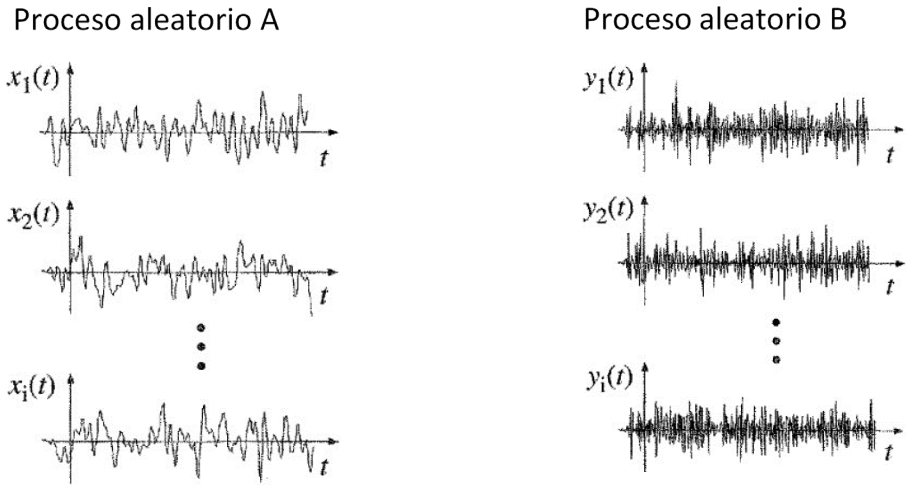


Fig. 98 – Procesos aleatorios distintos, pero con mismo valor esperado

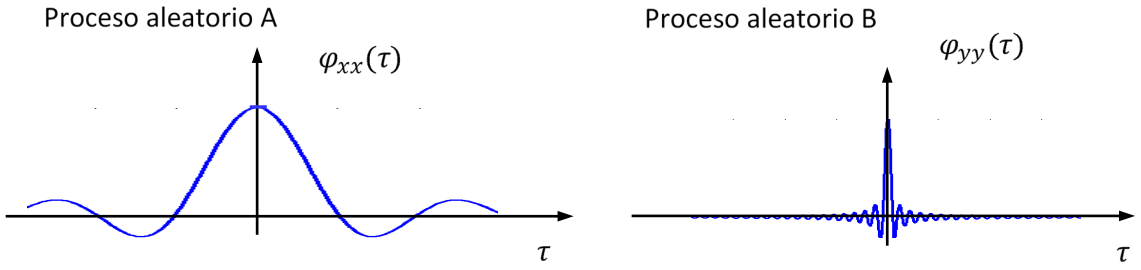


Fig. 99 – Función de autocorrelación para 2 procesos aleatorios distintos

Procesos aleatorios estacionarios en sentido estricto

Son procesos donde las propiedades estadísticas no cambian con el tiempo

Definición:

Un proceso aleatorio es estacionario si sus valores esperados de segundo orden solo dependen de las diferencias entre los tiempos observados.

$$E\{f(x(t_1), x(t_2))\} = E\{f(x(t_1 + \Delta t), x(t_2 + \Delta t))\}$$

Si analizamos las diferencias temporales (desplazamiento temporal):

$E\{f(x(t_1), x(t_2))\} \rightarrow \quad t_1 - t_2$

$E\{f(x(t_1 + \Delta t), x(t_2 + \Delta t))\} \quad \rightarrow \quad t_1 + \Delta t - (t_2 + \Delta t) \rightarrow \quad t_1 - t_2$

En este ejemplo tienen el mismo desplazamiento temporal, al ser proceso estacionario los valores esperados resultan iguales

Los valores esperados de primer orden son casos particulares de los de segundo orden

$f(x(t_1), x(t_2)) = g\big(x(t_1)\big)$; solo se analiza en t_1

$E\big\{g\big(x(t_1)\big)\big\} = E\big\{g\big(x(t_1 + \Delta t)\big)\big\}$; Valor esperado de primer orden para proceso estacionario

Es decir, que para un proceso estacionario el valor esperado de primer orden se mantiene constante si nos desplazamos en el tiempo

La media y la varianza, correspondientes a un proceso estacionario en sentido estricto se mantienen constantes con el tiempo

$\mu_x(t) = \mu_x \quad ; \quad {\sigma_x}^2(t) = {\sigma_x}^2$

La autocorrelación para un proceso estacionario en sentido estricto

$E\{x(t_1).x(t_2)\} = E\{x(t_1).x(t_1 - \tau)\} = E\{x(t_2 + \tau).x(t_2)\} = \varphi_{xx}(\tau)$

En la ecuación anterior el desplazamiento es $\tau = t_1 - t_2$ en los 4 términos

Es decir que la autocorrelación en un proceso estacionario estricto dependerá del desplazamiento τ y no del instante de tiempo que analice. Podemos tomar un mismo valor de desplazamiento τ , pero en otros instantes $\tau = t_3 - t_4$ y obtendremos mismo valor de autocorrelación.

Procesos estacionarios en sentido amplio (débil)

Se denomina estacionario en sentido amplio si se cumple la siguiente ecuación para la Autocorrelación y la media

$$E\{f(x(t_1), x(t_2))\} = E\{f(x(t_1 + \Delta t), x(t_2 + \Delta t))\}$$

Recordando en la autocorrelación se toma el producto: $f(x(t_1), x(t_2)) = x(t_1), x(t_2)$

Y en valor medio solo se toma un instante: $f(x(t_1), x(t_2)) = x(t_1)$

Entonces

$\mu_x(t) = \mu_x \quad ; \quad {\sigma_x}^2(t) = {\sigma_x}^2$

$E\{x(t_1).x(t_2)\} = E\{x(t_1).x(t_1 - \tau)\} = E\{x(t_2 + \tau).x(t_2)\} = \varphi_{xx}(\tau)$

$\tau = t_1 - t_2 \quad ;$

Procesos ergódicos (sentido estricto)

Un proceso estocástico se puede decir que resulta ergódico cuando sus propiedades estadísticas se pueden deducirse a partir de una única muestra larga y aleatoria del proceso

Definición:

Un proceso aleatorio es ergódico en la media, si los promedios temporales son equivalentes a los promedios del ensamble.

Un proceso es ergódico cuando los parámetros estadísticos en un conjunto de realizaciones (es decir promedios de conjunto) son iguales a los parámetros estadísticos en una única realización (promedios temporales).

Los procesos ergódicos siempre son estacionarios, sin embargo, lo contrario no siempre se cumple. Generalmente nunca tendremos un proceso estacionario real. A pesar de esto, la hipótesis de estacionariedad proporciona resultados bastante buenos para nuestros propósitos. Esto también se cumple con la suposición de ergodicidad. Muchas veces sólo se mide una realización de un proceso estocástico y resulta imprescindible suponer ergodicidad para poder continuar con el análisis de datos.

Primer orden:

$$\overline{f\big(x_i(t)\big)} = \lim_{T\to\infty} \frac{1}{2.T} \int_{-T}^{T} f\big(x_i(t)\big).dt$$

Segundo orden:

$$\overline{f\big(x_i(t), x_i(t-\tau)\big)} = \lim_{T\to\infty} \frac{1}{2.T} \int_{-T}^{T} f\big(x_i(t), x_i(t-\tau)\big).dt$$

Procesos ergódicos en sentido amplio

Se denomina ergódico en sentido amplio si las siguientes ecuaciones:

$$\overline{f(x_i(t))} = \lim_{T\to\infty} \frac{1}{2.T} \int_{-T}^{T} f(x_i(t)).dt$$

$$\overline{f(x_i(t), x_i(t-\tau))} = \lim_{T\to\infty} \frac{1}{2.T} \int_{-T}^{T} f(x_i(t), x_i(t-\tau)).dt$$

Se cumplen para:

$$f(x(t_1), x(t_2)) = x(t_1), x(t_2)$$
$$f(x(t_1), x(t_2)) = x(t_1)$$

La media es:

$$E\{x(t)\} = \mu_x = \overline{x(t)} = \lim_{T\to\infty} \frac{1}{2.T} \int_{-T}^{T} x(t).dt$$

Función de autocorrelación en procesos ergódicos en sentido amplio

$$\varphi_{xx}(\tau) = E\{x(t).x(t-\tau)\}$$

Su máximo se encuentra en cero, para demostrarlo planteamos $E\left\{\left(x(t) - x(t-\tau)\right)^2\right\} \geq 0$

$E\left\{\left(x(t) - x(t-\tau)\right)^2\right\} = E\{x^2(t)\} - 2.E\{x(t).x(t-\tau)\} + E\{x^2(t-\tau)\} = \varphi_{xx}(0) - 2.\varphi_{xx}(\tau) + \varphi_{xx}(0) \geq 0$

$\Longrightarrow \ \varphi_{xx}(0) \geq \varphi_{xx}(\tau) \ \Longrightarrow \ \varphi_{xx}(0)$ Tenemos el valor máximo de la autocorrelación

Para obtener el Límite inferior, planteamos: $E\left\{\left(x(t) + x(t-\tau)\right)^2\right\} \geq 0$

$$E\left\{\left(x(t) + x(t-\tau)\right)^2\right\} = E\{x^2(t)\} + 2.E\{x(t).x(t-\tau)\} + E\{x^2(t-\tau)\}$$
$$= \varphi_{xx}(0) + 2.\varphi_{xx}(\tau) + \varphi_{xx}(0) \geq 0$$
$$\Longrightarrow \ \varphi_{xx}(\tau) \geq -\varphi_{xx}(0)$$

En la Fig. 100 se muestran los valores máximos y mínimos que puede tomar la función de autocorrelación

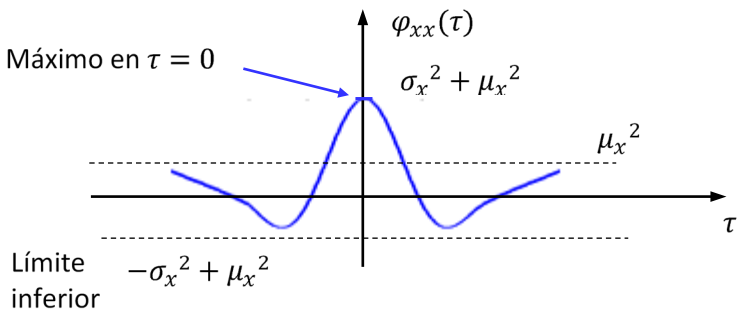


Fig. 100 – Valores máximos y mínimos de la autocorrelación

La función de autocorrelación es par: $\varphi_{xx}(\tau) = \varphi_{xx}(-\tau)$

$E\{x(t).x(t+\tau)\} = E\{x(t'-\tau).x(t')\} = E\{x(t').x(t'-\tau)\}$

$\Longrightarrow \quad \varphi_{xx}(\tau) = \varphi_{xx}(-\tau)$

Es decir, si nos desplazamos $+\tau$ o $-\tau$ obtenemos los mismos valores de autocorrelación

En la Fig. 101 se muestra un ejemplo de función de autocorrelación de voz humana

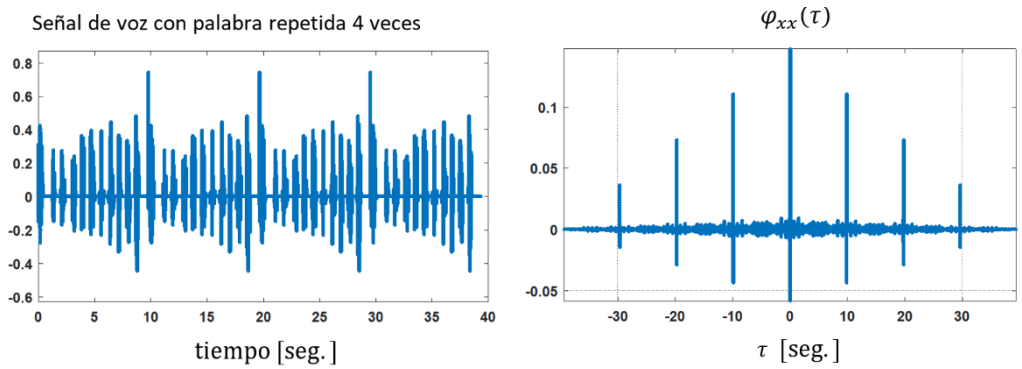


Fig. 101 – Ejemplo de función de autocorrelación de voz humana. Izq.) señal temporal, derecha) autocorrelación

Función de covarianza en procesos ergódicos en sentido amplio

$\Psi_{xx}(\tau) = E\{(x(t) - \mu_x).(x(t-\tau) - \mu_x)\}$

$\Psi_{xx}(\tau) = E\{x(t).x(t-\tau) - \mu_x.x(t) - \mu_x.x(t-\tau) + {\mu_x}^2\}$

$\Psi_{xx}(\tau) = E\{x(t).x(t-\tau)\} - \mu_x.E\{x(t)\} - \mu_x.E\{x(t-\tau)\} + {\mu_x}^2$

$\Psi_{xx}(\tau) = E\{x(t).x(t-\tau)\} - 2.{\mu_x}^2 + {\mu_x}^2$

$\Psi_{xx}(\tau) = \varphi_{xx}(\tau) - {\mu_x}^2$

Función de correlación cruzada

$\varphi_{xy}(t_1, t_2) = E\{x(t_1).y(t_2)\}$

$\varphi_{xy}(\tau) = E\{x(t+\tau).y(t)\} = E\{x(t).y(t-\tau)\}$; en ambos casos el desplazamiento es τ

Al ser ergódico se puede calcular como

$\overline{f\big(x_i(t), y_i(t-\tau)\big)} = \lim_{T\to\infty} \frac{1}{2.T} \int_{-T}^{T} f\big(x_i(t), y_i(t-\tau)\big).dt$

Densidad espectral de potencia (DEP)

El Teorema de Wiener-Khinchin , también es conocido como teorema de Wiener-Khintchine, teorema de Wiener-Khinchin-Einstein o teorema de Khinchin-Kolmogorov.

Este teorema dice que la transformada Fourier de la función de autocorrelación es la DEP.

$$S_{xx}(w) = F\{\varphi_{xx}(\tau)\}$$

A su vez, si aplicamos la Transformada Inversa de Fourier a la DEP obtenemos la autocorrelación: $F^{-1}\{S_{xx}(w)\} = \varphi_{xx}(\tau)$

La función de correlación es par, resulta: $Im\{S_{xx}(w)\} = 0$

Densidad espectral de potencia cruzada

$$S_{xy}(w) = F\{\varphi_{xy}(\tau)\}$$

Energía

La autocorrelación evaluada en $\tau = 0$ nos da la energía

$$E\{|x(t)|^2\} = \varphi_{xx}(0) = \frac{1}{2.\pi}\int_{-\infty}^{\infty} S_{xx}(w).dw$$

Ejemplo de DEP (Densidad espectral de potencia) para voz humana

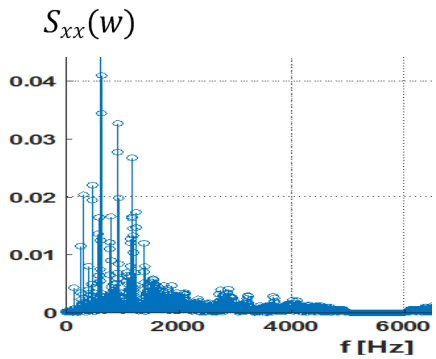


Fig. 102 – Densidad espectral de potencia (DEP) para voz humana

Ruido blanco

$S_{nn}(w) = N_0$

La DEP del ruido blanco n resulta constante. Se tiene la misma potencia en todo el espectro resulta No correlacionado

$$\varphi_{nn}(\tau) = F^{-1}\{N_0\} = N_0.\delta(\tau)$$

No es realizable ya que teóricamente tiene potencia infinita

Su Densidad Espectral de Potencia (DEP o DSP) es constante. La señal abarca todas las frecuencias con potencia constante en todo el rango de frecuencias. Por ejemplo, la potencia contenida en la señal en la banda 60 Hz a 80 Hz es la misma a la contenida en 5000 Hz a 5020 Hz para el ruido blanco. La señal no tiene frecuencia máxima, ya que abarca todo el espectro.

Ruidos de color

Son señales cuyo espectro no son planos, se dice que está coloreado.

- **Ruido Rosa**: su DEP es proporcional a $1/f$
- **Ruido Marrón**: su DEP es proporcional a $1/f^2$
- **Ruido Azul**: su DEP es proporcional a f
- **Ruido Violeta**: su DEP es proporcional a f^2

En la Fig. 103 se muestran simulaciones de diferentes ruidos mediante Matlab®, donde se grafica se DEP en escala logarítmica

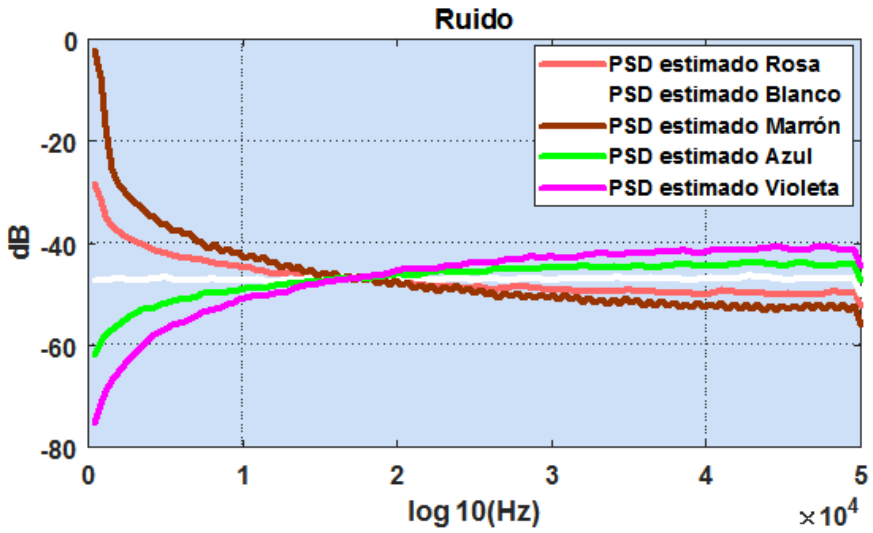


Fig. 103 – DEP (PSD) para diferentes ruidos

Decibelio: el decibelio o decibel es equivalente a la décima parte de un Bel. El Bel se calcula como el logaritmo en base 10 correspondiente a la relación de dos potencias o 2 intensidades. No se usa el Bel, ya que es una unidad muy grande, sino que se usa el decibel o también llamado decibelio

$$L = 10.log\frac{P^2}{P_r} = 20.log\frac{P}{P_r}$$

Para dos valores de potencia distintos P_1 y P_2, o dos tensiones V_1 y V_2, o dos intensidades I_1 e I_2, su relación en Db se puede expresar mediante la siguiente expresión:

$$db = 10.log_{10}\frac{P_1}{P_2}$$ (si se comparan Potencias)

$$db = 20.log_{10}\frac{V_1}{V_2}$$ (si se comparan Tensiones)

$$db = 20.log_{10}\frac{I_1}{I_2}$$ (si se comparan Corrientes)

Procesos estocásticos discretos

Se aplican las mismas definiciones de estacionariedad y ergodicidad que para procesos estocásticos continuos con la salvedad del tiempo discreto

Conclusiones

En teoría, el proceso estocástico corresponde a infinitas realizaciones diferentes del proceso. Cada realización del proceso estocástico se denomina serie temporal. El proceso estocástico lo representaremos de la siguiente manera:

$x[n], x[n-1], \ldots\ldots, x[n-M]$, es una secuencia que representa una serie de tiempo compuesta por la observación presente $x[n]$ realizada en el tiempo n y M observaciones pasadas del proceso realizadas en los tiempos $n-1, \ldots\ldots, n-M$.

El proceso estocástico resulta ***estacionario en sentido estricto*** si mantiene invariante en el tiempo sus propiedades estadísticas. La función densidad de probabilidad conjunta debe mantenerse, independiente del valor de tiempo n que analicemos para una cantidad M fija de observaciones.

Caracterización parcial de un proceso estocástico en tiempo discreto

Generalmente no es posible determinar (mediante medidas adecuadas) la función de densidad de probabilidad conjunta para un grupo arbitrario de observaciones realizadas en un proceso estocástico. En consecuencia, se utiliza una caracterización parcial del proceso especificando su primer y segundo momento.

Dado un proceso estocástico en tiempo discreto, expresado mediante la serie temporal: $x[n], x[n-1], \ldots\ldots, x[n-M]$, utilizaremos $x[n]$ para simplificar la escritura. El valor medio del proceso se obtiene mediante la siguiente expresión:

$$\mu[n] = E\{x[n]\} \qquad (21)$$

Donde $E\{\ \ \}$ denota el Valor esperado estadístico. La función de autocorrelación de un proceso se define como:

$$r[n, n-k] = E\{x[n].x^*[n-k]\} \text{ con } k = 0, \pm 1, \pm 2 \ldots. \qquad (22)$$

El asterisco indica valor complejo conjugado.

La función que calcula la autocovarianza del proceso está definida por:

$$c[n, n-k] = E\{(x[n] - \mu[n]).(x[n-k] - \mu[n-k])^*\} \qquad \text{con} \qquad k = \qquad (23)$$
$0, \pm 1, \pm 2 \ldots.$

Con las ecuaciones (21) y (23) obtenemos la siguiente ecuación que relaciona el valor medio y las funciones de autocorrelación y autocovarianza:

$$c[n, n-k] = r[n, n-k] - \mu[n].\mu^*[n-k] \qquad (24)$$

Para una caracterización parcial (segundo orden) del proceso, se necesita especificar: 1) el valor medio de la función: $\mu[n]$ y 2) la función de autocorrelación $r[n, n-k]$ o la función de autocovarianza $c[n, n-k]$ para varios valores de n y para el k de interés.

Las funciones de autocorrelación y autocovarianza son iguales cuando $\mu[n] = 0$ para todos los valores de n

Este modo de caracterización parcial presenta 2 ventajas:

a) Resulta acorde en mediciones experimentales.

b) Es muy adecuado para operaciones lineales en procesos estocásticos.

Para un proceso estocástico en tiempo discreto que es estrictamente estacionario, las ecuaciones anteriores se simplifican, ya que se cumplen dos condiciones

a) La función de valor medio del proceso es una constante:

$$\mu[n] = \mu, \quad \forall n \tag{25}$$

b) Las funciones de autocorrelación y autocovarianza dependen solamente de la diferencia del tiempo de observación n y $n-k$. Es decir, depende del retraso k como se ve a continuación:

$$r[n, n-k] = r[k] \tag{26}$$

$$c[n, n-k] = c[k] \tag{27}$$

Para $k = 0$, correspondiente a diferencia de tiempo cero, $r[0]$ es equivalente al valor cuadrático medio de $x[n]$

$$r[0] = E\{|x[n]|^2\} \tag{28}$$

Así también $c[0]$ es equivalente a la varianza de $x[n]$

$$c[0] = {\sigma_x}^2 \tag{29}$$

Las ecuaciones (25) y (27) no son suficientes para asegurar que el proceso estocástico en tiempo discreto sea estacionario en sentido estricto. Más bien, ***se puede decir que un proceso estocástico en tiempo discreto que no sea estrictamente estacionario, pero que se cumplen estas condiciones, es de sentido amplio o débilmente estacionario.*** Un proceso $\{x[n]\}$, o $x[n]$ para abreviar, es estacionario en el sentido amplio si y solo si (Doob, 1953) cumple:

$$E\{|x[n]|^2\} < \infty\,, \quad \forall n \tag{30}$$

Esta condición normalmente se cumple en procesos estocásticos de las ciencias físicas y la ingeniería.

Teorema de la media ergódica

Las expectativas, o promedios conjuntos, de un proceso estocástico son promedios "a través de diferentes realizaciones del proceso" durante un instante fijo de tiempo. Claramente, también podemos definir promedios de muestra a largo plazo, o promedios temporales que son promedios "a lo largo del proceso".

De hecho, los promedios temporales pueden usarse para construir un modelo estocástico de un proceso físico estimando parámetros desconocidos del modelo. Sin embargo, para que tal enfoque sea riguroso, tenemos que demostrar que los promedios temporales convergen con los promedios del

ensamble correspondientes del proceso en algún sentido estadístico. Un criterio popular para la convergencia es el del error cuadrático medio.

En este sentido, considere un proceso estocástico de tiempo discreto $x[n]$ *que es estacionario de sentido amplio*. Sea una constante μ la media del proceso y $c[k]$ su función de autocovarianza para el desplazamiento k. Para una estimación de la media μ, podemos usar el promedio temporal:

$$\hat{\mu}(N) = \frac{1}{N}\sum_{n=0}^{N-1} x[n] \tag{31}$$

Siendo N es el número total de muestras utilizadas en la estimación. Hay que tener en cuenta que la estimación $\hat{\mu}(N)$ es una variable aleatoria con una media y una varianza propias. En particular, encontramos en la ecuación (31) que el valor esperado de $\hat{\mu}(N)$ es:

$$E\{\hat{\mu}(N)\} = \mu \quad , \forall N \tag{32}$$

En este sentido de la Ecuación (32) decimos que el tiempo promedio temporal de $\hat{\mu}(N)$ es un estimador insesgado del valor medio del proceso. Además, decimos que el proceso $x[n]$ es ergódico en la media en sentido del error cuadrático medio si el valor del error cuadrático medio considerando $\hat{\mu}(N)$ y μ se aproxima a cero cuando N tiende a infinito:

$$\lim_{N\to\infty} E\{|\mu - \hat{\mu}(N)|^2\} = 0 \tag{33}$$

Si usamos la Ecuación (31) y operamos

$$E\{|\mu - \hat{\mu}(N)|^2\} = E\left\{\left|\mu - \frac{1}{N}\sum_{n=0}^{N-1} x[n]\right|^2\right\}$$

$$= \frac{1}{N^2}.E\left\{\left|\sum_{n=0}^{N-1}(x[n]-\mu)\right|^2\right\} = \frac{1}{N^2}.E\left\{\sum_{n=0}^{N-1}\sum_{k=0}^{N-1}(x[n]-\mu).(x[k]-\mu)^*\right\}$$

$$= \frac{1}{N^2}.\sum_{n=0}^{N-1}\sum_{k=0}^{N-1}E\{(x[n]-\mu).(x[k]-\mu)^*\} = \frac{1}{N^2}.\sum_{n=0}^{N-1}\sum_{k=0}^{N-1}c[n-k]$$

Tomando $l = n - k$, obtenemos:

$$E\{|\mu - \hat{\mu}(N)|^2\} = \frac{1}{N}\sum_{l=-N+1}^{N-1}\left(1 - \frac{|l|}{N}\right).c[l]$$

Teniendo en cuenta la condición de suficiencia para que el proceso $x[n]$ resulte ergódico en la media analizando el error cuadrático medio Ecuación (33), se obtiene:

$$\lim_{N\to\infty}\frac{1}{N}\sum_{l=-N+1}^{N-1}\left(1 - \frac{|l|}{N}\right).c[l] = 0 \tag{34}$$

En otras palabras, si el proceso $x[n]$ no está correlacionado asintóticamente en el sentido de la Ecuación (34), entonces el promedio temporal $\hat{\mu}(N)$ del proceso converge al promedio del ensamble μ en el sentido del error cuadrático medio. Este es el enunciado de una forma particular del teorema de la media ergódica (Gray y Davisson, 1986).

El uso del teorema de la media ergódica puede extenderse a otros promedios temporales del proceso. Considere, por ejemplo, el siguiente promedio temporal utilizado para estimar la función de la autocorrelación de un proceso estacionario de sentido amplio:

$$\hat{r}(k,N) = \frac{1}{N}\sum_{n=0}^{N-1} x[n].x[n-k] \quad , \quad 0 \le k \le N-1 \tag{35}$$

Se dice que el proceso $x[n]$ resulta ergódico en la correlación en el sentido del error cuadrático medio si el valor cuadrático medio de la diferencia entre el valor verdadero $r(k)$ y la estimación $\hat{r}(k,N)$ se acerca a cero cuando el número de muestras N tiende infinito.

Sea $z[n,k]$ un nuevo proceso estocástico de tiempo discreto relacionado con el proceso original $x[n]$ como se muestra:

$$z[n,k] = x[n].x[n-k] \tag{36}$$

Por lo tanto, al sustituir $z[n,k]$ por $x[n].x[n-k]$, podemos usar el teorema de la media ergódica para establecer las condiciones para que $z[n,k]$ sea ergódico en la media o, de manera equivalente, para que $x[n]$ sea ergódico en la correlación.

En la estimación de parámetros, la extracción de la media (es decir, el componente de CC) es la operación de preprocesamiento más importante realizada en una serie temporal antes de calcular la estimación que nos interesa. Este preprocesamiento da como resultado una serie temporal residual con media cero, donde el cálculo de la función de autocorrelación para diferentes retrasos es suficiente para la caracterización parcial del proceso.

De ahora en adelante, asumimos que el proceso estocástico $x[n]$ tiene media cero, como se muestra a continuación:

$$E\{x[n]\} = 0 \quad ; \quad \forall n$$

El conjunto de autocorrelaciones de $x[n]$ para un número finito de retrasos define la matriz de correlación del proceso.

Matriz de correlación

Dado un vector de muestras $\mathbf{x}[n]$ de tamaño $M\ x\ 1$, que representa una serie temporal con valor medio cero se expresa de la siguiente manera:

$$\mathbf{x}[n] = \left[x[n], x[n-1], \dots\dots, x[n-M+1]\right]^T \tag{37}$$

T indica matriz traspuesta

La matriz de correlación se define de la siguiente manera:

$$R = E\{x[n].x^H[n]\} \tag{38}$$

H indica trasposición Hermitiana, es decir matriz traspuesta y conjugada. Mediante las ecuaciones (37) y (38) y teniendo en cuenta la condición de proceso estacionario en sentido amplio se obtiene la matriz de correlación R, en donde todos los elementos de su diagonal $r(0)$ siempre son reales y los demás elementos pueden ser complejos

$$R = \begin{bmatrix} r(0) & r(1) & r(M-1) \\ r(-1) & r(0) & r(M-2) \\ \ldots. & & \ldots\ldots\ldots\ldots \\ r(-M+1) & r(-M+2) & r(0) \end{bmatrix} \quad (39)$$

La matriz de correlación R resulta muy importante en el análisis y diseño de filtros. La matriz de correlación de un ***proceso estocástico estacionario de tiempo discreto*** **cumple las siguientes propiedades**:

a) R es Hermitania, esto implica que $R^H = R$.
Además, la $r(-k) = r^*(k)$, siendo $r(k)$ la función de autocorrelación de $x[n]$ para un desplazamiento k. Para un proceso estacionario amplio se necesitan solamente M funciones de autocorrelación $r(k)$, con $k = 0, 1, \ldots. M-1$. Con esta propiedad la función de autocorrelación para procesos estocásticos en sentido amplio se escribe de la siguiente manera:

$$R = \begin{bmatrix} r(0) & r(1) & r(M-1) \\ r^*(1) & r(0) & r(M-2) \\ \ldots. & & \ldots\ldots\ldots\ldots \\ r^*(M-1) & r^*(M-2) & r(0) \end{bmatrix} \quad (40)$$

Si las funciones de autocorrelación $r(k)$ son reales, entonces la matriz resulta simétrica.

b) La matriz R es una matriz de Toeplitz. Es decir que todos los elementos de esta matriz diagonal son iguales y valen $r(0)$, además en su diagonal paralela todos sus elementos también son iguales y valen $r(1)$, también se cumple que en las demás diagonales paralelas sus elementos son iguales.
Entonces si el proceso estocástico estacionario en sentido amplio de tiempo discreto su matriz de autocorrelación R es una matriz de Toeplitz. También se cumple que si la matriz R es una matriz de Toeplitz entonces el proceso debe ser estacionario en sentido amplio.

c) La matriz R no es negativa

d) La matriz de autocorrelación R de un proceso estocástico estacionario en sentido amplio resulta no singular. Quiere decir que el determinante de R es distinto de 0. Esto se debe a que las muestras $x[n]$ tiene una componente de ruido agregado.
$|r(l)| < r(0) \;\; ; \;\; \forall l \neq 0$
Esto es importante porque implica que R es inversible, es decir existe R^{-1} y se calcula de la siguiente manera:
$R^{-1} = \frac{adj(R)}{\det(R)}$
Siendo $adj(R)$ la matriz adjunta de R o matriz de cofactores, se consigue sustituyendo cada término aij de R por el cofactor aij de R

e) Si los elementos que constituyen el vector de observación de un proceso estocástico de tiempo discreto estacionario se reordenan hacia atrás, el efecto es equivalente a la transposición de la matriz de correlación del proceso.
Sea $\mathbf{x}^{\boldsymbol{B}}[n]$ vector de M-por-1 obtenido reordenando hacia atrás los elementos que constituyen el vector de observación $\mathbf{x}[n]$. Se tiene

$$\mathbf{x}^{\boldsymbol{BT}}[n] = \left[x[n-M+1], x[n-M+2]x[n], x[n-1], \ldots\ldots, x[n]\right] \quad (41)$$

Si se calcula la autocorrelación de $x^B[n]$ se obtiene la matriz traspuesta de correlación del proceso:

$$E\{x^B[n].x^{BH}[n]\} = R^T \qquad (42)$$

f) Las matrices de correlación R_M y R_{M+1} de un proceso estocástico de tiempo discreto estacionario, pertenecientes a las observaciones M y $M+1$ del proceso, respectivamente, están relacionadas por:

$$R_{M+1} = \begin{bmatrix} r(0) & r^H \\ r & R_M \end{bmatrix} \qquad (43)$$

Modelos estocásticos

Los procesos estocásticos se pueden representar mediante un modelo basado en la propuesta de Yule (1927). Esta propuesta dice que se puede generar una serie temporal $u[n]$ que consta de observaciones altamente correlacionadas aplicando una serie de entradas o "choques" estadísticamente independientes a un filtro lineal, como se muestra en la Fig. 104. Las entradas son variables aleatorias con distribución fija, generalmente gaussiana con media cero y varianza constante. Esta serie de variables aleatorias constituye un proceso puramente aleatorio, comúnmente conocido como ruido blanco gaussiano. Debido al valor medio cero de la entrada $v[n]$, podemos describir la siguiente ecuación:

$$E\{v[n]\} = 0 \quad ; \ \forall n \qquad (44)$$

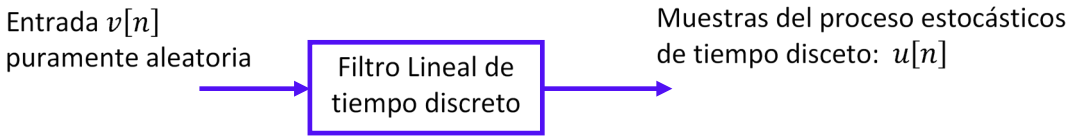


Fig. 104 - Modelo Estocástico

Debido a las características de $v[n]$, siendo σ_v^2 la varianza del ruido, teniendo en cuenta que el ruido blanco tiene un espectro de potencia plano, se obtiene:

$$E\{v[n]v^*[k]\} = \begin{cases} \sigma_v^2 \ ; & k = n \\ 0 \ ; & k \neq n \end{cases} \qquad (45)$$

En general la relación entre las entradas y salidas de un proceso estocásticos se puede describir de la siguiente forma:

$$\left[\begin{array}{c}\text{Valor actual de la salida del modelo estocástico}\end{array}\right] + \left[\begin{array}{c}\text{Combinación lineal de Valores pasados de salida del modelo}\end{array}\right] = \left[\begin{array}{c}\text{Combinación lineal de Valores pasados y presentes de entrada del modelo}\end{array}\right]$$

Un proceso estocástico así descrito se denomina proceso lineal.

La estructura del filtro lineal en la Fig. 104 está determinada por la manera en se formulan las dos combinaciones lineales indicadas en la ecuación (46). Por tanto, podemos identificar tres tipos populares de modelos estocásticos lineales:

1. Modelos autoregresivos (AR), en los que no se utilizan valores pasados de la entrada del modelo.

2. Modelos de tipo moving average (MA) o promedios móviles, en los que no se utilizan valores pasados de la salida del modelo.

3. Modelos mixtos Autoregresivos-moving average (ARMA), en los que la descripción de la ecuación (46) se aplica en su forma completa. Por lo tanto, esta clase de modelos estocásticos incluye modelos autoregresivos y de promedios móviles (MA) como casos particulares.

Procesos Autoregresivos (AR)

La serie temporal $u[n], u[n-1], \ldots\ldots, u[n-M]$ representa la realización de un proceso autoregresivo (AR) de orden M si satisface la siguiente ecuación en diferencias:

$$u[n] + a_1^*.u[n-1] + \ldots\ldots + a_M^*.u[n-M] = v[n] \qquad (47)$$

Donde $a_1, a_2, \ldots\ldots, a_M$ son las constantes o parámetros AR, y $v[n]$ es el ruido blanco. El término $a_k^*.u[n-k]$ es la versión escalar del producto interno de a_k y $u[n-k]$, con $k = 1,2 \ldots\ldots, M$

Para explicar el término autoregresivo, tomamos $w_k = -a_k$ y reescribimos la ecuación (47)

$$u[n] = w_1^*.u[n-1] + w_2^*.u[n-2] + \ldots\ldots + w_M^*.u[n-M] + v[n] \qquad (48)$$

Se observa que el valor presente de $\boldsymbol{u[n]}$ es una combinación lineal finita de los valores pasados del proceso $\boldsymbol{u[n-1], u[n-2], \ldots\ldots, u[n-M]}$ más un error $\boldsymbol{v[n]}$, por este motivo se llama "autoregresivo"

El siguiente modelo que relaciona una variable dependiente y con las variables independientes $x_1, x_2, \ldots x_M$ más un error llamado v se denomina modelo de regresión. Se dice que y es una "regresión" de $x_1, x_2, \ldots x_M$

$$y = \sum_{k=1}^{M} w_k^*.x_k + v$$

Volviendo a la ecuación (48), la variable $u[n]$ es una "regresión" sobre valores previos de sí misma de ahí el término "autoregresivo".

Si reescribimos la ecuación (47), siendo $a_0 = 1$, se observa que el primer término es la convolución entre la secuencia $\{u[n]\}$ y un conjunto de parámetros $\{a_n^*\}$

$$\sum_{k=0}^{M} a_k^*.u[n-k] = v[n] \qquad (49)$$

Al tener convolución en el tiempo, tendremos un producto en Transformada Z de las transformadas de $u[n]$ y de $\{a_n^*\}$. Siendo $U(z)$, $H_A(z)$ y $V(z)$ las transformadas Z de $\{u[n]\}$, $\{a_n^*\}$ y $v[n]$ respectivamente. Donde $H_A(z)$ es la transferencia del sistema.

$$H_A(z) = \sum_{n=0}^{M} a_n^*.z^{-n} \tag{50}$$

$$U(z) = \sum_{n=0}^{\infty} u[n].z^{-n} \tag{51}$$

$$V(z) = \sum_{n=0}^{\infty} v[n].z^{-n} \tag{52}$$

Transformando la ecuación (47), obtenemos:

$$H_A(z).U(z) = V(z) \tag{53}$$

La última ecuación puede tener 2 interpretaciones, depende si el proceso $u[n]$ se toma como salida o como entrada del sistema.

Si se considera $u[n]$ como entrada, como se muestra en la Fig. 105, a la salida se obtiene ruido blanco. Los parámetros del filtro corresponden uno a uno con los de la entrada $u[n]$. De esta manera se obtiene un *analizador de procesos AR* con transferencia $H_A(z) = \frac{V(z)}{U(z)}$.

La transformada inversa de $H_A(z)$ es $h_A(n)$, es decir, es la respuesta impulsional del analizador de procesos. Al tener respuesta impulsional con duración finita, es FIR (Finite Impulse Response). Sus polos se encuentran en el origen y la transferencia se define por la ubicación de los polos.

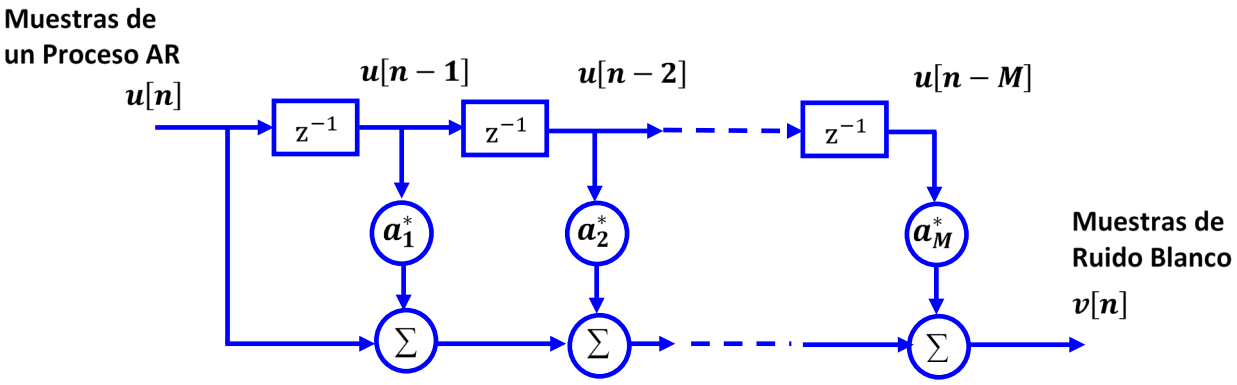


Fig. 105 - Analizador de procesos AR

En cambio si se ingresa ruido blanco a la entrada, se muestra en la Fig. 106, se obtiene un generador de procesos AR, siendo $u[n]$ la salida obtenida. La transferencia $H_G(z)$ es la salida del sistema dividido su entrada en términos de Transformada Z. Usando este concepto y las ecuaciones anteriores, se obtiene:

$$H_G(z) = \frac{U(z)}{V(z)} = \frac{1}{H_A(z)} = \frac{1}{\sum_{n=0}^{M} a_n^*.z^{-n}} \tag{54}$$

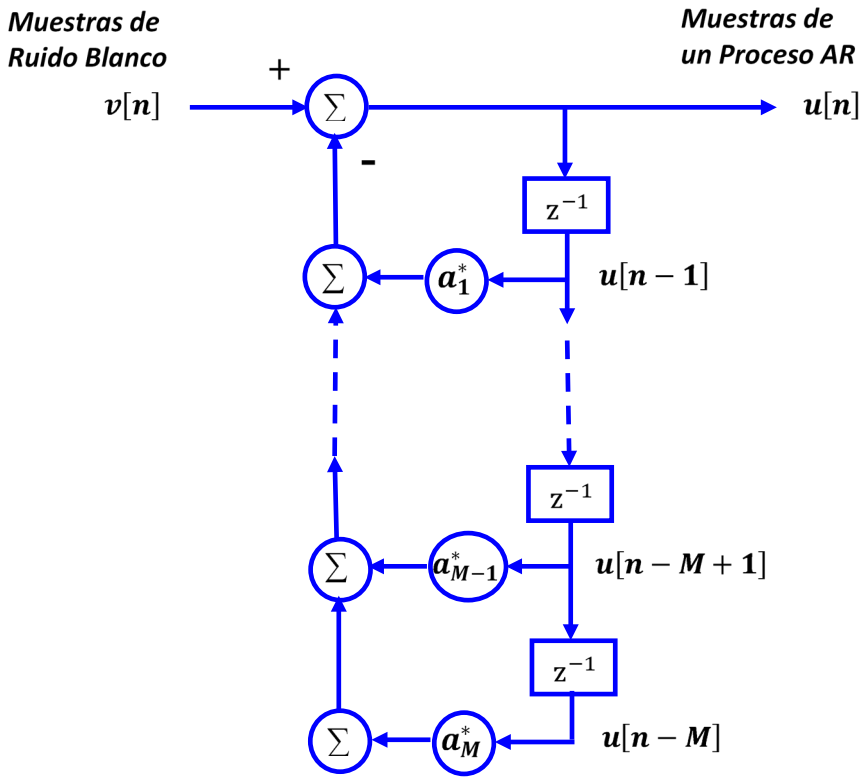


Fig. 106 - Generador de procesos AR

Analizado esta última ecuación (54), existen polos fuera del orígen, esto implica que el generador de procesos AR tiene respuesta infinita al impulso, denominado IIR (Infinite Impulse Response). En este caso la transferencia del sistema esta definida por la ubicación de sus polos:

$$H_G(z) = \frac{1}{(1 - p_1.z^{-1}).(1 - p_2.z^{-1}) \dots \dots \dots (1 - p_M.z^{-1})} \tag{55}$$

Siendo $p_1, p_2, \dots \dots p_M$ los polos de la transferencia $H_G(z)$, se calculan con las raíces del denominador:

$$\sum_{n=0}^{M} a_n^*.z^{-n} = 1 + a_1^*.z^{-1} + a_2^*.z^{-2} + \cdots \dots . + a_M^*.z^{-M} = 0 \tag{56}$$

Para que el generador de procesos AR resulte estable, todos sus polos deben estar dentro de la circunferencia de radio unitario (CRU) en el plano complejo Z. Esto es una condición necesaria y suficiente para tener un generador de procesos estacionario en sentido amplio.

Procesos del tipo moving average (MA)

En los modelos del tipo moving average (MA), es decir media móvil, el filtro lineal de la Fig. 104 resulta tipo FIR (respuesta finita al impulso), o también llamado all-zero por estar caracterizado por la ubicación de sus ceros.

$$\boxed{u[n] = \ v[n] + b_1^*.v[n-1] + \ \ldots\ldots + b_k^*.v[n-K]} \qquad (57)$$

$$u[n] = \sum_{i=0}^{K} b_i^*.v[n-i]$$

Las constantes $b_1, b_2, \ldots\ldots b_K$ se denominan parámetros del proceso MA, siendo k el orden del proceso.

Los términos de la derecha de la ecuación (57), excepto $v[n]$, representan la versión escalar del producto interno. Se utiliza como entrada ruido blanco $v[n]$, con valor medio cero y varianza σ_v^2

Si se dispone de un conjunto completo de ruido blanco, se puede obtener $u[n]$ mediante un promedio ponderado de los valores muestrales $v[n], v[n-1], \ldots\ldots, v[n-k]$

De la ecuación (57) se obtiene fácilmente el modelo MA de la Fig. 107. Si se ingresa ruido blanco a la entrada, a la salida se tiene $u[n]$ mediante el generador de procesos MA de orden k. En cambio, si a la entrada se ingresa $u[n]$ se genera ruido blanco a la salida, se requiere un filtro tipo IIR, del tipo all-poles que se caracteriza por la ubicación de sus polos. Se concluye que los filtros usados en la generación y análisis de un proceso MA son opuestos a los usados en un proceso AR.

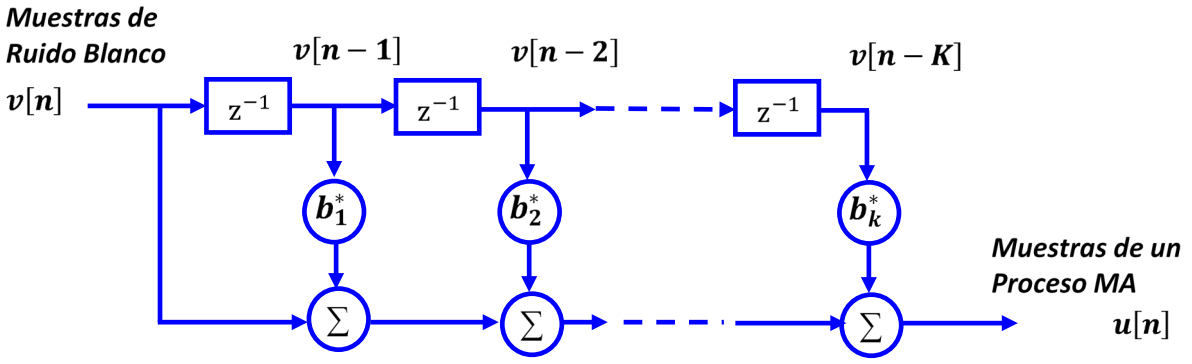


Fig. 107 – Modelo MA usado como generador de procesos

Procesos Autoregresivo-moving average (ARMA)

Mediante la combinación de los modelos AR y MA se genera un proceso $u[n]$ del tipo Autoregresivo-moving average (ARMA). Se utiliza un filtro lineal de tiempo discreto tipo IIR, caracterizado tanto por sus polos como por sus ceros. Mediante el esquema de la Fig. 104, si se ingresa ruido blanco $v[n]$ a la entrada, a la salida se tiene un proceso $u[n]$ del tipo ARMA descripto por la siguiente ecuación:

$$u[n] + a_1^*.u[n-1] + \ \ldots\ldots + a_M^*.u[n-M] = v[n] + b_1^*.v[n-1] + \ \ldots\ldots + b_k^*.v[n-k] \qquad (58)$$

Las constantes $a_1, a_2, \ldots . a_M$ y $b_1, b_2, \ldots . b_k$ se denominan parámetros del proceso ARMA, donde se usan las versiones escalares del producto interno. El orden del proceso se expresa con (M, k). Mediante la ecuación (58) se deduce la estructura del modelo ARMA. Observando la Fig. 106 correspondiente al generador de procesos AR y la Fig. 107 correspondiente a MA se deduce que ambos son casos particulares del generador de procesos ARMA (Fig. 108)

La función de transferencia del generador de procesos ARMA contiene polos y ceros. Para el caso de generador de ruido blanco, también contiene polos y ceros.

Para el cálculo de coeficientes en los modelos MA y ARMA se necesita resolver sistema de ecuaciones no lineales. Para el modelo AR se tienen ecuaciones lineales, el cálculo resulta más simple, y con menor carga computacional. Por este motivo el modelo AR es más utilizado respecto a MA y ARMA. El modelo AR también es más utilizado debido al teorema fundamental del análisis de series temporales.

Para el cálculo numérico computacional de los coeficientes del modelo AR descripto en la Fig. 105, se utiliza un sistema de ecuaciones lineales conocido como ecuaciones de Yule–Walker

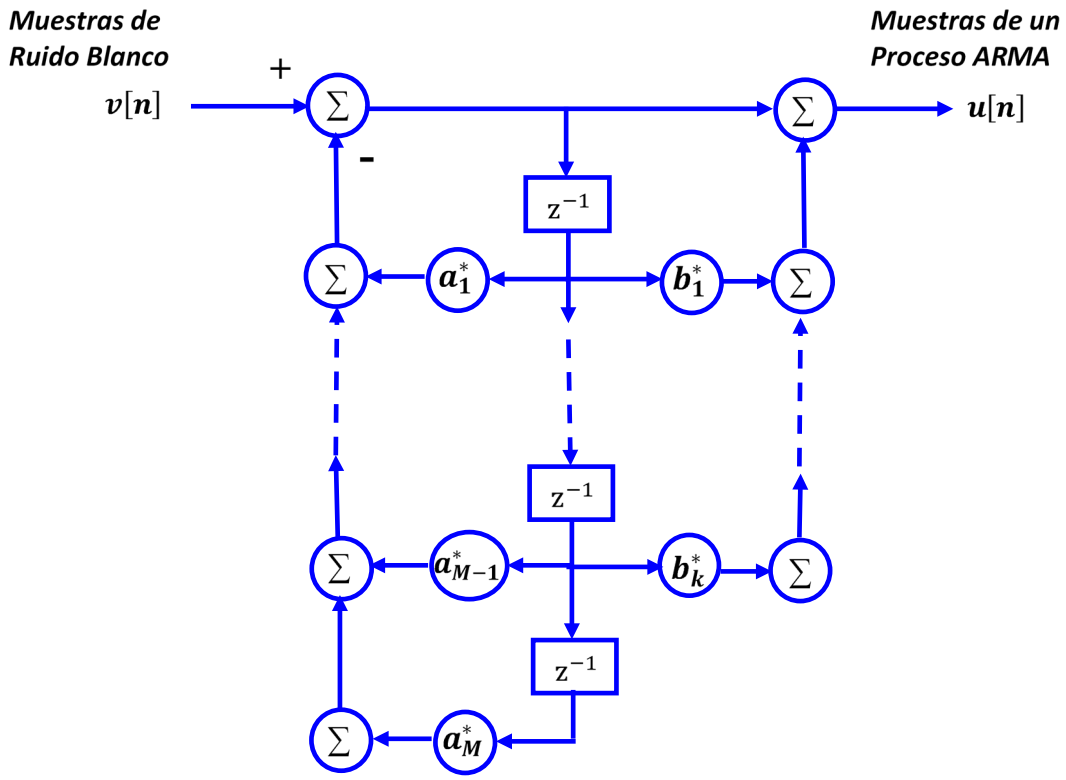


Fig. 108 – Modelo ARMA usado como generador de procesos, de orden $(\boldsymbol{M}, \boldsymbol{k})$*, con* $\boldsymbol{M > k}$

Teorema de Descomposición de Wold

En 1938, Wold demostró un teorema fundamental, el mismo dice que cualquier proceso estocástico de tiempo discreto estacionario puede descomponerse o separarse en la suma de 2 procesos: un *proceso lineal general* y otro *proceso predecible*, ambos procesos independientes, no correlacionados entre sí.

Más precisamente, Wold demostró el siguiente resultado: cualquier proceso estocástico estacionario en tiempo discreto $x[n]$ puede expresarse de la siguiente forma:

$$x[n] = u[n] + s[n] \tag{59}$$

Siendo

a) $u[n]$ y $s[n]$ procesos no correlacionados entre sí
b) $u[n]$ es un *proceso lineal general* representado por un modelo MA, dado por la siguiente ecuación

$$u[n] = \sum_{k=0}^{\infty} b_k^*.v[n-k] \tag{60}$$

Siendo $b_0 = 1$ y $\sum_{k=0}^{\infty}|b_k|^2 < \infty$

Con $v[n]$ ruido blanco no correlacionado con $s[n]$, se cumple: $E\{v[n].s^*[k]\} = 0$ para $\forall(n,k)$

c) $s[n]$ es un proceso predecible, es decir, el proceso se puede predecir a partir de su propio pasado con varianza de predicción cero.

Según la ecuación (60), el proceso lineal general $u[n]$ puede generarse alimentando un filtro todo cero con ruido blanco $v[n]$, como se puede observar en la Fig. 109. Los ceros correspondientes a la función de transferencia de este filtro son iguales a las raíces de la ecuación:

$$B(z) = \sum_{n=0}^{\infty} b_n^*.z^{-n} = 0$$

Una solución de particular interés es un filtro todo cero que es de fase mínima, lo que significa que todos los ceros del polinomio $B(z)$ se encuentran dentro del círculo unitario. En tal caso, podemos reemplazar el filtro todos ceros con un filtro de todos polos equivalente con la misma respuesta al impulso $h_n = b_n$, como en la Fig. 110. Esto significa que, a excepción de un componente predecible, un proceso estocástico de tiempo discreto estacionario también puede representarse como un proceso AR de orden apropiado, con la restricción que se acaba de mencionar en $B(z)$ (ceros en CRU). La diferencia básica entre los modelos MA y AR es que $B(z)$ opera sobre la entrada $v[n]$ en el modelo MA, mientras que la inversa $B^{-1}(z)$ opera sobre la salida $x[n]$ en el modelo AR.

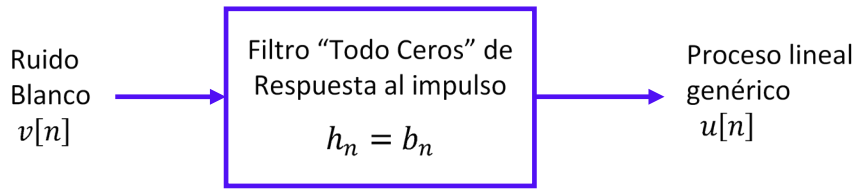


Fig. 109 – Modelo, basado en filtro todo cero, para generar un proceso lineal $\boldsymbol{u[n]}$

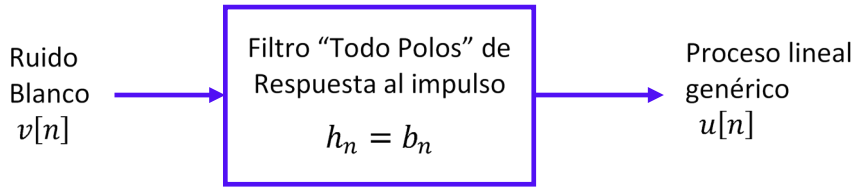


Fig. 110 – Modelo basado en filtro todos polos, para generar el proceso lineal genérico $\boldsymbol{u[n]}$.

Ambos filtros de las figuras tienen exactamente la misma respuesta al impulso.

Estacionariedad asintótica de un proceso autoregresivo

La ecuación (47) de un proceso AR:

$$u[n] + a_1^*.u[n-1] + \ \ldots\ldots + a_M^*.u[n-M] = v[n]$$

es una ecuación lineal en diferencias de coeficiente constante de orden M, tomamos a $v[n]$ como entrada y $u[n]$ la salida. Al usar el método clásico para resolver tal ecuación, podemos expresar formalmente la solución $u[n]$ como la suma de una función complementaria $u_c[n]$ y una respuesta o solución particular $u_p[n]$:

$$u[n] = u_c[n] + u_p[n] \qquad (61)$$

La solución $u[n]$ se evlua en 2 pasos:

a) La solución de $u_c[n]$ corresponde a la solución homogénea

$$u[n] + a_1^*.u[n-1] + \ \ldots\ldots + a_M^*.u[n-M] = 0$$

Generalmente $u_c[n]$ tiene la siguiente forma:

$$u_c[n] = B_1.p_1^n + B_2.p_2^n + \cdots \ldots + B_M.p_M^n \qquad (62)$$

Con $B_1, B_2, \ldots, B_M$ son constantes arbitrarias y $p_1, p_2, \ldots, p_M$ son las raíces de la ecuación (56):

$$1 + a_1^*.z^{-1} + a_2^*.z^{-2} + \cdots \ldots . + a_M^*.z^{-M} = 0$$

b) Solución particular: $u_p[n]$

También se puede utilizar la forma clásica:

$$u[n] = u_H[n] + u_p[n]$$

Si se imponen condiciones iniciales nulas, se tiene un proceso no estacionario

Reflexionando, está claro que esto debe ser así, ya que le hemos dado un "estado especial" al punto de tiempo n = 0 y la propiedad de invariancia bajo un desplazamiento del origen del tiempo no puede ser válida, incluso para momentos de segundo orden. Sin embargo, si el proceso $u[n]$ es capaz de "olvidar" sus condiciones iniciales, el proceso resultante es asintóticamente estacionario en el sentido de que se establece en un comportamiento estacionario cuando n se acerca al infinito (Priestley, 1981). Este requisito puede lograrse eligiendo los parámetros del modelo AR de la Fig. 106 correspondiente a un generador de procesos AR, de manera que la función complementaria $u_c[n]$ decaiga a cero cuando n se acerque al infinito. De ecuación (62), vemos que, para constantes arbitrarias en la ecuación, este requisito se cumple si y solo si

$$\boxed{|p_k| < 1 \ \ ; \forall k}$$

Por lo tanto, para la estacionariedad asintótica del proceso estocástico de tiempo discreto representado por la solución $u[n]$, es necesario que todos los polos del filtro en el modelo AR estén dentro del círculo de radio unitario en el plano z. Este requisito es intuitivamente satisfactorio.

Función de correlación de un proceso AR asintóticamente estacionario

Suponiendo que se satisfaga la condición de estacionariedad asintótica, podemos obtener una relación recursiva importante para la función de autocorrelación del proceso AR resultante. Comenzamos multiplicando la ecuación (47) en ambos lados: $u[n] + a_1^*.u[n-1] + \ldots\ldots + a_M^*.u[n-M] = v[n]$

por $u^*[n-l]$, y luego aplicamos el operador de Valor Esperado, obteniendo así:

$$E\left\{\sum_{k=0}^{M} a_k^*.u[n-k].u^*[n-l]\right\} = E\{v[n].u^*[n-l]\} \tag{63}$$

Reacomodando la sumatoria y teniendo en cuenta que $E\{u[n-k].u^*[n-l]\}$ es la función de autocorrelación del proceso AR para desplazamiento $l-k$

El término de la derecha: $E\{v[n].u^*[n-l]\} = 0$ para $l > 0$ debido a que no están correlacionadas

La ecuación (63) se simplifica de la siguiente forma:

$$\sum_{k=0}^{M} a_k^*.r[l-k] = 0 \quad ; l > 0 \tag{64}$$

Con $a_0 = 1$. La autocorrelación del proceso AR cumple la siguiente ecuación:

$$r[l] = w_1^*.r[l-1] + w_2^*.r[l-2] + \cdots\ldots\ldots + w_M^*.r[l-M] \;\; ; \;\; l > 0 \tag{65}$$

Con $w_k = -a_k$. La ecuación anterior es análoga a la ecuación en diferencias del proceso AR

Calculando función de correlación, la ecuación anterior la expresamos de la siguiente manera:

$$r[m] = \sum_{k=1}^{M} C_k.p_k^m \tag{66}$$

Siendo $C_1, C_2, \ldots\ldots C_M$ constantes y $p_1, p_2, \ldots\ldots p_M$ son las raíces de la ecuación característica (56):

$$1 + a_1^*.z^{-1} + a_2^*.z^{-2} + \cdots\ldots + a_M^*.z^{-M} = 0$$

Encontramos que, en general, la función de autocorrelación de un proceso AR asintóticamente estacionario consiste en una mezcla de exponenciales amortiguadas y ondas sinusoidales amortiguadas.

Ecuaciones de Yule-Walker

Para definir un modelo AR de orden M descripto en la Fig. 106, hay que especificar 2 conjuntos de parámetros:

- Los coeficientes AR: $a_1, a_2, \ldots\ldots a_M$
- La varianza σ_v^2 del ruido blanco $v[n]$ usado como excitación

Veremos un conjunto por vez.

Partimos de la ecuación de $r[l]$, (ver ecuación 65):

$$r[l] = w_1^*.r[l-1] + w_2^*.r[l-2] + \cdots\ldots\ldots + w_M^*.r[l-M]$$

Para $l = 1,2, \ldots . M$ obtenemos un conjunto de M ecuaciones simultáneas con los valores $r(0), r(1), \ldots \ldots r(M)$ de la función de autocorrelación del proceso AR como valores conocidos. Y los parámetros AR $a_1, a_2, \ldots \ldots a_M$ como incógnitas. Este conjunto de ecuaciones puede expresarse en forma matricial, obteniendo el conjunto de ecuaciones de Yule-Walker:

$$\begin{bmatrix} r(0) & r(1) & & r(M-1) \\ r^*(1) & r(0) & & r(M-2) \\ & \ldots. & & \ldots \ldots \ldots \ldots \\ r^*(M-1) & r^*(M-2) & & r(0) \end{bmatrix} . \begin{bmatrix} w_1 \\ w_2 \\ \\ w_M \end{bmatrix} = \begin{bmatrix} r^*(1) \\ r^*(2) \\ \\ r^*(M) \end{bmatrix} \tag{67}$$

Con $w_k = -a_k$. Teniendo en cuenta la matriz de autocorrelación:

$$R = \begin{bmatrix} r(0) & r(1) & & r(M-1) \\ r^*(1) & r(0) & & r(M-2) \\ & \ldots. & & \ldots \ldots \ldots \ldots \\ r^*(M-1) & r^*(M-2) & & r(0) \end{bmatrix}$$

Las ecuaciones de Yule-Walker (YW) se pueden escribir de la siguiente manera:

$$\boxed{R.w = r} \tag{68}$$

Asumiendo que la matriz R es no singular, es decir admite matriz inversa, se obtiene:

$$\boxed{w = R^{-1}.r} \tag{69}$$

Siendo $w = [w_1, w_2, \ldots \ldots w_M]^T$, y

$r^H = [r(1), r(2), \ldots \ldots, r(M)]$, donde H denota Hermitania: traspuesta y conjugada

$r(0)$ es la autocorrelación con corrimiento 0

A partir de estas dos ecuaciones, vemos que podemos determinar de forma única tanto la matriz R y el vector r, dada la secuencia de autocorrelación $r(0), r(1), \ldots \ldots, r(M)$. Por lo tanto, usando la ecuación (69), podemos calcular el vector de coeficientes w y, por lo tanto, los coeficientes AR: $\boxed{a_k = -w_k}$ con $k = 1,2, \ldots . M$. En otras palabras, se deduce que existe una relación única entre los coeficientes $a_1, a_2, \ldots \ldots, a_M$ del modelo AR y los coeficientes de correlación normalizados $p_1, p, \ldots \ldots, p_M$ del proceso AR $u[n]$, como se muestra a continuación:

$$\{a_1, a_2, \ldots \ldots, a_M\} \rightleftharpoons \{p_1, p, \ldots \ldots, p_M\} \tag{70}$$

Siendo p_k el componente k, dado por:

$$\boxed{p_k = \frac{r(k)}{r(0)}} \text{; con } k = 1,2, \ldots . M \tag{71}$$

<u>Varianza del Ruido Blanco</u>

Para $l = 0$, tomamos el Valor esperado de la parte derecha de la ecuación (63):

$E\{\sum_{k=0}^{M} a_k^*.u[n-k].u^*[n-l]\} = E\{v[n].u^*[n-l]\}$

Y teniendo en cuenta la ecuación (47): $u[n] + a_1^*.u[n-1] + \ldots \ldots + a_M^*.u[n-M] = v[n]$, se obtiene:

$$E\{v[n].u^*[n]\} = E\{v[n].v^*[n]\} = \sigma_v^2 \tag{72}$$

Siendo σ_v^2 la varianza de media cero del ruido blanco. Al tomar $l = 0$ en la ecuación (63) y realizando conjugación compleja en ambos lados, se obtiene:

$$\sigma_v^2 = \sum_{k=0}^{M} a_k . r(k) \tag{73}$$

Con $a_0 = 1$ para la varianza de ruido blanco. Es decir, tomando las correlaciones $r(0), r(1), \dots \dots, r(M)$ se puede determinar la varianza de ruido blanco σ_v^2

Ejemplo de Cálculo para proceso Autoregresivo AR de orden 2

Se muestra un ejemplo de orden 2 para un generador de procesos AR con constantes reales, basado en la Fig. 111. La descripción en el dominio temporal del proceso se expresa con la siguiente ecuación en diferencias:

$$u[n] + a_1 . u[n-1] + a_2 . u[n-2] = v[n]$$

Siendo $v[n]$ un proceso de ruido blanco que tiene valor medio cero y varianza σ_v^2. La Fig. 112 muestra una realización de este proceso de ruido blanco. Se elige varianza σ_v^2 de manera de tener varianza de $u[n]$ igual a uno.

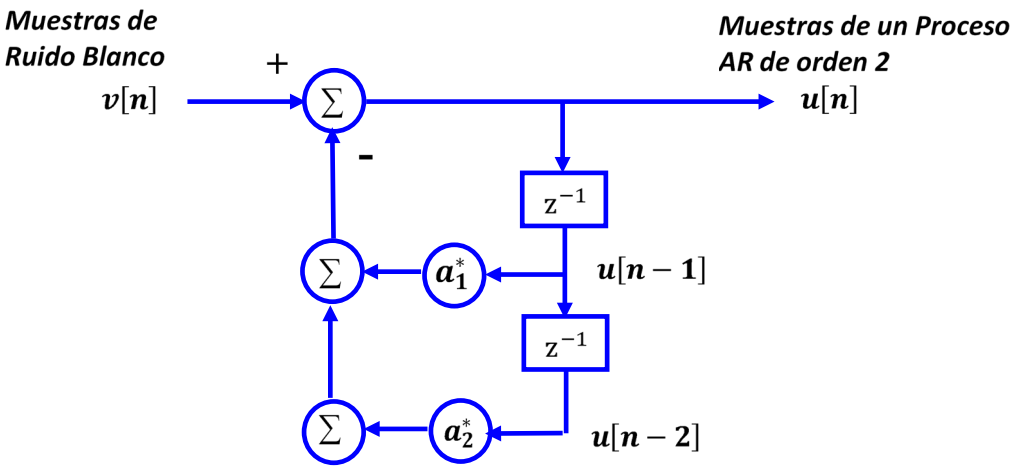


Fig. 111 – Modelo de generador de procesos de tipo AR con orden 2

a) Analizamos las condiciones de estacionariedad asintótica

El proceso mencionado tiene la siguiente ecuación característica:

$$1 + a_1 . z^{-1} + a_2 . z^{-2} = 0$$

Las raíces de la ecuación son $p_1, p_2 = p_{1,2} = \frac{-a_1 \pm \sqrt{{a_1}^2 - 4.a_2}}{2}$

Para asegurar estacionariedad asintótica, se requiere que estas raíces estén dentro de la circunferencia de radio unitaria (CRU), es decir el módulo de ambas debe ser menor a 1:

$$|p_{1,2}| < 1 \;\Rightarrow\; \left|\frac{-a_1 \pm \sqrt{{a_1}^2 - 4.a_2}}{2}\right| < 1 \quad ;$$

Si $a_1 = 0$: $\quad |-j.\sqrt{a_2}| < 1 \quad ; \quad \boxed{-1 \leq a_2 \leq 1}$

$\left|-a_1 \pm \sqrt{{a_1}^2 - 4.a_2}\right| < 2 \quad ; \quad \Rightarrow \quad \left|-a_1 - \sqrt{{a_1}^2 - 4.a_2}\right| < 2 \quad ; \quad \left|a_1 + \sqrt{{a_1}^2 - 4.a_2}\right| < 2 \quad ;$
$\left|a_1 + \sqrt{{a_1}^2 - 4.a_2}\right| < 2 \;\; ; \left|\sqrt{{a_1}^2 - 4.a_2}\right| < 2 - |a_1| \;\; ; {a_1}^2 - 4.a_2 < (2 - |a_1|)^2 \;\; ;$

${a_1}^2 - 4.a_2 < 4 - 4.|a_1| + {a_1}^2 \Rightarrow \qquad \boxed{|a_1| < 1 + a_2}$

Para esto se deben cumplir las condiciones: $-1 \leq a_2 + a_1$, $-1 \leq a_2 - a_1$ y $-1 \leq a_2 \leq 1$

Las constantes a_1 y a_2 deben estar dentro de la región limitada por el triángulo de la Fig. 113

b) Función de Autocorrelación

La función de autocorrelación $r[l]$ de un proceso AR asintóticamente estacionario para el retardo l cumple con la ecuación en diferencias (65):

$$r[l] = w_1^*.r[l-1] + w_2^*.r[l-2] + \cdots \ldots \ldots + w_M^*.r[l-M]$$

$$w_k = -a_k$$

Para un proceso de AR de orden 2 se tiene:

$$r[m] + a_1.r[m-1] + a_2.r[m-2] = 0 \quad ; \; m > 0 \tag{74}$$

Para los valores iniciales, se tiene (se explica más adelante):

$$r[0] = \sigma_u^2 \quad ; \quad r[1] = \frac{-a_1}{1+a_2}\sigma_u^2 \tag{75}$$

Resolviendo la ecuación (74) se tiene:

$$r[m] = \sigma_x^2.\left[\frac{p_1.({p_2}^2 - 1)}{(p_2 - p_1).(p_1.p_2 + 1)}.{p_1}^m - \frac{p_2.({p_1}^2 - 1)}{(p_2 - p_1).(p_1.p_2 + 1)}.{p_2}^m\right]$$

Donde p_1 y p_2 son las raíces definidas anteriormente: $p_{1,2} = \frac{-a_1 \pm \sqrt{{a_1}^2 - 4.a_2}}{2}$

Se pueden tener 2 casos, raíces reales o complejas:

a) Raíces reales, se debe cumplir:

$${a_1}^2 - 4.a_2 > 0$$

Esto corresponde a las regiones 1 y 2 debajo de la parábola en la Fig. 113

En la región 1, la función de autocorrelación permanece positiva y va decayendo, lo que corresponde a una raíz dominante positiva. Ver Fig. 114 (izq.) para los siguientes parámetros AR: $a_1 = -0{,}1$ y $a_2 = -0{,}8$

En la Fig. 112 (b), se muestra la variación temporal de la salida del modelo de la Fig. 111, con los valores $a_1 = -0{,}1$ y $a_2 = -0{,}8$. Esta salida es producida por la entrada de ruido blanco

Para la región 2 de la Fig. 113, la función de autocorrelación alterna en signo a medida que se atenúa, lo que corresponde a una raíz dominante negativa. Esta situación se observa en la Fig. 114 (derecha) para los parámetros AR: $a_1 = 0{,}1$ y $a_2 = -0{,}8$. En la Fig. 112 (a), se muestra la variación temporal de la salida del modelo de la Fig. 111, con los valores a_1 y a_2 asignados. Esta salida es producida por la entrada de ruido blanco.

b) Raíces complejas conjugadas, se debe cumplir:

$${a_1}^2 - 4.a_2 < 0$$

Esto corresponde a las regiones superior a la parábola en la Fig. 113. La función de autocorrelación presenta un comportamiento pseudo periódico, como se puede observar en la Fig. 115, se utilizan estos parámetros $a_1 = -0{,}975$ y $a_2 = 0{,}95$

En la Fig. 112 (c), se muestra la variación temporal de la salida del modelo de la Fig. 111, con los valores mencionados de a_1 y a_2. Esta salida es producida por la entrada de ruido blanco.

Ecuaciones de Yule-Walker para el ejemplo de Cálculo de un proceso Autoregresivo AR de orden 2

La ecuación (67) **de Yule-Walker** para valores reales de autocorrelación y proceso AR de orden: $M = 2$, toma la siguiente forma:

$$\begin{bmatrix} r[0] & r[1] \\ r[1] & r[0] \end{bmatrix} . \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} r[1] \\ r[2] \end{bmatrix} \tag{76}$$

Donde se usa $r[1] = r[-1]$ por ser un proceso con valores reales. Resolviendo la ecuación anterior, mediante Regla de Crámer, se obtiene:

$w_1 = -a_1 = \frac{r[1].r[0] - r[1].r[2]}{r^2[0] - r^2[1]} = \frac{r[1].[r[0] - r[2]]}{r^2[0] - r^2[1]}$

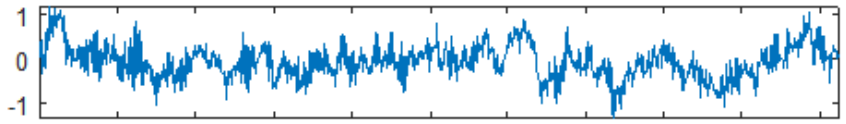
$w_2 = -a_2 = \frac{r[0].r[2] - r^2[1]}{r^2[0] - r^2[1]}$

La autocorrelación en cero es $\sigma_u^2 = r[0]$. Si usamos la ecuación(76), podemos expresar $r[1]$ y $r[2]$ en términos de a_1 y a_2

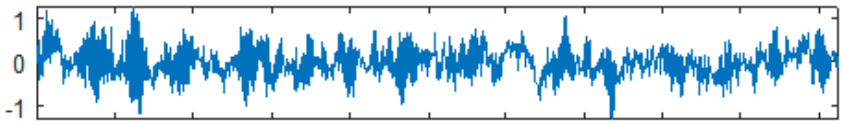
$r[1] = r[0].w_1 + r[1].w_2$; $r[1] = -\sigma_u^2.a_1 - r[1].a_2$; despejamos $r[1]$y en forma similar obtenemos $r[2]$

$$r[1] = \left[\frac{-a_1}{1+a_2}\right].\sigma_u^2 \quad ; \quad r[2] = \left[-a_2 + \frac{{a_1}^2}{1+a_2}\right].\sigma_u^2 \tag{77}$$

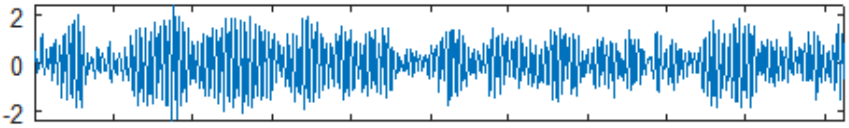
Esta solución explica los valores iniciales de $r[0]$ y $r[1]$ de la ecuación (75)



a) parámetros AR: $a_1 = -0{,}1$ y $a_2 = -0{,}8$



b) $a_1 = -0{,}1$ y $a_2 = -0{,}8$.



c) $a_1 = -0{,}975$ y $a_2 = 0{,}95$

Fig. 112 – (a), (b), (c) salidas correspondientes del modelo AR de orden 2 para los parámetros los distintos valores de $\boldsymbol{a_1}$ y $\boldsymbol{a_2}$ analizados.

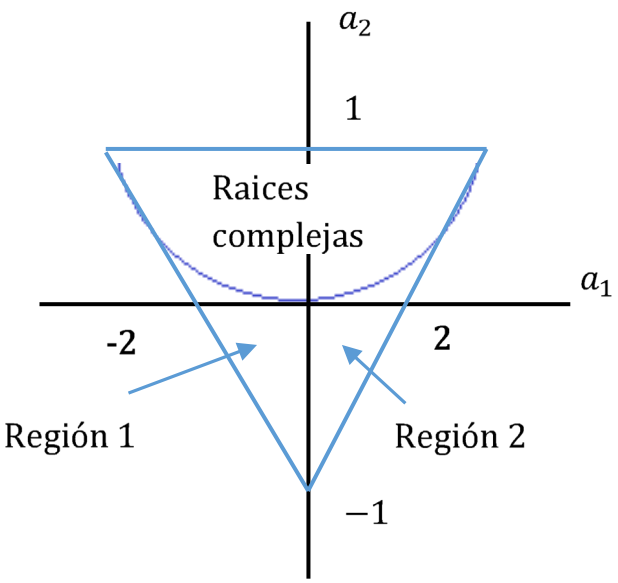


Fig. 113 – Región permitida de los parámetros $\boldsymbol{a_1}$ *y* $\boldsymbol{a_2}$

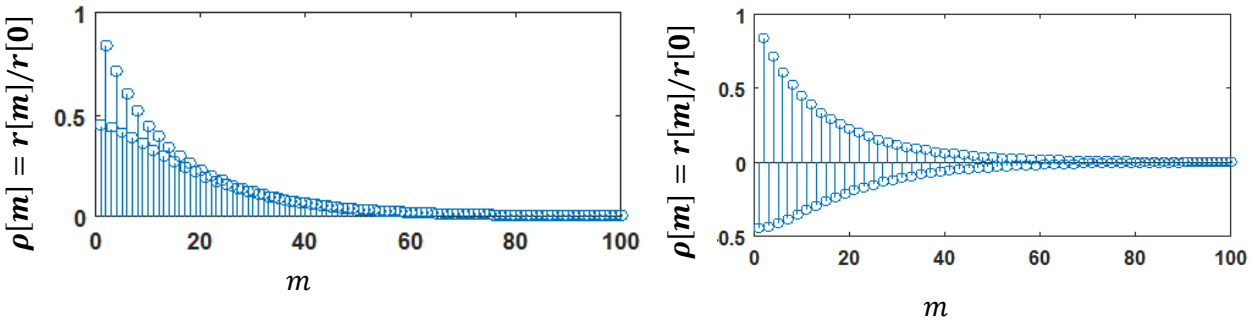


Fig. 114 – Gráficos de la función de autocorrelación normalizada para un proceso de segundo orden con valores reales; izquierda) $\boldsymbol{r(1) > 0}$ *; derecha)* $\boldsymbol{r(1) < 0}$*;*

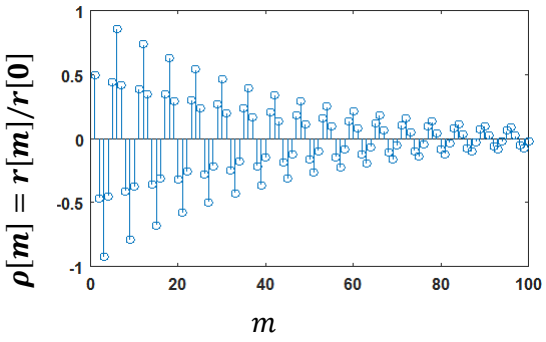


Fig. 115 – Gráficos de la función de autocorrelación normalizada del proceso de segundo orden con valores reales; para raíces complejas conjugadas;

Anteriormente se expresaron las condiciones para la estacionariedad asintótica del proceso AR de segundo orden en términos de los parámetros AR: a_1 y a_2. Se debe cumplir $|p_{1,2}| < 1$, involucra:

$-1 \leq a_2 + a_1$, $-1 \leq a_2 - a_1$ y $-1 \leq a_2 \leq 1$. Usando las expresiones de $r[1]$ y $r[2]$ en términos de a_1 y a_2 dadas en la (77), podemos reformular las condiciones de estacionariedad asintótica:

$$|\rho_{1,2}| < 1 \; ; \quad -1 < \rho_1 < 1 \quad ; \quad -1 < \rho_2 < 1 \tag{78}$$

${\rho_1}^2 < \frac{1}{2}.(1 + \rho_2)$; Coeficientes de correlación normalizados: $\rho_1 = \frac{r[1]}{r[0]}$; $\rho_2 = \frac{r[2]}{r[0]}$

Varianza del ruido blanco para el ejemplo de Cálculo de un proceso Autoregresivo AR de orden 2

Si tomamos la ecuación (73): $\sigma_v^2 = \sum_{k=0}^{M} a_k.r(k)$, para $M = 2$ obtenemos:

$\sigma_v^2 = r[0] + a_1.r[1] + a_2.r[2]$

Reemplazando: $r[1] = \left[\frac{-a_1}{1+a_2}\right].\sigma_u^2$; $r[2] = \left[-a_2 + \frac{{a_1}^2}{1+a_2}\right].\sigma_u^2$ de la ecuación (77) y despejando $\sigma_u^2 = r(0)$, obtenemos:

$$\sigma_u^2 = \frac{1 + a_2}{1 - a_2}.\left(\frac{\sigma_v^2}{(1 + a_2)^2 - {a_1}^2}\right)$$

Para los 3 conjuntos de parametros AR analizados anteriormente,se calcula la varianza de ruido blanco de $v[n]$ y se muestra en la Tabla II. Se asume que $\sigma_u^2 = 1$

$\sigma_v^2 = \frac{1-a_2}{1+a_2}.((1 + a_2)^2 - {a_1}^2).\sigma_u^2$

Tabla II – Parámetros AR y varianza del ruido

a_1	a_2	$\sigma_v^2 = \frac{1 - a_2}{1 + a_2}.((1 + a_2)^2 - {a_1}^2)$
$-0{,}1$	$-0{,}8$	$0{,}27$
$0{,}1$	$-0{,}8$	$0{,}27$
$-0{,}975$	$0{,}95$	$0{,}0731$

Selección del modelo

La representación de un proceso estocástico mediante un modelo lineal se puede utilizar para síntesis o análisis. En síntesis, generamos una serie de tiempo deseada asignando un conjunto prescrito de valores a los parámetros del modelo y alimentándolo con ruido blanco de media cero y varianza prescrita. En el análisis, por otro lado, estimamos los parámetros del modelo procesando una serie de tiempo determinada de longitud finita. En la medida en que la estimación sea estadística, necesitamos una medida adecuada del ajuste entre el modelo y los datos observados. Esto implica que, a menos que tengamos alguna información previa, el procedimiento de estimación debe incluir un criterio para seleccionar el orden del modelo (es decir, los grados de libertad del modelo).

Para el caso de un proceso AR determinado por la ecuación (47), el orden del modelo es igual a M.

$$u[n] + a_1^*.u[n-1] + \ \ldots\ldots + a_M^*.u[n-M] = v[n]$$

En el caso de un proceso MA definido por la ecuación (57), el orden del modelo es igual a K

$$u[n] = \ v[n] + b_1^*.v[n-1] + \ \ldots\ldots + b_k^*.v[n-k]$$

En el caso de un proceso ARMA definido por la ecuación (58), el orden del modelo es igual a (M, k)

$$u[n] + a_1^*.u[n-1] + \ \ldots\ldots + a_M^*.u[n-M] = v[n] + b_1^*.v[n-1] + \ \ldots\ldots + b_k^*.v[n-k]$$

En la literatura se describen varios criterios para seleccionar el orden del modelo (Priestley, 1981; Kay, 1988). En esta sección, describimos dos criterios muy importantes, uno de los cuales fue iniciado por Akaike (1973, 1974) y el otro por Rissanen (1978) y Schwartz (1978); Ambos criterios resultan del uso de argumentos teóricos de información, pero se utilizan formas totalmente distintas.

a) Criterio de Información teórica (AIC) de Akaike

Sea $u_i = u(i)$, $i = 1,2, \ldots . , N$, indica los datos obtenidos por N observaciones independientes de un proceso estocástico de tiempo discreto estacionario y $g(u_i)$ indica la función de densidad de probabilidad de u_i.

Sea $f_U(u_i/\hat{\theta}_m)$ la función densidad de probabilidad condicional de u_i, dado $\hat{\theta}_m$: vector estimado de parámetros que modelan el proceso. Sea m el orden del modelo, de modo que podemos escribir

$$\hat{\theta}_m = [\hat{\theta}_{1m}, \hat{\theta}_{2m}, \ldots . \hat{\theta}_{mm}]^T$$

Existen varios modelos que compiten entre sí para representar el proceso de interés. Un criterio de información teórica (AIC) propuesto por Akaike selecciona el modelo para el cual se tiene valor mínimo de la siguiente función:

$$\boxed{AIC(m) = -2.L(\hat{\theta}_m) + 2.m} \tag{79}$$

La función

$$L(\hat{\theta}_m) = max \sum_{i=1}^{N} ln\left(f_U(u_i/\hat{\theta}_m)\right) \tag{80}$$

donde ln denota el logaritmo natural. El criterio expresado en ecuación (80) se obtiene minimizando la divergencia de Kullback-Leibler, que se utiliza para proporcionar una medida de la divergencia entre la función densidad de probabilidad verdadera "desconocida" $g(x)$ y la función densidad de probabilidad condicional $f_U(u_i/\hat{\theta}_m)$ dado por el modelo de datos observados.

La función $L(\hat{\theta}_m)$, se conoce como logaritmo de estimación de parámetros para máxima verosimilitud (maximum-likelihood estimation)

El segundo término $2.m$, de la ecuación (79) representa una penalización por complejidad del modelo que hace que AIC (m) sea una estimación de la divergencia Kullback-Leibler. El primer término de la ecuación tiende a disminuir rápidamente con el orden del modelo m. Por otro lado, el segundo término aumenta linealmente con m. El resultado es que si graficamos AIC (m) versus m, el gráfico, en general, mostrará un valor mínimo definido, y el orden óptimo del modelo está determinado por el valor de m en el que AIC (m) es mínimo, valor que se denomina AIC mínimo.

La idea es minimizar la información agregada a la serie temporal modelándola como un proceso AR, MA o ARMA de orden finito, ya que cualquier información agregada es información virtualmente falsa para una situación real.

b) Criterio de longitud de descripción mínima (minimum description length (MDL) criterion)

Rissanen (1978, 1989) utilizó un enfoque completamente diferente para resolver el problema de identificación del modelo estadístico. Específicamente, comenzó con la noción de que un modelo puede verse como un dispositivo para describir las características regulares de un conjunto de datos observados, con el objetivo de buscar un modelo que capture mejor las características o restricciones regulares que dan a los datos su estructura especial. Reconociendo que la presencia de restricciones reduce la incertidumbre sobre los datos, observamos que el objetivo puede ser igualmente el de codificar los datos de la manera más corta o menos redundante. El término "codificación", como se usa aquí, se refiere a una descripción exacta de los datos observados. En consecuencia, el número de dígitos binarios necesarios para codificar tanto los datos observados, cuando se aprovechan las restricciones ofrecidas por un modelo, como el modelo en sí puede usarse como criterio para medir la cantidad de las mismas restricciones y, por lo tanto, la bondad del modelo.

Por lo tanto, podemos establecer formalmente el criterio de la longitud mínima de descripción formulado por Rissanen (MDL) de la siguiente manera:

Dado un conjunto de datos de interés y una familia de modelos estadísticos que compiten entre sí, el mejor modelo es el que proporciona la descripción más corta de los datos.

En términos matemáticos, este modelo está definido por (Rissanen, 1978, 1989; Wax, 1995) la siguiente ecuación:

$$MDL = -L\left(\hat{\theta}_m\right) + \frac{1}{2} m.\ln\left(N\right) \tag{81}$$

Siendo m es el número de parámetros ajustados independientemente en el modelo y N es el tamaño de la muestra (número de observaciones). Al igual que con el criterio de la teoría de la información de Akaike, $L\left(\hat{\theta}_m\right)$ es el logaritmo de las estimaciones de máxima verosimilitud (maximum-likelihood) de los parámetros del modelo. Comparamos las ecuaciones:
(79) $AIC(m) = -2.L\left(\hat{\theta}_m\right) + 2.m$

(81): $MDL = -L\left(\hat{\theta}_m\right) + \frac{1}{2} m.\ln\left(N\right)$,

Observamos que la principal diferencia entre los criterios AIC y MDL radica en el término dependiente de la estructura.

Según Rissanen (1989), el criterio MDL ofrece los siguientes atributos:

• El modelo permite la codificación más corta de los datos observados y captura todas las propiedades de los datos observados de la mejor manera posible.

• El criterio MDL es un estimador de orden de modelo consistente en el sentido en que converge al orden del modelo real a medida que se agranda el tamaño de la muestra.

• El modelo es óptimo en el contexto de problemas de regresión lineal y modelos ARMA.

Quizás el punto más significativo para tener en cuenta es que, en casi todas las aplicaciones que involucran el criterio MDL, no se han reportado resultados anómalos o modelos con propiedades indeseables en la literatura.

Notas:

- La idea de longitud mínima para descripción de objetos individuales definibles en forma recursiva proviene de Kolmogorov (1968).

- Schwartz (1989) ha obtenido un resultado similar, utilizando un enfoque bayesiano. En particular, considera el comportamiento asintótico de los estimadores de Bayes bajo una clase especial previa. Estos antecedentes ponen probabilidad positiva en los subespacios que corresponden a los modelos competidores. La decisión se toma seleccionando el modelo que arroja la máxima probabilidad a posteriori.

Resulta que, en el límite de muestras grandes, los dos enfoques adoptados por Schwartz y Rissanen producen esencialmente el mismo resultado. Sin embargo, el enfoque de Rissanen es mucho más general, mientras que el enfoque de Schwartz se limita al caso de que las observaciones sean independientes y provengan de una distribución exponencial.

Ejercicios de Procesos estocásticos

Ejercicio 2.1

Realización de un proceso estocástico en Python

Se desea generar las realizaciones de un Procesos Estocástico correspondiente a la siguiente función:

$$y = 8.\cos(5.t + \theta)$$

Siendo θ una variable aleatoria distribuida uniformemente

Resolución

```
#######  Generación de un Proceso Estocástico   ##############
def boton_05_Proc_Estoc_t ():
    import numpy as np  # importamos
    import matplotlib.pyplot as  plt
    t=np.arange(-1,1,0.0001) #genero rango de -3 a 3 cada 0.0001 unidades
    f1=plt.figure(1)
    #genera las 50 trayectorias"
    for x in range(50):
      y = []   #vacia= lista
      tita = np.random.uniform(0, 2*np.pi) #obtiene un dato aleatorio
      y=8*np.cos(5*t+tita) #  calcula el 'y"" correspondiente"
      plt.plot(t,y, color='black')  # pone el "y" en la grafica
    y= (4/np.pi)*(np.sin (5*t+2*np.pi)-np.sin (5*t)) # obtiene ""y"" de E[x(t)]"
    plt.plot (t,y,color= 'red' , label = 'Valor esperado') # pone el y  en la grafica
    y= (32/np.pi)*(np.pi+0.5*np.sin (10*t+4*np.pi)-0.5*np.sin (10*t)) # obtiene y
    #plt .plot (t,y,color='blue', label='Varianza')  # pone el "y" en la grafica
    plt.title ('Práctica 1 - Distribución uniforme')
```

```
    plt.ylabel('x(t)') ; plt.xlabel('t') ;  plt.legend(loc=4)
    plt.show() #muestra la grafica
boton_05_Proc_Estoc_t ()
```

En la Fig. 116 se muestran los resultados obtenidos

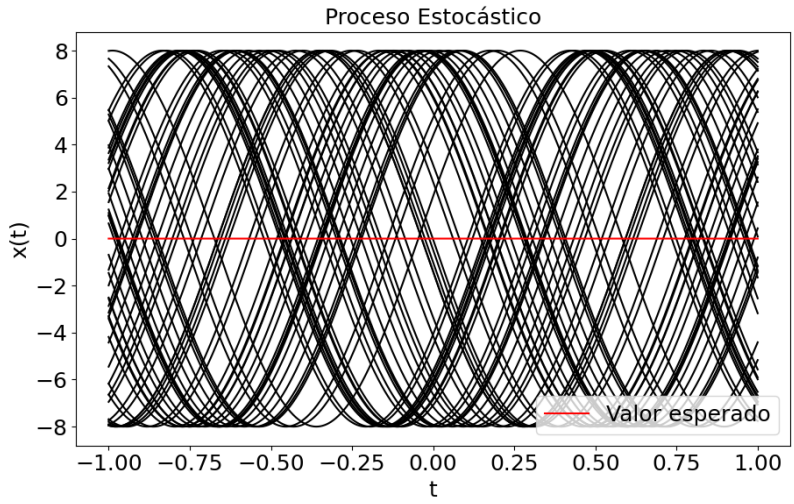


Fig. 116 – Realizaciones de un proceso estocástico

Ejercicio 2.2

Realización de un proceso estocástico en Matlab®

Mediante Matlab® simular un Generador de Señales electrónico Defectuoso. Graficar algunas muestras

Resolución

```
function Ejemplo1
%% Punto 1.a) Generar un ensamble adecuado. Y graficar sus primeras 5 muestras
 n= 1000 ; % 1000 realizaciones
 for j=1:n
  x= generador1 ;
  % x= x + 5*j ; % sacar comentario para generador defectuoso
   X(j, :) = x;
 end
 dt = 0.01 ; t = 0 : dt : (length(x)-1)*dt ;
 figure(10) ;title('Punto a) Grafico de 5 muestras')
 for i=1:5
  subplot(5,1,i); plot(t, X(i, :)); axis tight
 end
```

```
%% Punto 1.b)Evaluar proceso respecto de su media.
   tau=t ;
   m= size(X,2) ;
   for tau1=1:m
      mu1(tau1)=mean(X(:,tau1));
   end
   figure(20); plot(tau,mu1); axis tight;  grid on
   title('Valores medios del proceso')
   a = mean(mu1) ;
   text(1,5.5, ['Valor Medio de cada período: ' num2str(a)],'Color','red','FontSize',14)
   n= size(X,1) ;
   for tau1=1:n
      mu2(tau1)=mean(X(tau1,:));
   end
   figure(21); plot(mu2); axis tight;  title('Valores medios de cada realización')
   a = mean(mu2) ;
   text(1,5.5, ['Valor Medio: ' num2str(a)],'Color','red','FontSize',14)
end
%% generador1 Rampa
function [x] = generador1
   dt = 0.01 ;
   t1=0: dt: 10-dt ;
   a = 10;  b=0.5;  c = 6;
   x = [];
   for i=1:8
     Ruido =  c*rand(1, length(t1))  -c/2 ;
     A = a ; %A = a+  b*rand(1) ;
     x1= (rampa(t1)-rampa(t1-A) - A*escalon(t1-A)) + Ruido  ;
     x = [x x1];
   end
   % Retorno x ;
end
```

En la Fig. 117 se muestran lo resultados en Matlab®

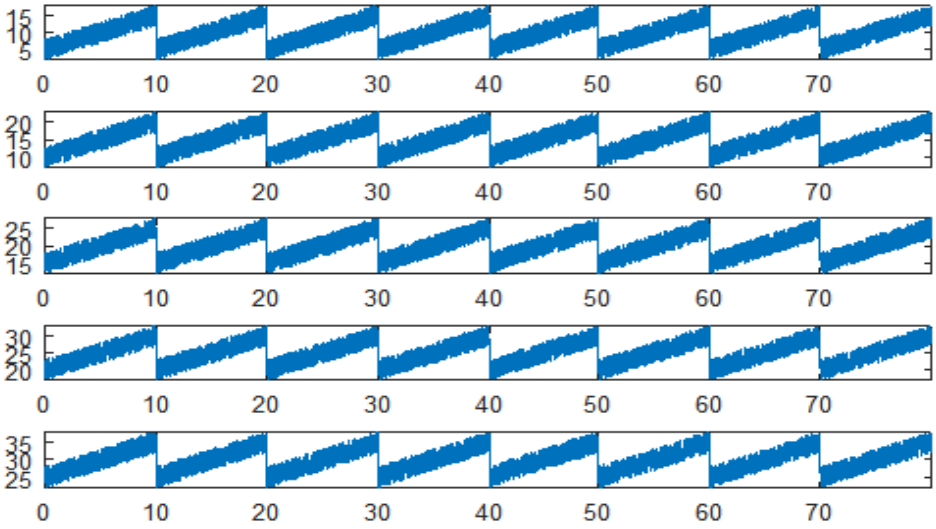


Fig. 117 – Generador Defectuoso: Aumenta el valor medio en cada realización

Ejercicio 2.3

Procesos estacionarios

Determinar si X_t es un proceso estacionario. Siendo:

$$X_t = \begin{cases} Y_t & ; \quad si\ t\ es\ impar \\ Y_t + 2 & ; \quad si\ t\ es\ par \end{cases}$$

Donde $\mathbf{Y_t}$ **es otro proceso «estacionario»** con $E\{Y_t\} = \mu$

Por ejemplo: Y_t pueden corresponder a las mediciones de tensión de un sensor analógico (temperatura, humedad, etc)

Resolución:

$\varphi_{xx}(t_1, t_2) = \varphi_{xx}(\tau) = E\{X_{t1}, X_{t2}\}$ con $\tau = t_1 - t_2$

es constante para un mismo τ, depende de τ

Pero la esperanza $E\{X_t\}$ no es constante, ya que:

$$E\{X_t\} = \begin{cases} \mu & ; \quad si\ t\ es\ impar \\ \mu + 2 & ; \quad si\ t\ es\ par \end{cases}$$

⇒ $\boldsymbol{X_t}$ ***¡No es un proceso estacionario !***

En cada realización varía su valor continuo

Ejercicio 2.4

Proceso estacionario senoidal

Determinar si $x(t)$ es un ***proceso estacionario respecto de la media***,

siendo: $\boldsymbol{x(t) = \cos(t + \phi)}$;

ϕ se distribuye uniformemente entre $[-\pi, \pi]$.

Es un generador con ***fase estocástica***

Resolución

Para que sea estacionario respecto de la media, el promedio del ensamble debe ser constante para todo t

$$E\{x(t)\} = \mu_x = cte$$

Cada realización está ligada a la variación continua de ϕ, con densidad de probabilidad uniforme $f(\phi)$, por lo que usamos integral en vez de sumatoria para calcular la esperanza.

Para cada t_i calculamos su esperanza:

$$E\{x(t_i)\} = \int_{\phi=-\infty}^{\infty} x(\phi).f(\phi).d\phi = \int_{\phi=-\pi}^{\pi} \cos(t_i + \phi).\frac{1}{2.\pi}.d\phi = 0 \;\; ; \forall t_i$$

Integral Da 0, coseno periódico en $2.\pi$

$\boxed{E\{x(t)\} = \mu_x = 0}$ Es estacionario respecto de la media

Ejercicio 2.5

Análisis de ergodicidad en Proceso estacionario

Determinar si $x(t)$ es un ***Ergódico respecto de la media***,

siendo: $\boldsymbol{x(t) = \cos(t + \phi)}$;

ϕ se distribuye uniformemente entre $[-\pi, \pi]$.

Es un generador con ***fase estocástica***

Resolución

Para que sea ergódico respecto de la media:

$$\overline{x(t)} = E\{x(t)\} = \mu_x$$

1) Valor medio - Análisis temporal de las muestras (ϕ=cte)

$$\overline{x_i(t)} = lim_{T\to\infty} \frac{1}{2.T} \int_{-T}^{T} x_i(t).dt = lim_{T\to\infty} \frac{1}{2.T} \int_{-T}^{T} \cos(t + \phi).dt = 0; \;\; \forall \phi$$

$$\overline{x_i(t)} = \overline{x(t)} = 0$$

2) Valor medio - Análisis del ensamble *($t = cte$)*

Cada realización depende de $\boldsymbol{\phi}$.

Calculado en punto anterior: $E\{x(t)\} = \mu_x = 0$

El proceso es ***Ergódico respecto de la media dado que:***

$$\boxed{\overline{x(t)} = E\{x(t)\} = \mu_x = 0}$$

Para que $x(t)$ *sea* ***Ergódico respecto de la autocorrelación***,

siendo: $\boldsymbol{x(t) = \cos(t + \phi)}$;

ϕ se distribuye uniformemente entre $[-\pi, \pi]$.

1) Autocorrelación - Análisis temporal de las muestras (ϕ=cte) temporal

$$R_{xx}(\tau) = lim_{T\to\infty} \frac{1}{2.T} \int_{-T}^{T} x_i(t).dt = lim_{T\to\infty} \frac{1}{2.T} \int_{-T}^{T} cos(t+\phi_i).cos((t+\tau)+\phi_i).dt$$

$$R_{xx}(\tau) = \frac{1}{2.T} \int_{-T}^{T} cos(t+\phi_i).cos(t+\tau+\phi_i).dt$$

Aplicando identidad trigonométrica y resolviendo la integral se llega a:

$$R_{xx}(\tau) = \frac{1}{2}\cos(\tau)$$

Análisis del ensamble t_1 *y* t_2 *constantes*

Cada realización depende de $\boldsymbol{\phi}$.

$$\varphi_{xx}(t_1,t_2) = E\{cos(t_1+\phi).cos(t_2+\phi)\}$$

Aplicando identidad trigonométrica

$$\varphi_{xx}(t_1,t_2) = E\{cos(t_1+t_2+2.\phi)+cos(t_1-t_2)\}$$

$$\varphi_{xx}(t_1,t_2) = \frac{1}{2}E\{cos(t_1+t_2+2.\phi)+cos(t_1-t_2)\}$$

$$\varphi_{xx}(t_1,t_2) = \frac{1}{2}.\frac{1}{2.\pi}\left\{\int_{-\pi}^{\pi} cos(2.\phi+(t_1+t_2)).d\phi + cos(t_1-t_2)\int_{-\pi}^{\pi} d\phi\right\}$$

Siendo t_1 *y* t_2 *constantes. Coseno varía en* $\boldsymbol{\phi}$: es periódico cada $2.\pi$

$$\varphi_{xx}(t_1,t_2) = \frac{1}{2}.cos(t_1-t_2) = \frac{1}{2}.cos(\tau)$$

Dado que es estacionario, solo depende de $\tau = t_2 - t_1$.

Resulta ergódico respecto de la FAC:

$$\varphi_{xx}(t_1,t_2) = R_{xx}(\tau) = \frac{1}{2}\cos(\tau)$$

Ejercicios de Procesos estacionarios y modelos

Breve repaso

Proceso ARMA

$u[n] + a_1^*.\ u[n-1] + \ \ldots\ldots + a_M^*.u[n-M] = v[n] + b_1^*.\ v[n-1] + \ \ldots\ldots + b_k^*.\ v[n-k]$

Siendo $v[n]$ ruido blanco:

$E\{v[n]\} = 0 \quad ;\ \forall n$

$$E\{v[n]v^*[n]\} = \begin{cases} \sigma_v^2 \ ; & k = n \\ 0 \ ; & k \neq n \end{cases}$$

Proceso AR:

$u[n] + a_1^*.\ u[n-1] + \ \ldots\ldots + a_M^*.u[n-M] = v[n]$

$u[n] = \ w_1^*.\ u[n-1] + w_2^*.\ u[n-2] + \ \ldots\ldots + w_M^*.u[n-M] + \ v[n]$

$\sum_{k=0}^{M} a_k^*.u[n-k] = v[n]$

Proceso MA

$u[n] = \ v[n] + b_1^*.\ v[n-1] + \ \ldots\ldots + b_k^*.\ v[n-K]$

$u[n] = \sum_{i=0}^{K} b_i^*.v[n-i]$

Ejercicio 2.6

Análisis de Procesos AR, MA y ARMA

Dada la siguiente ecuación en diferencias correspondiente a un Proceso:

$$u[n] + a_1.\ u[n-1] + a_2.\ u[n-2] = v[n]$$

Siendo $a_1 = -0{,}1$ y $a_2 = -0{,}7$

Se pide:

a) Indicar tipo de proceso (AR, MA o ARMA)

b) Indicar si es estable. En caso de ser estable indicar el comportamiento de la función de autocorrelación normalizada $r[m]/r[0]$.

Solo se pide descripción breve (1 renglón)

Ayuda: ver ejercicios anteriores

Ejercicio 2.7

Análisis de Procesos AR, MA y ARMA

Dado el proceso:

$Y_t = 3 + V_t + 0{,}8.V_{t-1} - 0{,}5.V_{t-2} \quad ;\ \sigma_V^2 = 1$

Se pide:

a) Calcular Valor Esperado del proceso, su varianza y sus autocovarianzas.

b) Calcular la autocorrelación

c) Graficar el correlograma

Resolución

a) $Y_t = 3 + V_t + 0{,}8.V_{t-1} - 0{,}5.V_{t-2} \quad ;\ \sigma_V^2 = 1$

Es un proceso MA, ecuación:

$u[n] = \ v[n] + b_1^*.v[n-1] + \ \ldots\ldots + b_k^*.v[n-K]$

$$E\{v[n]v^*[n]\} = \begin{cases} \sigma_v^2 \ ; & k = n \\ 0 \ \ ; & k \neq n \end{cases}$$

Utilizaremos esta nomenclatura:

$Y_t = 3 + V_t + 0{,}8.V_{t-1} - 0{,}5.V_{t-2}$; V_t Es el ruido blanco, $\sigma_V^2 = 1$; $E\{V[n]\} = 0$; $\forall n$

$$E\{V[n]V^*[n]\} = \begin{cases} \sigma_V^2 \ ; & k = n \\ 0 \ ; & k \neq n \end{cases}$$

Calculamos el valor Esperado

$\mu = E\{Y_t\} = E\{3 + V_t + 0{,}8.V_{t-1} - 0{,}5.V_{t-2}\} = E\{3\} + E\{V_t\} + E\{0{,}8.V_{t-1}\} + E\{-0{,}5.V_{t-2}\} = 3$

Calculamos la varianza:

$\gamma_0 = E\{(Y_t - 3)^2\} = E\{(V_t + 0{,}8.V_{t-1} - 0{,}5.V_{t-2})^2\}$

$\gamma_0 = E\left\{\left((V_t + 0{,}8.V_{t-1}) - 0{,}5.V_{t-2}\right)^2\right\}$

$\gamma_0 = E\{(V_t + 0{,}8.V_{t-1})^2\} + E\{2.(V_t + 0{,}8.V_{t-1}).(-0{,}5.V_{t-2})\} + E\{(-0{,}5.V_{t-2})^2\}$

Analizamos por separado (paso opcional):

$E\{(V_t + 0{,}8.V_{t-1})^2\} = E\{V^2\} + E\{2.V_t.0{,}8.V_{t-1}\} + E\{(0{,}8.V_{t-1})^2\} = \sigma_V^2 + 0 + (0{,}8)^2.\sigma_V^2$

$E\{2.(V_t + 0{,}8.V_{t-1}).(-0{,}5.V_{t-2})\} = 0$ Hacemos distributiva, no están correlacionados

$E\{(-0{,}5.V_{t-2})^2\} = (0{,}5)^2.\sigma_V^2$

$\gamma_0 = \sigma_V^2 + 0 + (0{,}8)^2.\sigma_V^2 + (0{,}5)^2.\sigma_V^2 = \sigma_V^2.(1 + (0{,}8)^2 + (0{,}5)^2)$

$\gamma_0 = 1{,}89.\sigma_V^2 = 1{,}89$

Calculamos las autocovarianzas:

$\gamma_1 = E\{(Y_t - 3).(Y_{t-1} - 3)\} = E\{(V_t + 0{,}8.V_{t-1} - 0{,}5.V_{t-2}).(V_{t-1} + 0{,}8.V_{t-2} - 0{,}5.V_{t-3})\}$

$\gamma_1 = E\{(V_t + 0{,}8.V_{t-1} - 0{,}5.V_{t-2}).(V_{t-1} + 0{,}8.V_{t-2} - 0{,}5.V_{t-3})\}$

Realizamos distributiva y tenemos en cuenta que:

$E\{V_t.V_{t-1}\} = 0$; $E\{V_t.V_{t-2}\} = 0$; $E\{V_t.V_{t-3}\} = 0$; $E\{V_{t-1}.V_{t-2}\} = 0$

$\gamma_1 = E\{(0{,}8.V_{t-1}.V_{t-1})\} + E\{(-0{,}5x0{,}8.V_{t-2}.V_{t-2})\}$

$\gamma_1 = 0{,}8.E\{V_{t-1}.V_{t-1}\} - 0{,}4.E\{(V_{t-2}.V_{t-2})\} = 0{,}8.\sigma_V^2 - 0{,}4.\sigma_V^2 = 0{,}4.\sigma_V^2$

$\gamma_1 = 0{,}4$

$\gamma_2 = E\{(Y_t - 3).(Y_{t-2} - 3)\} = E\{(V_t + 0{,}8.V_{t-1} - 0{,}5.V_{t-2}).(V_{t-2} + 0{,}8.V_{t-3} - 0{,}5.V_{t-4})\}$

$\gamma_2 = E\{-0{,}5.V_{t-2}.V_{t-2}\} = -0{,}5.\sigma_V^2$; $\gamma_2 = -0{,}5$

$\gamma_j = 0 \quad \forall j > 2$

b) Función de autocorrelación:

$\rho_1 = \frac{\gamma_1}{\gamma_0} = \frac{0{,}4}{1{,}89} = 0{,}2116$

$\rho_2 = \frac{\gamma_2}{\gamma_0} = \frac{-0{,}5}{1{,}89} = -0{,}2646$

$\rho_j = \frac{\gamma_j}{\gamma_0} = 0 \quad \forall j > 2$

c) Graficamos el Correlograma, que es la representación de las funciones de correlación ρ_j

Solo usamos: $V_t + 0{,}8.V_{t-1} - 0{,}5.V_{t-2}$

v=[1 0.8 -0.5];

```
[Rvv, tau]=xcorr(v)
stem(tau, Rvv, 'linewidth',3)
figure ; stem(tau, Rvv/Rvv(3), 'linewidth',3)
grid on ;
set(gca,'FontSize',12,'FontWeight','bold');
```

En la Fig. 118 se muestran los resultados obtenidos

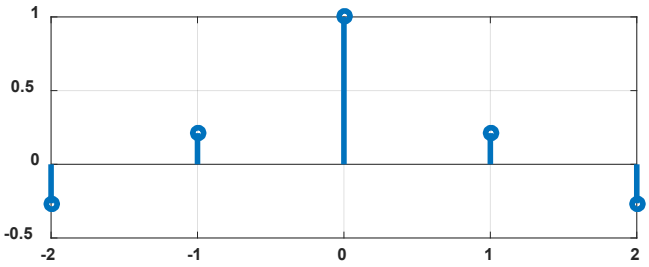


Fig. 118 – Gráfico de correlograma normalizado

Ejercicio 2.8

Identificación de Modelo autoregresivo (AR)

Mediante Matlab® identificar el modelo AR de la Fig. 119, con parámetros $[1 \; -1{,}75 \;\; 0{,}9]$

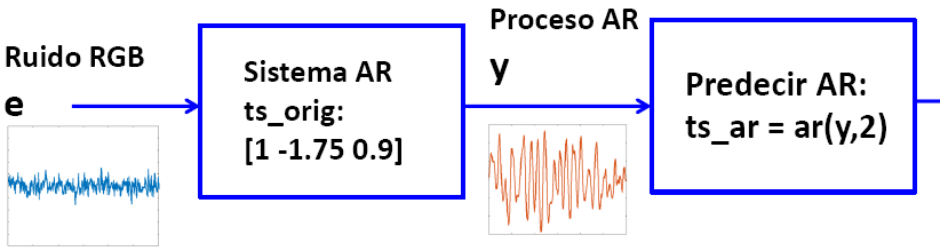


Fig. 119 – Modelo AR para identificar

Resolución

```
%Generamos Modelo y simulamos la salida
% Generamos serie temporal, creando y simulando un proceso autoregresivo (AR)
% ts_orig desde yk=a1.yk-1 +a2 yk-2 +ek  % ek random Gaussian noise.
% El ruido es la entrada no medida del sistema
   %random number generator seed para que se repita.
   clc; close all; clear all
   ts_orig = idpoly([1 -1.75 0.9]);  % idpoly Modelo 151olynomial
   rng('default')
   e = idinput(300,'rgs'); % genera señal Random Gaussian signal
```

```
y_obs = sim(ts_orig,e); % simula la salida y_obs
%convertimos y_obs a objeto iddata object y con Ts=1
y = iddata(y_obs);
%  Graficamos e y salida
plot(e, 'linewidth',2)  ;hold on
plot(y_obs, 'linewidth',2)
title('Ruido de entrada y salida original del sistema')
legend('RGS Noise','Model Output')
% Opcional, comparamos los espectros
 % etfe y spa métodos no paramétricos para análisis frecuencial
% Comparamos la densidad de potencia estimada con etfe, con spa y con el modelo original.
Ts_etfe = etfe(y);
ts_spa = spa(y);
figure; spectrum(ts_etfe,ts_spa,ts_orig);
legend('ts_{etfe}','ts_{spa}','ts_{orig}')
% Estimamos modelo paramétrico usando estructura AR de orden 2
% Y comparamos los espectros.
Ts_ar = ar(y,2);
figure; spectrum(ts_spa,ts_ar,ts_orig);
legend('ts_{spa}', 'ts_{ar}', 'ts_{orig}')
% Predecimos y comparamos las salidas del modelo
% Comparamos la precisión de la predicción en tres pasos, o porcentaje de ajuste, para el modelo original %  el modelo AR,  utilizando la función compare. Aquí, compare calcula la respuesta predicha de los modelos % ts_orig y ts_ar con los datos de salida del modelo original y,
% asumiendo que la entrada ek no medida es cero. El cuarto argumento, 3, es el número de pasos a
% predecir.  Función compare: Compara la salida identificada del modelo y la salida medida
% y es la salida del modelo, ts_orig es el modelo, ts_ar
figure ;  compare(y,ts_orig,ts_ar,3);
```

Se muestran los resultados en Fig. 120 y Fig. 121

Parámetros AR del modelo

```
1 -1.75 0.9

Parámetros estimados:

1.0000   -1.7621    0.9086
```

Fig. 120 – Parámetros originales y parámetros estimados

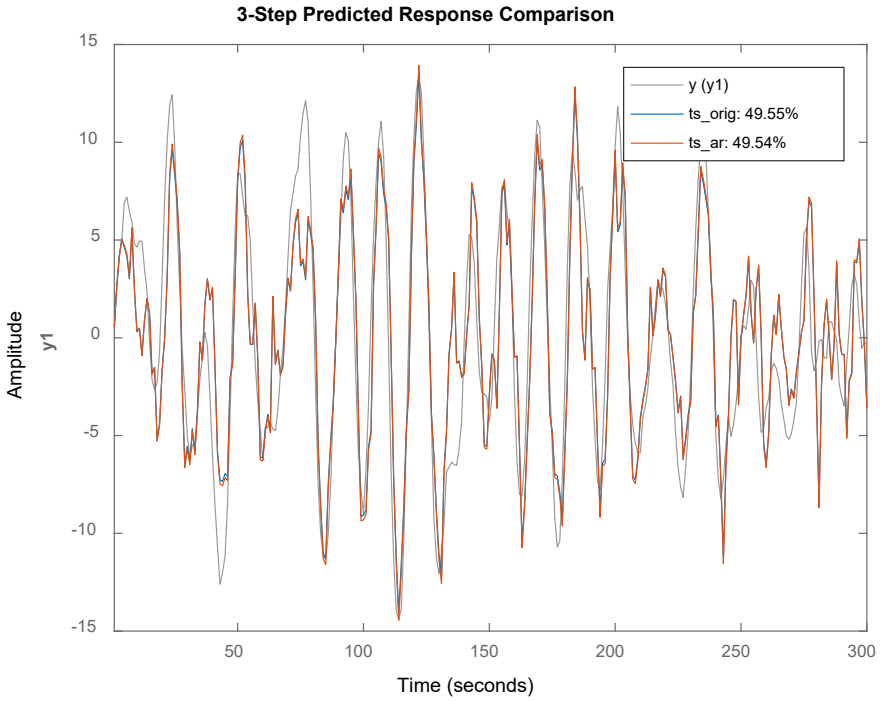


Fig. 121 – Gráfico de señal original y señal predecida

Ejercicio 2.9

Estimación del orden de un Modelo autoregresivo (AR)

Estimar el orden del Modelo AR de una serie temporal mediante la decadencia de los coeficientes de reflexión

Resolución

```
%% Estimar el orden de un modelo a través la decadencia de sus coeficientes de reflexión
% Generar un vector de coeficientes de un proceso AR(2), mediante el filtrado
% de 1024 muestras de ruido blanco.
% Utilizamos generador de números aleatorios por defecto
rng default
v=0.2*randn(1024,1) ;
y = filter(1, [1 -0.75 0.5] , v);
% Utilizamos Yule-Walker y ajustamos el proceso a un modelo AR(10)
% Obtenemos la salida y el trazado en los coeficientes de reflexión.
[ar_coefic , NoiseVariance , reflect_coefic] = aryule(y , 10);
figure; stem(reflect_coefic)
axis([ -0.05,  10.5,  -1,  1]) ;
title('Coeficientes de Reflexión (desplazamiento)')
```

Se obtiene el correlograma de la Fig. 122. Viendo el gráfico, utilizaremos modelo AR de orden 2

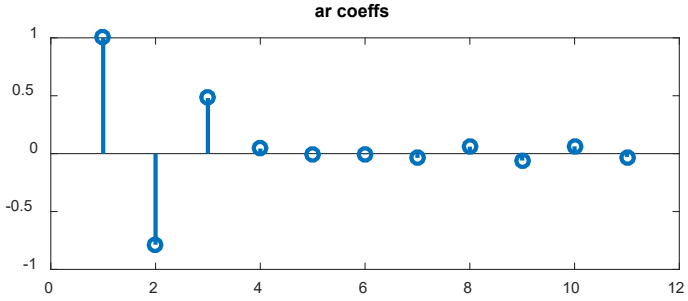


Fig. 122 – Gráfico de correlograma

Descarga de los códigos de los ejercicios

III - Análisis espectral

Densidad espectral de potencia

La función de autocorrelación es una descripción de las estadísticas de segundo orden del dominio temporal de un proceso estocástico. La contraparte en el dominio frecuencial de este parámetro estadístico es la (DEP): densidad espectral de potencia, también denominada espectro de potencia o, simplemente, espectro. De hecho, la DEP de un proceso estocástico está firmemente establecida como la descripción más útil de las series temporales que se encuentran comúnmente en la ingeniería y las ciencias físicas.

Para definir la densidad espectral de potencia, considere un proceso estocástico estacionario de tiempo discreto de sentido amplio cuya media es cero y cuya función de autocorrelación se describe con $r(l)$ para un desplazamiento $l = 0, \pm 1, \pm 2, \ldots..$ Sea la serie temporal infinitamente larga $x[n]$, con $n = 0, \pm 1, \pm 2, \ldots.$ describe una sola realización del proceso. Inicialmente, enfocamos la atención en una porción de *"ventana"* de esta serie temporal escribiendo:

$$x_N[n] = \begin{cases} x[n], & \text{con } n = 0, \pm 1, \ldots \pm N \\ 0, & |n| > N \end{cases} \tag{82}$$

Por definición, la transformada Fourier en tiempo discreto para la serie temporal en ventana: $x_N[n]$ viene dada mediante la siguiente ecuación:

$$X_N(w) = \sum_{n=-N}^{N} x_N[n].e^{-j.w.n} \tag{83}$$

La longitud de la ventana temporal, es decir 2N + 1, puede tender a infinito.

Debido a la función exponencial $e^{-j.w.n}$, la transformada resulta periódica en el intervalo de w correspondiente a $[-\pi, \pi]$, siendo w la frecuencia angular. En general, $X_N(w)$ tiene un valor complejo; su conjugado complejo se indica con asterisco y utilizamos variable k

$$X^*{}_N(w) = \sum_{k=-N}^{N} x_N^*[k].e^{j.w.k} \tag{84}$$

Si multiplicamos ambas funciones obtenemos: $|X_N(w)|^2 = X_N(w).X^*{}_N(w)$

$$|X_N(w)|^2 = \sum_{n=-N}^{N} \sum_{k=-N}^{N} x_N[n].x_N^*[k].e^{-j.w(n-k)} \tag{85}$$

Cada realización de $X_N(w)$ genera un resultado de $|X_N(w)|^2$

Aplicamos operador Esperanza en ambos miembros obtenemos y cambiamos el orden en la sumatoria; colocamos la esperanza dentro de la sumatoria. Esto se puede realizar debido a que tenemos operaciones lineales

$$E\{|X_N(w)|^2\} = \sum_{n=-N}^{N} \sum_{k=-N}^{N} E\{x_N[n].x_N^*[k]\}.e^{-j.w(n-k)} \tag{86}$$

Para el proceso estocástico de tiempo discreto estacionario en sentido amplio en discusión, la función de autocorrelación de $x_N[n]$ para el desplazamiento $n-k$ es:

$$r_N[n-k] = E\{x_N[n].x_N^*[k]\} \tag{87}$$

Basándonos en la definición de ventana de $x_N[n]$, reescribimos la ecuación anterior:

$$r_N[n-k] = \begin{cases} E\{x_N[n].x_N^*[k]\} = r[n-k] & para \quad -N \le (n,k) \le N \\ 0 & para\ otro\ caso \end{cases} \tag{88}$$

Teniendo en cuenta las ecuaciones (86) y (88), obtenemos:

$$E\{|X_N(w)|^2\} = \sum_{n=-N}^{N} \sum_{k=-N}^{N} r[n-k].e^{-j.w(n-k)} \tag{89}$$

Tomando $l = n-k$, y recordando que la convolución de 2 pulsos (ventana mencionada anteriormente) es un triángulo, la ecuación anterior se escribe de la siguiente manera:

$$\frac{1}{N} E\{|X_N(w)|^2\} = \sum_{l=-N}^{N} \left(1 - \frac{|l|}{N}\right).r[l].e^{-j.w.l} \tag{90}$$

La ecuación anterior puede interpretarse como la transformada de Fourier en tiempo discreto del producto de dos funciones temporales:

a) La función de autocorrelación $r[l]$ para el desplazamiento l,
b) Una ventana triangular conocida como ventana de Bartlett $w_B[l]$

$$w_B[l] = \begin{cases} 1 - \dfrac{|l|}{N}, & |l| \le N \\ 0, & |l| \ge N \end{cases} \tag{91}$$

Si N tiende a infinito, $w_B[l]$ se acerca a la unidad para todo l. En consecuencia, podemos escribir

$$\lim_{N\to\infty} \frac{1}{N} E\{|X_N(w)|^2\} = \sum_{l=-N}^{N} r[l].e^{-j.w.l} \tag{92}$$

Donde $r[l]$ es la función de autocorrelación de la serie temporal original $x[n]$. La ecuación anterior se cumple bajo la siguiente condición: $\lim_{N\to\infty} \frac{1}{2.N+1} \sum_{l=-N+1}^{N-1} \left(|l|.r[l].e^{-j.w.l}\right) = 0$

La ecuación (92) nos define $S(w)$:

$$S(w) = \lim_{N\to\infty} \frac{1}{N} E\{|X_N(w)|^2\} \tag{93}$$

donde $|X_N(w)|^2/N$ se denomina periodograma de la serie temporal de ventana $x_N[n]$. Tener en cuenta que no se puede cambiar el orden entre la esperanza y el límite. Tenga en cuenta también que el periodograma converge a $S(w)$ solo en el valor medio, no en el cuadrado medio o cualquier otra forma.

Cuando el límite existe, el valor de $S(w)$ tiene la siguiente interpretación (Priestley, 1981):

$S(w).dw$ = promedio de la contribución a la potencia total de los componentes de un proceso estocástico estacionario de sentido amplio con frecuencias angulares ubicadas entre w y $w + dw$; el promedio se toma sobre todas las posibles realizaciones del proceso.

En consecuencia, el valor de $S(w)$ es la "densidad espectral de potencia esperada", que resulta ser la *"densidad espectral de potencia del proceso"*. Teniendo en cuenta la ecuación (92) obtenemos la ecuación buscada.

Densidad espectral de potencia $S(w)$:

$$S(w) = \lim_{N\to\infty} \frac{1}{N} E\{|X_N(w)|^2\} = \sum_{l=-\infty}^{\infty} r[l].e^{-j.w.l} \qquad (94)$$

En resumen, el término que contiene límite nos brinda una definición básica de la densidad espectral de potencia para un proceso estocástico estacionario de sentido amplio, y la sumatoria (parte derecha) define la relación matemática entre la función de autocorrelación y la DEP: densidad espectral de potencia de dicho proceso.

Propiedades de la Densidad Espectral de Potencia

Propiedad 1: *La densidad espectral de potencia de un proceso estocástico estacionario de sentido amplio es la Transformada Fourier de la función de autocorrelación y viceversa.*

Considere un proceso estocástico estacionario de sentido amplio representado por la serie temporal $x[n]$, suponiendo que tiene una longitud infinita. Sea $r[l]$ la función de autocorrelación de tal proceso para el desplazamiento l, y sea $S(w)$ la densidad espectral de potencia del proceso.

Este par fundamental de ecuaciones constituye las relaciones de ***Einstein-Wiener-Khintchine***.

Según la propiedad 1, estas dos funciones están relacionadas por el par de relaciones:

$$S(w) = \sum_{l=-\infty}^{\infty} r[l].e^{-j.w.l} \quad ; \quad -\pi < w \leq \pi$$

$$r[l] = \frac{1}{2.\pi} \int_{-\pi}^{\pi} S(w).e^{j.w.l}.dw \quad ; \quad l = 0, \pm 1, \pm 2, \ldots..$$

La primera ecuación se obtuvo anteriormente. La segunda ecuación se obtiene directamente aplicando la transformada inversa de Fourier en tiempo discreto. Esta propiedad se puede expresar de la siguiente forma:

$$S(w) = F\{r[l]\}$$
$$r[l] = F^{-1}\{S(w)\}$$
$$r[l] \longleftrightarrow S(w)$$

Propiedad 2. Las frecuencias de la DEP (densidad espectral de potencia) de $S(w)$ se analizan en el intervalo $-\pi < w \leq \pi$.

$S(w)$ es periódica, fuera del intervalo mencionado se utiliza esta relación

$$\boxed{S(w) = S(w + 2.k.\pi) \;\; ; \;\; k = 0, \pm 1, \pm 2, \ldots.}$$

Propiedad 3. *La DEP (densidad espectral de potencia) de un proceso estocástico en tiempo discreto estacionario es real.* Partiendo de la definición anterior de $S(w)$, reescribimos:

$$S(w) = \sum_{l=-\infty}^{\infty} r[l].e^{-j.w.l} = r[0] + \sum_{k=1}^{\infty} r[k].e^{-j.w.k} + \sum_{k=-\infty}^{-1} r[k].e^{-j.w.k}$$

Reemplazando k con – k, modificando sumatoria y siendo $r[-k] = r^*[k]$, se obtiene:

$$S(w) = r[0] + \sum_{k=1}^{\infty} \left(r[k].e^{-j.w.k} + r^*[k].e^{+j.w.k}\right)$$

$$S(w) = r[0] + 2.\sum_{k=1}^{\infty} Re\left\{r[k].e^{-j.w.k}\right\}$$

En la última ecuación se muestra que $S(w)$ es real. Re indica parte real

Propiedad 4. La densidad espectral de potencia de un proceso estocástico estacionario en tiempo discreto con valores reales siempre es par (es decir, simétrico).

Para un proceso estocástico con valores reales, encontramos que $\boxed{S(w) = S(-w)}$, lo que indica que $S(w)$ es una función par de w; es decir, es simétrico con respecto al origen.

En cambio, si el proceso tiene un valor complejo, su DEP no es necesariamente par.

SI el proceso tiene un valor complejo, siendo $r[-k] = r^*[k]$, en cuyo caso encontramos que $S(-w) \neq S(w)$ no es una función par de w.

Propiedad 5. El valor cuadrático medio para un proceso estocástico de tiempo discreto y estacionario es igual, excepto por el factor de escala $1/(2.\pi)$, al área que está bajo la curva de la densidad espectral de potencia para $-\pi < w \leq \pi$

Partiendo de la definición anterior de $r[l]$,

$r[l] = \frac{1}{2.\pi}\int_{-\pi}^{\pi} S(w).e^{j.w.l}.dw \;\; ; \;\; l = 0, \pm 1, \pm 2, \ldots.$

Para $l = 0$ resulta y dado que $r[0]$ es igual al valor cuadrático medio del proceso se describe matemáticamente la propiedad 5:

$$\boxed{r[0] = \frac{1}{2.\pi}\int_{-\pi}^{\pi} S(w).dw}$$

El valor cuadrático medio del proceso es igual a la potencia esperada del proceso desarrollado a través de una resistencia de carga de 1 Ω. Sobre esta base, los términos "potencia esperada" y "valor cuadrático medio" se utilizan indistintamente en lo que sigue.

Propiedad 6. La densidad espectral de potencia para un proceso estocástico de tiempo discreto estacionario no es negativa.

Es decir, $\boxed{S(w) \geq 0\,; \quad \text{para } \forall w}$

Partiendo de la definición anterior: $S(w) = \lim_{N\to\infty} \frac{1}{N} E\{|X_N(w)|^2\}$

En primer lugar, observamos que $|X_N(w)|^2$ representa la magnitud al cuadrado de la transformada de Fourier en tiempo discreto de una porción de ventana de la serie temporal $x_N[n]$, no es negativa para todo w. Entonces la esperanza $E\{|X_N(w)|^2\}$ tampoco es negativo para todo w. Por lo tanto, usando la definición básica de $S(w)$ en términos de $X_N(w)$ se demuestra la propiedad descripta.

Transmisión de un proceso estacionario a través de un filtro lineal

Se analiza un filtro de tiempo discreto que lineal, invariante en el tiempo y estable. Dado un filtro caracterizado por la función de transferencia discreta $H(z)$, definida como la división entre la transformada z de la salida del filtro y la transformada z de la entrada del filtro. Suponga que alimentamos el filtro con un proceso estocástico de tiempo discreto estacionario con densidad espectral de potencia $S(w)$, como se muestra en la Fig. 123. Sea $S_0(w)$ la DEP (densidad espectral de potencia) de la salida del filtro. Entonces podemos escribir

$$\boxed{S_0(w) = \left|H(e^{j.w})\right|^2 . S(w)} \tag{95}$$

La respuesta en frecuencia de un filtro es $H(e^{j.w})$, y es igual a la función de transferencia discreta $H(z)$ evaluada dentro del círculo unitario en el plano z. La característica importante de este resultado es que el valor de la densidad espectral de salida a la frecuencia angular w depende puramente de la respuesta de amplitud al cuadrado del filtro y de la densidad espectral de la potencia de entrada a la misma frecuencia angular w. La ecuación anterior es una relación fundamental en la teoría de procesos estocásticos. Para obtener la ecuación, podemos proceder de la siguiente manera: Sea $y[n]$ la salida del filtro de la Fig. 123, producida en respuesta a una entrada $x[n]$ aplicada al filtro. Suponiendo que $x[n]$ representa un proceso estocástico de tiempo discreto estacionario de sentido amplio, encontramos que $y[n]$ también representa un proceso estocástico de tiempo discreto estacionario de sentido amplio modificado por la operación de filtrado. Por lo tanto, dada la función de autocorrelación de la entrada del filtro $x[n]$, escrita como:

$$r_x(l) = E\{x[n].x[n-l]\} \tag{96}$$

Podemos expresar la función de autocorrelación de la salida del filtro $y[n]$ de la siguiente manera:

$$r_y(l) = E\{y[n].y[n-l]\} \tag{97}$$

La salida $y[n]$ está relacionado con $x[n]$ por la suma de convolución:

$$y[n] = \sum_{i=-\infty}^{\infty} h[i].x[n-i] \tag{98}$$

En forma similar, tomando $n - l$ en vez de n, podemos escribir

$$y^*[n-l] = \sum_{k=-\infty}^{\infty} h^*[k].x^*[n-l-k] \tag{99}$$

Sustituyendo las ecuaciones (98) y (99) en la ecuación (97)

$$r_y(l) = E\left\{\sum_{i=-\infty}^{\infty} h[i].x[n-i].\left(\sum_{k=-\infty}^{\infty} h^*[k].x^*[n-l-k]\right)\right\}$$

Intercambiando los órdenes de Esperanza y sumatoria:

$$r_y(l) = \sum_{i=-\infty}^{\infty}\sum_{k=-\infty}^{\infty} h[i].h^*[k].E\{x[n-i].x^*[n-l-k]\}$$

Si analizamos las diferencias: $(n-i)-(n-l-k) = -i+l+k = k-i+l$

$$r_x(k-i+l) = E\{x[n-i].x^*[n-l-k]\}$$

Encontramos que las funciones de autocorrelación $r_y(l)$ y $r_x(l)$, para el desplazamiento l, están relacionadas por

$$r_y(l) = \sum_{i=-\infty}^{\infty}\sum_{k=-\infty}^{\infty} h[i].h^*[k].r_x(k-i+l) \tag{100}$$

Finalmente, tomando las transformadas de Fourier de tiempo discreto en ambos lados de la última ecuación, y usando la propiedad 1. Y sabiendo que la función de transferencia de un filtro lineal es igual a la transformada de Fourier de su respuesta impulsional: $H(e^{j.w}) = F\{h[n]\}$, obtenemos el resultado descrito de la ecuación (95):

$$S_0(w) = \left|H(e^{j.w})\right|^2.S(w)$$

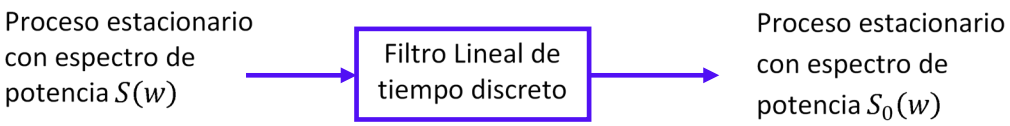


Fig. 123 - Transmisión de proceso estacionario a través de un filtro lineal de tiempo discreto.

Analizador de espectro de energía

Analizamos un filtro lineal de tiempo discreto diseñado para tener una respuesta en frecuencia pasa banda. Es decir, la respuesta de amplitud del filtro está definida por:

$$\left|H(e^{j.w})\right| = \begin{cases} 1, & |w - w_c| \le \Delta w \\ 0, & resto\ del\ intervalo \quad -\pi < w \le \pi \end{cases} \tag{101}$$

Esta respuesta de amplitud se muestra en la Fig. 124. Suponemos que el ancho de banda de frecuencia angular del filtro $(2.\Delta w)$, es lo suficientemente pequeño como para que el espectro dentro de este

ancho de banda sea esencialmente constante. Luego, usando la ecuación $S_0(w) = \left|H(e^{j.w})\right|^2.S(w)$, podemos escribir:

$$S_0(w) = \begin{cases} S(w_c), & |w - w_c| \leq \Delta w \\ 0, & resto\ del\ intervalo \quad -\pi < w \leq \pi \end{cases} \tag{102}$$

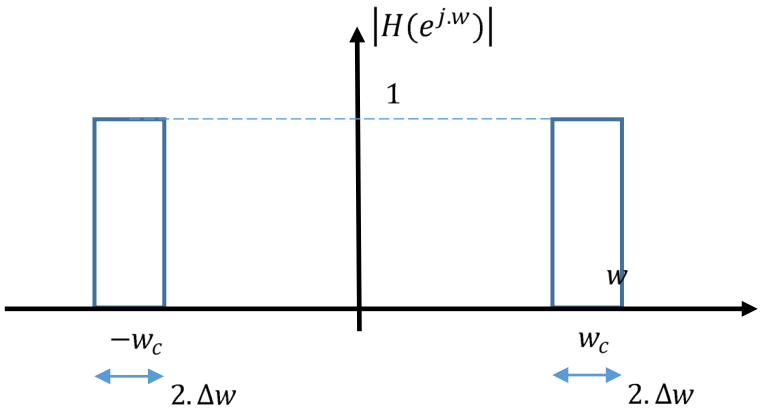


Fig. 124 – Filtro Pasa Banda Ideal

A continuación, utilizando la propiedad 5 de la DEP: $r[0] = \frac{1}{2.\pi}\int_{-\pi}^{\pi} S(w).dw$

Reemplazando la salida $S_0(w)$

$$P_0 = \frac{1}{2.\pi}\int_{-\pi}^{\pi} S_0(w).dw$$

Y utilizando la propiedad 4: : $S(w) = S(-w)$ para procesos temporales con datos reales (con parte imaginaria nula) ; se puede expresar el valor cuadrático medio de la salida del filtro correspondiente a una entrada estocástica de valor real como:

$P_0 = \frac{1}{2.\pi}\int_{-\pi}^{\pi} S_0(w).dw = \frac{2.\Delta w}{2.\pi}.S(w_c) + \frac{2.\Delta w}{2.\pi}.S(-w_c) = 2.\frac{\Delta w}{\pi}.S(w_c)$ (para datos reales)

Se obtiene:

$$S(w_c) = \frac{\pi.P_0}{2.\Delta w} \tag{103}$$

Donde $\Delta w/\pi$ es la fracción del intervalo de Nyquist que corresponde a la correspondiente banda de paso del filtro. La ecuación anterior establece que el valor de la densidad espectral de potencia de la entrada del filtro $x[n]$, medida en la frecuencia central w_c del filtro ($S(w_c)$), es equivalente al valor cuadrático medio P_0 de la salida del filtro, escalado por un factor constante. Por tanto, podemos utilizar la ecuación anterior como base matemática para construir un analizador de espectro de potencia, como se muestra en la Fig. 125. Idealmente, el filtro pasa banda de tiempo discreto empleado aquí debería satisfacer dos requisitos:

Debe tener un ancho de banda fijo y una frecuencia central ajustable. Claramente, en un diseño de filtro práctico, solo podemos aproximarnos a estos dos requisitos ideales. Además, se debe tener en cuenta que la lectura del medidor de potencia promedio en el extremo de salida de la figura aproxima (para un tiempo promedio finito) la potencia esperada de un proceso ergódico $y[n]$.

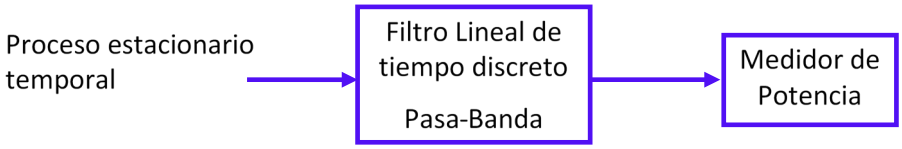


Fig. 125 – Analizador de espectros de Potencia

Ejemplo con Ruido blanco

Un proceso estocástico de media cero es blanco si su DEP (densidad espectral de potencia) $S(w)$ resulta constante para todas las frecuencias:

$$S(w) = \sigma^2 \quad ; \quad para \ -\pi < w \leq \pi$$

donde σ^2 es la varianza de una muestra tomada del proceso. Suponga que este proceso pasa a través de un filtro de pasa banda de tiempo discreto caracterizado como en la Fig. 124. Luego, de la ecuación (103) despejamos P_0, encontramos que el valor cuadrático medio a la salida del filtro es

$P_0 = \frac{S(w_c).2.\Delta w}{\pi}$; reemplazando obtenemos: $P_0 = \frac{\sigma^2.2.\Delta w}{\pi}$

El ruido blanco tiene la propiedad de que dos de sus muestras no están correlacionadas, como lo muestra la función de autocorrelación

$$r(\tau) = \sigma^2.\delta_{\tau,0^+}$$

Siendo $\delta_{\tau,0}$ es la delta de Kronecker:

$$\delta_{\tau,0} = \begin{cases} 1, & \tau = 0 \\ 0, & otro\ caso \end{cases}$$

También se puede expresar con esta nomenclatura:

$$\boxed{r(\tau) = \sigma^2.\delta(\tau)}$$

Si el ruido blanco es gaussiano, entonces dos muestras cualesquiera del proceso son estadísticamente independientes. En cierto sentido, el ruido blanco gaussiano representa la aleatoriedad más fuerte.

Estimación del espectro de potencia

En la práctica resulta importante estimar la DEP (densidad espectral de potencia) en un proceso estacionario de sentido amplio. Desafortunadamente, este problema se complica por el hecho de que existe una desconcertante variedad de procedimientos de estimación de los espectros de potencia, y cada procedimiento supuestamente tiene o muestra alguna propiedad óptima. La situación empeora por el hecho de que, a menos que se tenga cuidado en la selección del método correcto, podemos terminar con conclusiones engañosas. En la literatura se pueden identificar dos familias filosóficamente distintas de métodos de estimación del espectro de potencia: los métodos paramétricos y los métodos no paramétricos. Las ideas básicas detrás de estos métodos se discuten a continuación.

Métodos paramétricos

En los métodos paramétricos para la estimación del espectro, comenzamos postulando un modelo estocástico para la situación en cuestión. Dependiendo del modelo específico adoptado, podemos identificar tres enfoques paramétricos diferentes para la estimación del espectro:

1) Procedimientos de identificación de modelos.
   En esta clase de métodos paramétricos, se supone una función racional o un polinomio en $e^{-j.w}$ para la función de transferencia del modelo, y se utiliza una fuente de ruido blanco como entrada al modelo, como se muestra en la Fig. 126 El espectro de potencia de la salida del modelo resultante proporciona la estimación de espectro deseada. Dependiendo de la aplicación de interés, podemos adoptar uno de los siguientes modelos (Kay y Marple, 1981; Marple, 1987; Kay, 1988):

   (i) Un modelo autorregresivo (AR) con una función de transferencia de todos polos.

   (ii) Un modelo de media móvil (MA) con una función de transferencia de todos ceros.

   (iii) Un modelo de media móvil autorregresiva (ARMA) con una función de transferencia polos-ceros.

   Los espectros de potencia resultantes medidos en las salidas de estos modelos se denominan espectros AR, MA y ARMA, respectivamente. Con referencia a la relación entrada-salida de la ecuación (95): $S_0(w) = \left|H(e^{j.w})\right|^2.S(w)$, Se establece el espectro de potencia de la entrada $S(w) = \sigma^2$. Luego encontramos que el espectro de potencia de salida del modelo $S_0(w) = \left|H(e^{j.w})\right|^2.\sigma^2$. Por tanto, el problema se convierte en estimar los parámetros del modelo [es decir, parametrizar la función de transferencia $H(e^{j.w})$ de manera que el proceso producido en la salida del modelo proporcione una representación aceptable (en algún sentido estadístico) del proceso estocástico en estudio. Un enfoque de este tipo para la estimación del espectro de potencia puede verse como un problema de identificación del modelo (sistema).

   Entre los espectros dependientes del modelo definidos aquí, el espectro AR es, el más popular. La razón de esta popularidad se debe a 2 motivos: (1) la forma lineal del sistema de ecuaciones simultáneas que involucran los parámetros desconocidos del modelo AR y (2) la disponibilidad de algoritmos eficientes para calcular la solución.

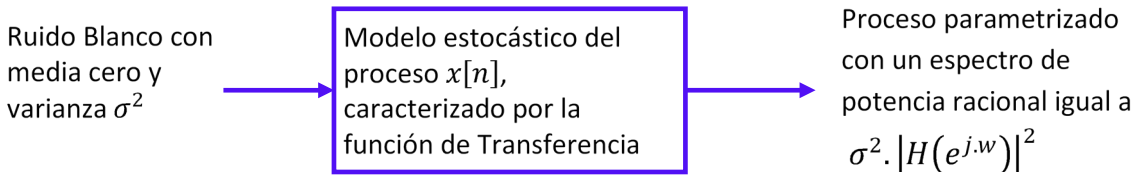


Fig. 126 –Procedimiento de identificación de modelos para la estimación del espectro de potencia.

2) Método de respuesta sin distorsión de varianza mínima (MVDR).
   Para describir este segundo enfoque paramétrico para la estimación del espectro de potencia, considere la situación que se muestra en la Fig. 127. El proceso $x[n]$ se aplica a un filtro de respuesta al impulso de duración finita (FIR) (es decir, un filtro de tiempo discreto con una función de transferencia todos ceros). En el método MVDR, los coeficientes de filtro se eligen para minimizar la varianza (que es la misma que la potencia esperada para un

proceso de media cero) de la salida del filtro, sujeto a la restricción de que la respuesta de frecuencia del filtro es igual a uno (valor unitario) para alguna frecuencia angular w_0. Bajo esta restricción, el proceso $x[n]$ pasa a través del filtro sin distorsión en la frecuencia angular w_0. Además, las señales con frecuencias angulares distintas de w_0 tienden a atenuarse.

3) Métodos basados en la descomposición de matrices (Eigendecomposition-based methods.) En esta última clase de métodos de estimación de espectro paramétrico, la descomposición propia de la matriz de correlación **R** del proceso $x[n]$ se utiliza para definir dos subespacios distintos correspondientes a un subespacio de señal y subespacio de ruido. Esta forma de partición se utiliza luego para derivar un algoritmo apropiado para estimar el espectro de potencia (Schmidt, 1979, 1981).

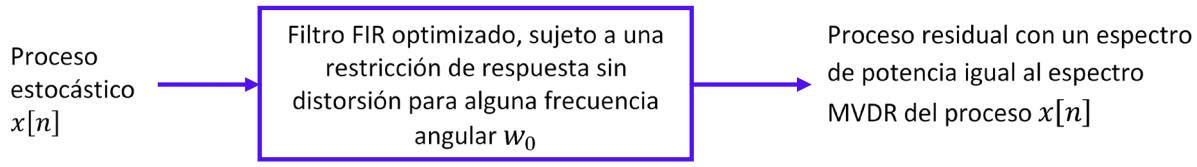


Fig. 127 – Diagrama que ilustra el procedimiento MVDR para la estimación del espectro de potencia.

Métodos no paramétricos

En los métodos no paramétricos de estimación del espectro de potencia, no se hacen suposiciones con respecto al proceso estocástico en estudio. El punto de partida de discusión es la ecuación fundamental de *Representación espectral de Crámer para un proceso estacionario.* Según esta representación, un proceso estocástico de tiempo discreto $\mathrm{x[n]}$ se describe mediante la transformada inversa de Fourier (Thomson, 1982):

$$x[n] = \frac{1}{2.\pi}. \int_{-\pi}^{\pi} e^{j.w.n}. dZ(w) \tag{104}$$

$dZ(w)$ define el proceso de incremento

Para estimar el espectro de potencia se utiliza esta ecuación:

$$X_N(w) = \frac{1}{2.\pi}. \int_{-\pi}^{\pi} K_N(w - v). dZ(v) \tag{105}$$

Siendo $K_N(w)$ el núcleo de Dirichlet:

$$K_N(w) = \frac{e^{j.w.N}.\left(1 - e^{-j.w.(2.N+1)}\right)}{1 - e^{-j.w}} = \frac{seno\left((2.N+1).w/2\right)}{seno(w/2)} \tag{106}$$

Dependiendo de la forma en que se interprete la ecuación (105), podemos distinguir dos enfoques no paramétricos diferentes:

1. Métodos basados en periodogramas. Tradicionalmente, la ecuación fundamental (105) se trata como una convolución de dos funciones de frecuencia. Una, $X(w)$, representa la transformada Fourier en tiempo discreto para una serie temporal infinitamente larga, $\{x[n]\}$;

esta función surge de la definición de la variable de incremento $dZ(w)$ como el producto de $X(w)$ y el incremento de frecuencia dw. La otra función de frecuencia es el núcleo $K_N(w)$, definido por la ecuación (106). Este enfoque nos lleva a considerar la ecuación (94): $S(w) = \lim_{N\to\infty} \frac{1}{N} E\{|X_N(w)|^2\}$, como la definición básica de la densidad espectral de potencia $S(w)$ y por lo tanto el periodograma $|X_N(w)|^2/N$ como punto de partida para el análisis de datos. Sin embargo, el periodograma adolece de una seria limitación en el sentido de que no es una estadística suficiente para analizar en profundidad la densidad espectral de potencia. Esto implica que la información de fase ignorada en el uso del periodograma es esencial. En consecuencia, la insuficiencia estadística del periodograma se hereda mediante cualquier estimación que se base en este método o similar que use periodograma.

2. Método de ventanas múltiples.
   Un enfoque no paramétrico más constructivo es tratar la ecuación fundamental (105) como una ecuación integral de Fredholm de primer tipo para la variable de incremento $dZ(w)$; el objetivo aquí es obtener una solución aproximada de la ecuación con propiedades estadísticas cercanas a las de $dZ(w)$ en algún sentido (Thomson, 1982). La clave para lograr este objetivo importante es el uso de ventanas definidas por un conjunto de secuencias especiales conocidas como secuencias de Slepian, o secuencias esferoidales proladas (cierta geometría) discretas, fundamentales para el estudio de sistemas limitados en el tiempo y la frecuencia. La propiedad notable de esta familia de ventanas es que las distribuciones de energía de las ventanas se suman de una manera muy especial que define colectivamente un contenedor de frecuencia rectangular ideal (en el sentido de la concentración total de energía dentro del contenedor versus la concentración de energía fuera del contenedor). Esta propiedad, a su vez, nos permite cambiar la resolución espectral por propiedades espectrales mejoradas (por ejemplo, varianza reducida de la estimación espectral).

En general, un proceso estocástico de tiempo discreto $x[n]$ tiene un espectro mixto en el sentido de que su espectro de potencia contiene dos componentes: uno determinista y uno continuo. El componente determinista manifiesta el primer momento correspondiente al proceso de incremento $dZ(w)$ y está expresamente dado por:

$$E\{dZ(w)\} = \sum_k a_k . \delta(w - w_k) . dw \tag{107}$$

Donde $\delta(w)$ representa la función delta de Dirac en w, es decir en el dominio de la frecuencia. Las w_k son las frecuencias angulares de los componentes periódicos o de línea contenidos en el proceso $x[n]$, y los a_k son sus amplitudes. El componente continuo, por otro lado, representa el segundo momento central del proceso de incremento, es decir

$$E\{|dZ(w) - E\{dZ(w)\}|^2\} \tag{108}$$

Es importante que se observe cuidadosamente la distinción del primer momento y del segundo momento.

Los espectros calculados que utilizan los métodos paramétricos tienden a tener picos más nítidos y mayor resolución que los obtenidos con los métodos no paramétricos (clásicos). La aplicación de estos métodos paramétricos es, por tanto, muy adecuada para estimar el componente determinista y, en particular, para localizar las frecuencias de componentes periódicos en ruido blanco aditivo cuando la relación señal/ruido es alta. Otra técnica bien probada para estimar el componente determinista es el

método clásico de máxima verosimilitud (máximum Likelihood). Por supuesto, si las leyes físicas que gobiernan la generación de un proceso coinciden con un modelo estocástico (por ejemplo, el modelo AR) de una manera exacta o aproximadamente en algún sentido estadístico, entonces, el método paramétrico correspondiente a ese modelo puede usarse para estimar el espectro de potencia del proceso. Sin embargo, si el proceso estocástico estudiado tiene un espectro de potencia puramente continuo y se desconoce el mecanismo físico subyacente responsable de la generación del proceso, entonces el procedimiento recomendado es el método no paramétrico de múltiples ventanas.

Limitamos nuestra atención a las clases 1 y 2 de métodos paramétricos de estimación del espectro, ya que su teoría encaja naturalmente en los filtros adaptativos.

Densidad de correlación espectral

$$r^{\alpha}(k) = \frac{1}{N}.\left\{\sum_{n=0}^{N-1} E\left[x[n].x^{*}[n-k].e^{-j.2.\pi.\alpha.n}\right]\right\}.e^{j.\pi.\alpha.k}$$

La función de autocorrelación cíclica $r^{\alpha}(k)$, tiene las siguientes propiedades:

a) $r^{\alpha}(k)$ es periódica en α con período 2
b) Para algún α, de la ecuación anterior tenemos: $r^{\alpha+1}(k) = (-1)^{k}.r^{\alpha}(k)$
c) Para el caso especial de $\alpha = 0$, se tiene: $r^{0}(k) = r(k)$. Siendo $r(k)$ la función ordinaria de autocorrelación del proceso estacionario

$S^{\alpha}(w) = \sum_{k=-\infty}^{\infty} r^{\alpha}(k).e^{-jw.k}$; $\quad -\pi < w \leq \pi$

$S^{\alpha}(w)$ es la densidad espectral de la correlación, resulta compleja para $\alpha \neq 0$

Para el caso especial de $\alpha = 0$, se tiene: $S^{0}(w) = S(w)$. Siendo $S(w)$ la densidad espectral de potencia ordinaria

Resumen y discusión

En esta unidad estudiamos la caracterización parcial de un proceso estocástico en tiempo discreto y estacionario, que, en el dominio del tiempo, se describe de manera única en términos de dos parámetros:

1. La media, que es una constante;

2. La función de autocorrelación, que depende únicamente de la diferencia de tiempo entre dos muestras del proceso.

La media del proceso naturalmente puede ser cero, o siempre se puede restar del proceso para producir un nuevo proceso con media cero. Por esa razón, en gran parte de la discusión en sistemas adaptativos, se supone que la media del proceso es cero.

Por lo tanto, dado un vector de observación Mx1 de observaciones $x[n]$ pertenecientes a un proceso estocástico complejo, estacionario y de tiempo discreto con media cero, podemos describir parcialmente el proceso definiendo una matriz de correlación **R** de tamaño MxM como la esperanza estadística del producto externo de $x[n]$ consigo mismo, es decir, $x[n].x^{H}[n]$. La matriz **R** es hermitiana, toeplitz y casi siempre definida positiva.

Otro tema que discutimos en el capítulo es la noción de modelo estocástico, cuya necesidad surge cuando se nos da un conjunto de datos experimentales que se sabe que son de naturaleza estadística y se requiere analizarlos. En este contexto, hay dos requisitos generales para un modelo adecuado:

1. Un *número adecuado de parámetros ajustables* para que el modelo capture el contenido de información esencial de los datos de entrada.

2. *Tratabilidad matemática del modelo*.

El primer requisito, en efecto, significa que la complejidad del modelo debe coincidir estrechamente con la complejidad del mecanismo físico subyacente responsable de generar los datos de entrada; cuando se cumple, se evitan problemas asociados con el desajuste o sobreajuste de los datos de entrada.

El segundo requisito se suele satisfacerse mediante la elección de un modelo lineal.

Dentro de la familia de modelos estocásticos lineales, generalmente se prefiere el modelo autorregresivo (AR) respecto al modelo de media móvil (MA) y el modelo de media móvil autorregresiva (ARMA). Esto se justifica y se explica con una razón importante: a diferencia de la situación en un modelo MA o ARMA, el cálculo de los coeficientes AR se rige por un sistema de ecuaciones lineales, a saber, las ecuaciones de Yule-Walker. Además, excepto por un componente predecible, podemos aproximar un proceso estocástico estacionario en tiempo discreto mediante un modelo AR de orden alto, sujeto a ciertas restricciones. Para seleccionar un valor adecuado para el orden del modelo, podemos utilizar un criterio teórico de la información según Akaike o según el criterio de longitud mínima de descripción (MDL) descripto por Rissanen. Una característica útil del criterio MDL es que es un estimador consistente de orden de modelo.

Otra forma importante de caracterizar un proceso estocástico estacionario de sentido amplio esta dado mediante la de densidad espectral de potencia o también conocido como espectro de potencia. Identificamos tres parámetros espectrales distintos que dependen de las características estadísticas del proceso:

1) La densidad espectral de potencia $S(w)$, definida como la transformada Fourier en tiempo discreto de la función de autocorrelación ordinaria de un proceso estacionario en sentido amplio. Para tal proceso, la función de autocorrelación es hermitiana, donde $S(w)$ toma valores reales. En consecuencia, $S(w)$ destruye la información de fase del proceso. A pesar de esta limitación, la densidad espectral de potencia se acepta comúnmente como un parámetro útil para mostrar las propiedades de correlación de un proceso estacionario en sentido amplio.
2) Poliespectra $C_k(w_1, w_2, \ldots . w_{k-1})$, definido como la transformada de Fourier multidimensional de los procesos estacionarios acumulados. Para estadísticas de segundo orden, $k = 2$, $C_2(w_1)$ se reduce a la densidad espectral de potencia ordinaria $S(w)$. Para estadísticas de orden superior, $k > 2$, los poliespectros $C_k(w_1, w_2, \ldots . w_{k-1})$ adoptan formas complejas. Esta propiedad de los poliespectros los convierte en una herramienta útil para afrontar situaciones en las que se requiere el conocimiento de la fase. Sin embargo, para que los poliespectros sean significativos, el proceso debe ser no gaussiano, y la explotación de la información de fase contenida en los poliespectros requiere el uso de un filtrado no lineal.
3) La densidad de correlación espectral: $S^{\alpha}(w)$, no pierde la información de fase. Se define como la transformada de Fourier en tiempo discreto de la función de autocorrelación cíclica de un proceso que es cicloestacionario en el sentido amplio. Para un $\alpha \neq 0$, $S^{\alpha}(w)$ toma valores complejos; para $\alpha \neq 0$, se reduce a $S(w)$. La característica útil de $S^{\alpha}(w)$ es que conserva la información de fase, que puede explotarse mediante filtrado lineal, independientemente de que el proceso sea o no gaussiano.

Las diferentes propiedades de los parámetros estadísticos de la densidad espectral de potencia ordinaria, los poliespectros y la densidad de correlación espectral otorgan a cada uno de ellos diferentes áreas de aplicación en el filtrado adaptativo.

Las teorías de los procesos ciclo estacionarios de segundo orden y las poliespectrales convencionales se han reunido con el nombre de poliespectrales cíclicos. En pocas palabras, los poliespectrales cíclicos son acumulativos espectrales en los que las frecuencias individuales involucradas pueden sumar cualquier frecuencia de ciclo α, mientras que deben sumar cero para las poliespectrales.

Ejercicios

Ejercicio 3.1

Análisis espectral de sistemas – Teorema de Wiener–Khinchin (W-K)

Dado el diagrama de un filtro pasa banda descripto en la Fig. 128, se pide:

a) Calcular $R_{yy}(\tau)$ y $G_{yy}(w)$ por 2 métodos diferentes

b) Considerando la entrada $x(t)$ Ruido Blanco Gaussiano con DEP G_0, calcular la DEP de la salida y graficar

c) Calcular $R_{xy}(\tau)$ y $G_{xy}(w)$

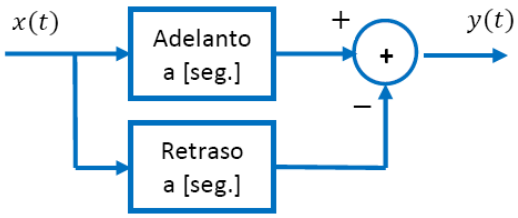


Fig. 128 – Diagrama de filtro pasa banda

Resolución

a) **Método 1: utilizamos DEP**

$y(t) = x(t+a) - x(t-a)$

$R_{yy}(\tau) = \mathcal{F}^{-1}\{G_{yy}(w)\}$; $\boxed{G_{yy}(w) = G_{xx}(w).|H(w)|^2}$

Calculamos $H(w)$:

Por Transformada de Fourier de $y(t)$:

$Y(w) = e^{j.w.a}.X(w) - e^{-j.w.a}.X(w) = X(w).2.j.seno(w.a)$

$H(w) = \frac{Y(w)}{X(w)} = 2.j.seno(w.a)$;

Reemplazando en $G_{yy}(w)$

$\boxed{G_{yy}(w) = G_{xx}(w).\ 4.seno^2(w.a)}$

Calculamos $R_{yy}(\tau)$:

$seno^2(\alpha) = \frac{1}{2}.\left(1 - \cos(2.\alpha)\right)$

$\rightarrow \;\; G_{yy}(w) = G_{xx}(w).\; 2.\left(1 - \cos(2.w.a)\right)$

$G_{yy}(w) = 2.G_{xx}(w) - 2.G_{xx}(w).\cos(2.w.a)$

$G_{yy}(w) = 2.G_{xx}(w) - 2.G_{xx}(w).\frac{e^{j.2.wa} + e^{-j.2.wa}}{2}$

$G_{yy}(w) = 2.G_{xx}(w) - G_{xx}(w).e^{j.2.wa} - G_{xx}(w).e^{-j.2.wa}$

Antitransformando $G_{yy}(w)$:

$$\boxed{R_{yy}(\tau) = 2.R_{xx}(\tau) - R_{xx}(\tau + 2.a) - R_{xx}(\tau - 2.a)}$$

Resolución Método 2: Por Definición: $\boxed{R_{yy}(\tau) = E\{y(t).y(t+\tau)\}}$

Reemplazamos: $\boxed{y(t) = x(t+a) - x(t-a)}$

$R_{yy}(\tau) = E\{[x(t+a) - x(t-a)].[x(t+\tau+a) - x(t+\tau-a)]\}$

$$R_{yy}(\tau) = E\{[x(t+a).x(t+\tau+a) - x(t+a).x(t+\tau-a) \; - x(t-a).x(t+\tau+a) + x(t-a).x(t+\tau-a)]\}$$

$R_{yy}(\tau) = E\{x(t+a).x(t+a+\tau)\} - E\{x(t+a).x(t-a+\tau)\} - E\{x(t-a).x(t+a+\tau)\} + E\{x(t-a).x(t-a+\tau)\}$

Considerando al sistema Estacionario, la autocorrelación depende solamente de las diferencias:

$t + \tau - t = \tau$; Para cada uno de los 4 términos obtenemos las diferencias:

$t + a + \tau - (t+a) = \tau \;\; ; \;\; t - a + \tau - (t+a) = \tau - 2.a$

$t + a + \tau - (t-a) = \tau + 2.a \;\; ; \;\; t - a + \tau - (t-a) = \tau$

$R_{yy}(\tau) = R_{xx}(\tau) - R_{xx}(\tau - 2.a) - R_{xx}(\tau + 2.a) + R_{xx}(\tau)$

$\boxed{R_{yy}(\tau) = 2.R_{xx}(\tau) - R_{xx}(\tau - 2.a) - R_{xx}(\tau + 2.a)}$ Ídem resultado anterior

La DEP de la salida la obtenemos mediante el Teorema W-K: $G_{yy}(w) = \mathcal{F}\{R_{yy}(\tau)\}$

b) Considerando la entrada $x(t)$ Ruido Blanco Gaussiano con DEP G_0, calcular la DEP de la salida y graficar

Del Método1: $G_{yy}(w) = 4.G_{xx}(w).\; seno^2(w.a)$

$G_{yy}(w) = 4.G_0.\; seno^2(w.a) \;\; ; w_{ceros}.a = k.\pi$

En la Fig. 129 se grafica la DEP correspondiente a la salida del sistema.

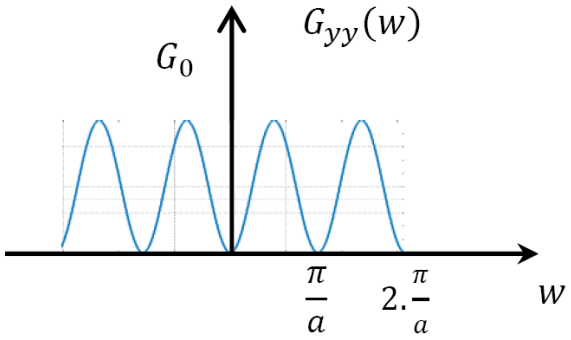


Fig. 129 – Densidad espectral de Potencia a la salida: $G_{yy}(w)$

Conclusiones: $R_{yy}(\tau)$ y $G_{yy}(w)$ por 2 métodos diferentes: Teorema W-K y por Definición

Resulta más simple el primer método

c) Calcular $R_{xy}(\tau)$ y $G_{xy}(w)$

Siendo: $y(t) = x(t+a) - x(t-a)$

$R_{xy}(\tau) = E\{x(t).y(t+\tau)\} = E\{x(t).[x(t+\tau+a) - x(t+\tau-a)]\}$

Aplicamos propiedad distributiva y propiedad: $E\{x(t) + g(t)\} = E\{x(t)\} + E\{g(t)\}$

$R_{xy}(\tau) = E\{x(t).x(t+\tau+a)\} - E\{x(t).x(t+\tau-a)\}$

$$\boxed{R_{xy}(\tau) = R_{xx}(\tau+a) - R_{xx}(\tau-a)}$$

$G_{xy}(w) = \mathcal{F}\{R_{xy}(\tau)\}$ Teorema W-K

$G_{xy}(w) = G_{xx}(w).e^{j.w.a} - G_{xx}(w).e^{-j.w.a} = G_{xx}(w).\left(e^{j.w.a} - e^{-j.w.a}\right)$

$G_{xy}(w) = G_{xx}(w).2.j.sen(w.a)$

Ejercicio 3.2

Análisis espectral de sistemas – Teorema de Wiener–Khinchin

Se puede hallar la respuesta de un sistema ingresando a la entrada $x(t) = \delta(t)$ y medir su salida, obteniendo: $y(t) = h(t) * x(t) \quad \rightarrow y(t) = h(t) * \delta(t) = h(t)$

Sin embargo, en la práctica es difícil generar una función similar a la función ideal $\delta(t)$.

Por tal motivo, se utiliza otra técnica moderna para estimar la transferencia de un sistema, ***«se obtiene la correlación cruzada entre su entrada y su salida cuando se excita al sistema con Ruido Blanco Gaussiano (RBG)»***. Este proceso se esquematiza en la Fig. 130.

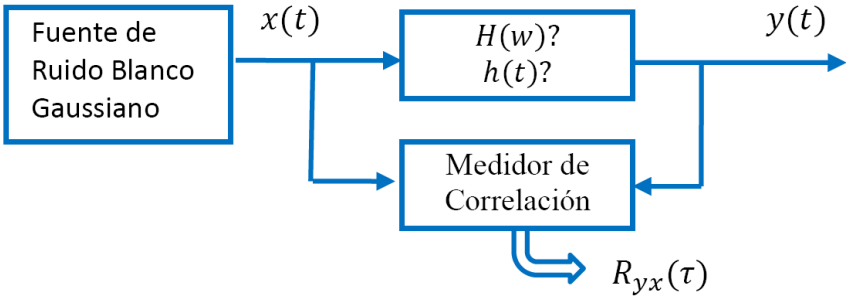


Fig. 130 – Diagrama para medición de correlación cruzada y fuente de RBG

Suponiendo que al colocar RBG a la entrada con $R_{xx}(\tau) = N_0.\delta(t)$, se obtuvo la siguiente densidad espectral de potencia cruzada: $G_{yx}(w) = 20.e^{-5.|w|}$.

Se pide:

a) Calcule la respuesta en frecuencia del sistema

b) Calcule la respuesta al impulso del sistema

c) Repetir a) y b) suponiendo que se obtuvo $G_{yx}(w) = e^{-a.w}$

Resolución

$G_{xx}(w) = \mathcal{F}\{R_{xx}(\tau)\} = \mathcal{F}\{N_0.\delta(t)\} = N_0$; Aplicamos Teorema W-K:

$\boxed{G_{xx}(w) = N_0}$ (constante ∀w).

$h(t)$ es la respuesta impulsional desconocida del sistema.

$R_{yx}(\tau) = y(\tau) * x(-\tau) = h(\tau) * x(\tau) * x(-\tau) = h(\tau) * R_{xx}(\tau)$

$R_{yx}(\tau) = h(\tau) * R_{xx}(\tau)$ Aplicando Teorema W-K: $G_{yx}(w) = \mathcal{F}\{R_{yx}(\tau)\}$

$\boxed{G_{yx}(w) = H(w).G_{xx}(w)}$ (se demuestra la DEP Cruzada)

a) $G_{yx}(w) = 20.e^{-5.|w|}$ (enunciado)

$G_{yx}(w) = H(w).G_{xx}(w)$; $H(w) = \frac{G_{yx}(w)}{G_{xx}(w)} = \frac{1}{N_0}.G_{yx}(w)$

$\boxed{H(w) = \frac{20}{N_0}.e^{-5.|w|}}$

b) $h(t) = \mathcal{F}^{-1}\{H(w)\} = \mathcal{F}^{-1}\left\{\frac{20}{N_0}.e^{-5.|w|}\right\}$

$e^{-a.|t|} \longleftrightarrow \frac{2.a}{a^2+w^2}$; Aplicamos dualidad: $\frac{2.a}{a^2+t^2} \longleftrightarrow 2.\pi.e^{-a.|w|}$; $\frac{a}{\pi(a^2+t^2)} \longleftrightarrow e^{-a.|w|}$

$\rightarrow \boxed{h(t) = \frac{100}{N_0}.\frac{1}{\pi(5^2+t^2)}}$

Conclusiones: La Respuesta Impulsional h(t) la obtenemos con la Transformada de Fourier Inversa de la correlación medida en el proceso. «No se usa $x(t) = \delta(t)$, sino RBG»

Ejercicio 3.3

Identificación de Sistemas con correlación cruzada y Teorema W-K

Dado el esquema de la Fig. 131 analizado anteriormente, se pide implementar un programa en Matlab® para identificar un sistema desconocido

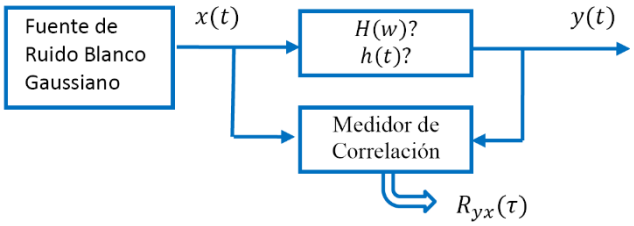


Fig. 131 – Diagrama para medición de correlación cruzada y fuente de RBG

Resolución

Usamos esta fórmula $H(w) = \frac{G_{yx}(w)}{G_{xx}(w)}$

Resolvemos utilizando Transformada de Fourier Discreta, a continuación, se muestra el programa en Matlab®.

```
%% Uso de correlación cruzada para identificar sistema FIR desconocido
 L_x = 35; % Largo de la señal de entrada
 x = rand(1,L_x); % Señal de entrada
 L_h = 14; % Cantidad de coeficientes del filtro
 %h = 1:L_h;    % Filtro con coeficientes 1  2  3  4  ...
 h=rand(1,L_h) ;
 L_y = L_x + L_h -1; % Largo máximo de la salida
 % próxima potencia 2
 L_fft = 2^nextpow2(L_y);
 x1 = [x,zeros(1,L_fft-L_x)]; % completamos con ceros entrada
 y1 = filter(h,1,x1); % Filtramos
 X = fft(x1);   Y = fft(y1); % Espectros de entrada y de salida
 Gxx = conj(X) .* X; % DEP de la entrada
 Gxy = conj(X) .* Y; % DEP cruzada
 Hxy = Gxy ./ Gxx;   % Respuesta en frecuencias H
 hxy = ifft(Hxy);    % Respuesta al impulso
% Mostramos coeficientes de h estimada (los primeros)
 hxy(1:L_h)
% Graficamos respuesta en frecuencia
```

```
 freqz(hxy,1,L_fft);  % plot estimated freq response
 figure ;stem(h) ; hold on ; stem(hxy,'r');  axis tight
legend('Coeficientes Reales','Coeficientes Estimados') ;
% Anexo cálculo de errores
 err = norm(hxy - [h,zeros(1,L_fft-L_h)])/norm(h);
 disp(sprintf(['Error en la respuesta impulsional = ','%0.14f%%'],100*err));
 err = norm(Hxy-fft([h,zeros(1,L_fft-L_h)]))/norm(h);
 disp(sprintf(['Error en la respuesta en frecuencia = ','%0.14f%%'],100*err));
```

Resultados Matlab®

Impulse Response Error = 0.00000000000026%

Frequency Response Error = 0.00000000000206%

Se obtiene error de estimación muy bajo. En la Fig. 132 se muestran los resultados obtenidos.

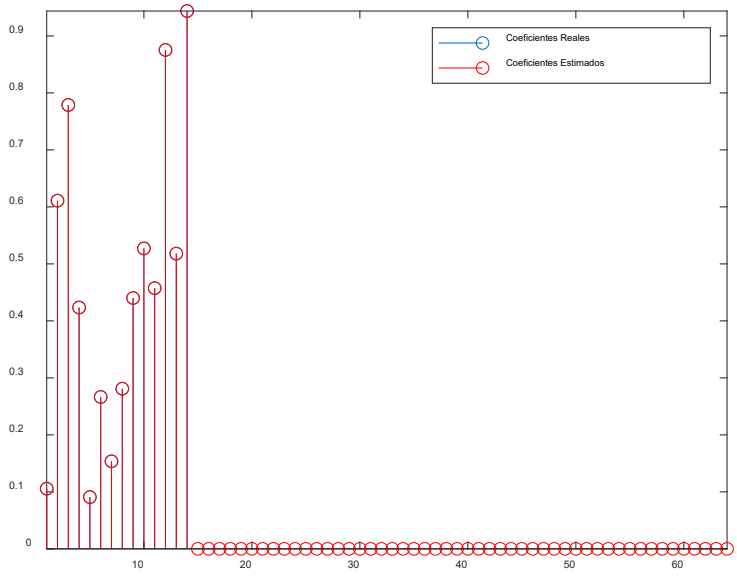


Fig. 132 – Resultados con coeficientes reales y coeficientes estimados

Ejercicio 3.4

Análisis de señales de ruido y señales no correlacionadas

Se disponen de las siguientes señales provenientes de dos receptores de radio:

$x_1(t) = S(t) + N_1(t) \quad ; x_2(t) = S(t) + N_2(t)$

Donde $S(t)$, $N_1(t)$ y $N_2(t)$ son variables aleatorias de media nula, no correlacionadas y estacionarias. Siendo $S(t)$ la señal transmitida, $N_1(t)$ y $N_2(t)$ son ruidos agregados en la transmisión. Calcular por definición:

a) La autocorrelación de $x_1(t)$ y la de $x_2(t)$

b) La correlación cruzada entre $x_1(t)$ y $x_2(t)$

c) La autocorrelación de $x_1(t) + x_2(t)$

Resolución

a) $R_{xx}(\tau) = E\{x(t).x(t+\tau)\}$

$R_{x1x1}(\tau) = E\{[S(t) + N_1(t)].[S(t+\tau) + N_1(t+\tau)]\}$

$R_{x1x1}(\tau) = E\{S(t).\ S(t+\tau) + \ S(t).N_1(t+\tau) \ + N_1(t).S(t+\tau) \ + N_1(t).N_1(t+\tau)\}$

$R_{x1x1}(\tau) = E\{S(t).\ S(t+\tau)\} + \ E\{S(t).N_1(t+\tau)\} + \ E\{N_1(t).S(t+\tau)\} \ + \ E\{N_1(t).N_1(t+\tau)\}$

$E\{S(t).N_1(t+\tau)\} = 0$ *son señales No correlacionadas*

$R_{x1x1}(\tau) = R_{ss}(\tau) + 0 + 0 + R_{N1.N1}(\tau)$

$\boxed{R_{x1x1}(\tau) = R_{ss}(\tau) + R_{N1.N1}(\tau)}$; Análogamente se llega a: $\boxed{R_{x2x2}(\tau) = R_{ss}(\tau) + R_{N2.N2}(\tau)}$

Ejercicio 3.5

Aplicación del Teorema de Wiener–Khinchin

Mediante un ejemplo se pide aplicar con Matlab® la Propiedad 1 – Teorema W-K

$$S(w) = F\{r[l]\}$$

$$r[l] = F^{-1}\{S(w)\}$$

$$r[l] \longleftrightarrow S(w)$$

Resolución: DEP: $S(w) = X(w).X(w)^* = F\{x\}.F\{x\}^*$

$$\boxed{R_{xx} = r[l] = F^{-1}\{S(w)\} = F^{-1}\{X(w).X(w)^*\}}$$

Debemos verificar *con Matlab®*

$$R_{xx} = F^{-1}\{X(w).X(w)^*\}$$

```
% Corr(x,y) <---> FFT(x)FFT(y)*
% Corr(x,x) <---> FFT(x)FFT(x)*
% Generamos señal
   x = rand(100,1);  len = length(x) ;
% Calculamos autocorrelación mediante la Transformada Inversa de DEP
   largo_fft = 2^nextpow2(2*len - 1) ;
   r1 = ifft( fft(x,largo_fft) .* conj(fft(x,largo_fft)) );
```

```
 % reordenamos valores al centro (vector lags): -(len-1):+(len-1)
   Rxx1 = [r1(end-len+2:end) ; r1(1:len)];
% Calculamos autocorrelación mediante xcorr
[Rxx2,lags] = xcorr(x) ; plot(lags, Rxx2, lags, Rxx1)
% Comparamos resultados de ambos métodos
   all( (Rxx2-Rxx1) < 1e-10 )
```

Respuesta = 1; Son Iguales Rxx2 y Rxx1

Ejercicio 3.6

Formación de haces Adaptativo.

Ejercicios en Matlab® de MVDR

Repasar Adaptación Espacial de la Unidad I, ver Fig. 133

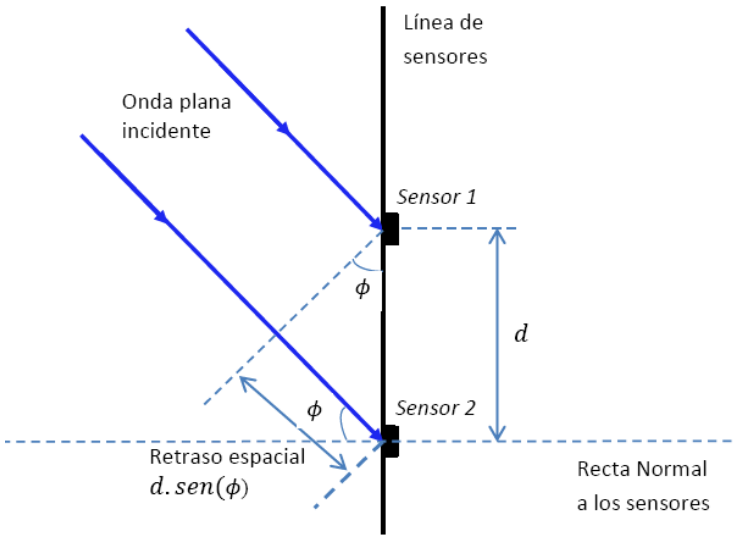


Fig. 133 – Esquema de adaptación espacial

Analizar el funcionamiento de los siguientes programas

a) Escribir en línea de comandos Matlab®

openExample('phased/MVDRBeamformerExample')

A continuación, se muestra el programa del punto a) modificado para una mejor comprensión.

```
% Construir una matriz de líneas espaciadas en media longitud de onda de 10 elementos.
% Elegir dos direcciones de llegada de interés: una a 30 ° de azimuth y la otra a 45 ° de azimuth.
```

```
% Suponga que ambas direcciones están a 0 ° de elevación. Calcule los pesos del formador de haz
% MVDR para cada dirección. Especifique una matriz de covarianza espacial de sensor que contenga
% señales que llegan de -60 ° y 60 ° y ruido a -10 dB.
% Configuramos la matriz de covarianza espacial de matriz y del sensor
 N = 10 ;
 d = 0.5 ;
 elemento_posicion = (0:N-1) *d;
 Sn  = sensorcov(elemento_posicion,[-60, 60],db2pow(-10));
% Calculamos los pesos del formador de haz MVDR.
 w  =  mvdrweights(elemento_posicion,[30 45],Sn);
% Graficamos los dos patrones de matriz MVDR.
 plot_angulo = -90: 90;
 vv = steervec(elemento_posicion, plot_angulo);
 plot(plot_angulo, mag2db(abs(w'*vv)), 'linewidth',2)
 grid on
 xlabel('Ángulo Azimuth (grados)');
 ylabel('Potencia Normalizada (dB)');
 legend('30 grados','45 grados');
 title('Patrón de matriz MVDR')
```

En la Fig. 134 se muestran los resultados Matlab® para cada dirección del formador de haz. Una sección tiene la ganancia máxima esperada a 30 grados y la otra a 45 grados. Los nulos en -60 y 60 grados surgen de la propiedad fundamental del formador de haz MVDR de suprimir la potencia en todas las direcciones excepto en la dirección de llegada.

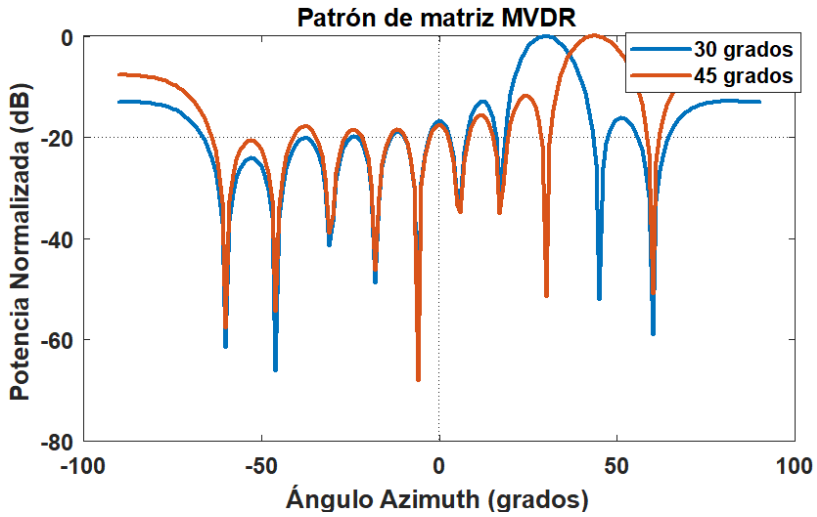


Fig. 134 – Patrón de matriz MVDR para 30 y 45 grados

b) Escribir en línea de comandos Matlab®

openExample('phased/slexBeamscanMVDRDOAExampleExample')

En la Fig. 135 se muestran los resultados del espectro MVDR.

El siguiente programa de Matlab® contiene un ejemplo más completo: Abrir slexBeamscanMVDRDOAExampleExample.m

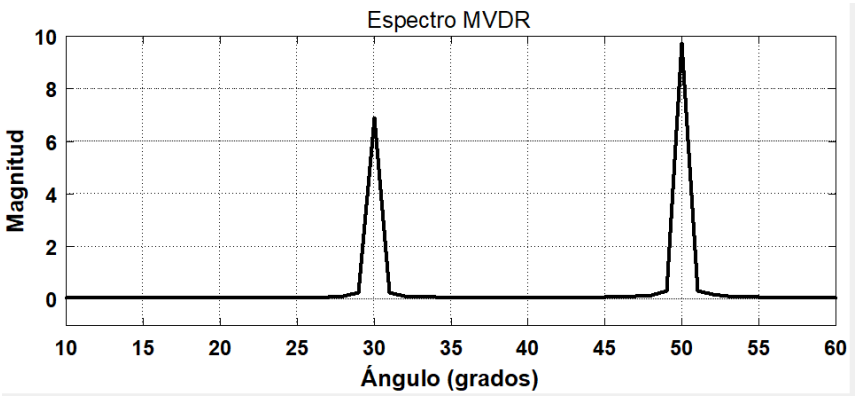


Fig. 135 – Espectro MVDR para ondas de 30 y 50 grados

Descarga de los códigos de los ejercicios

IV – Algoritmos de Filtrado Lineal Adaptativo

Introducción al desarrollo de algoritmos de filtros lineales adaptativos

No existe una solución única para el problema del filtrado adaptativo lineal. Más bien, tenemos un conjunto de herramientas representado por una variedad de algoritmos recursivos, cada uno de los cuales ofrece ventajas propias. El desafío que debe enfrentar el usuario del filtro adaptativo es, en primer lugar, comprender las capacidades y limitaciones de varios algoritmos de filtrado adaptativo y, en segundo lugar, utilizar este conocimiento en la selección del algoritmo apropiado para la aplicación en cuestión.

En la literatura se ha desarrollado una amplia variedad de algoritmos recursivos para el funcionamiento de filtros adaptativos lineales. La elección de un algoritmo sobre otro está determinada por uno o más de los siguientes factores:

• **Robustez**. Para que un filtro adaptativo sea robusto, pequeñas perturbaciones (es decir, perturbaciones con poca energía) solo pueden resultar en pequeños errores de estimación. Las perturbaciones pueden deberse a una variedad de factores, internos o externos al filtro.

• **Requisitos computacionales.** Aquí las cuestiones de interés incluyen (a) el número de operaciones (es decir, multiplicaciones, divisiones y sumas / restas) necesarias para hacer un ciclo de adaptación completo del algoritmo, (b) el tamaño de la memoria necesaria para almacenar los datos y el programa, y (c) la inversión necesaria para programar el algoritmo en una computadora.

• **Tasa de convergencia**. Esto se define como el número de ciclos de adaptación necesarios para que el algoritmo, en respuesta a entradas estacionarias, converja "lo suficientemente cerca" de la solución de Wiener óptima en el sentido del error cuadrático medio. Una tasa rápida de convergencia permite que el algoritmo se adapte rápidamente a un entorno estacionario de estadísticas desconocidas.

• **Desajuste**. Para un algoritmo de interés, este parámetro proporciona una medida cuantitativa de la cantidad por la cual el valor final del error cuadrático medio, promediado sobre un conjunto de filtros adaptativos, se desvía de la solución de Wiener.

• **Seguimiento**. Cuando un algoritmo de filtrado adaptativo opera en un entorno no estacionario, se requiere del algoritmo que rastree las variaciones estadísticas en el entorno. El rendimiento de seguimiento del algoritmo, sin embargo, está influenciado por dos características contradictorias: (1) tasa de convergencia y (2) fluctuación de estado estable debido al ruido del algoritmo.

• **Estructura.** Esto se refiere a la estructura del flujo de información en el algoritmo, determinando la manera en que se implementa en forma de hardware. Por ejemplo, un algoritmo cuya estructura presenta una alta modularidad, paralelismo o simultaneidad es muy adecuado para la implementación mediante la integración de dispositivos a gran escala (VLSI).

• **Propiedades numéricas**. Cuando un algoritmo se implementa numéricamente, se producen inexactitudes debido a errores de cuantificación, que a su vez se deben a la conversión analógica a digital de los datos de entrada y la representación digital de los cálculos internos. Por lo general, es la última fuente de errores de cuantificación la que plantea un problema de diseño serio. En particular, hay dos cuestiones básicas de interés: la estabilidad y la precisión numérica. La estabilidad numérica es una característica inherente de un algoritmo de filtrado adaptativo. La precisión numérica, por otro lado, está determinada por el número de bits. Se dice que un algoritmo de filtrado adaptativo es numéricamente robusto cuando es insensible a las variaciones en la longitud de la palabra utilizada en su implementación digital.

Estos factores, a su manera, también entran en el diseño de filtros adaptativos no lineales, excepto por el hecho de que ahora ya no tenemos un marco de referencia bien definido en forma de filtro Wiener. Más bien, hablamos de un algoritmo de filtrado no lineal que puede converger a un mínimo local o, con suerte, a un mínimo global en la superficie de rendimiento del error.

Básicamente, podemos identificar dos enfoques distintos para implementar algoritmos correspondientes a filtros adaptativos lineales, LMS y RLS

En la Fig. 136 se muestra un esquema básico de un sistema adaptativo.

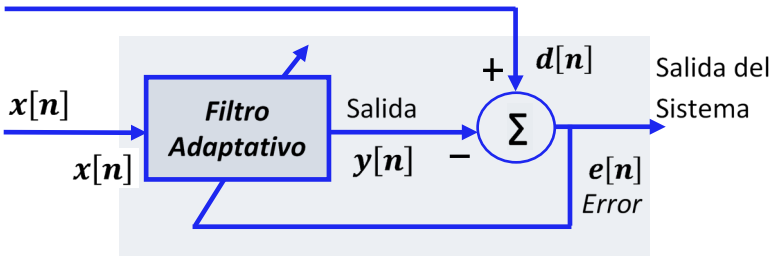


Fig. 136 – Sistema adaptativo

$x[n]$: entrada del filtro

$y[n]$: salida del filtro

$e[n]$: error generado

$d[n]$: función deseada

El filtro, después de un tiempo que realice varias iteraciones, deberá generar una aproximación a la función deseada $d[n]$. Se debe reducir el error $e[n]$. En filtros adaptativos se utiliza mucho minimizar el error cuadrático medio (MSE) mediante la siguiente función de costo:

$$J = E\{e^2[n]\} = E\{d^2[n] - 2.d[n].y[n] + y^2[n]\}$$

El filtro del sistema adaptativo podría ser un filtro IIR, filtro lattice, un combinador lineal de entradas adaptativo u otra estructura. Utilizaremos el filtro FIR.

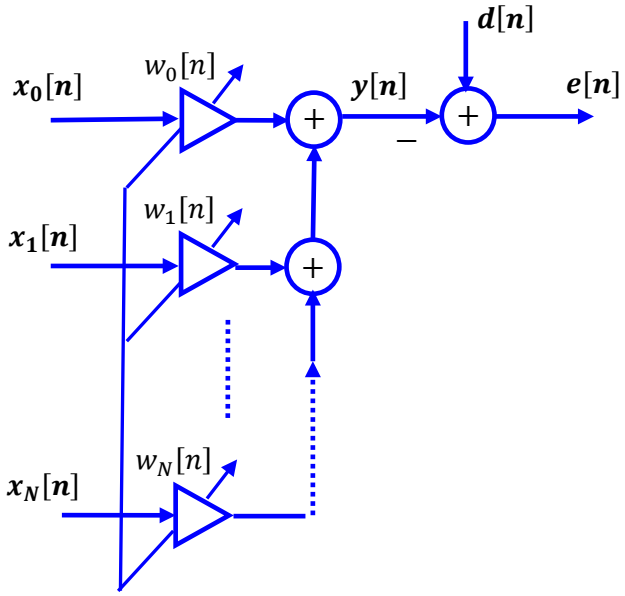


Fig. 137 – Combinador Lineal de Entradas Adaptativo

Se pueden implementar diferentes filtros. Analizaremos el caso del filtro FIR, escribimos su ecuación característica para con coeficientes fijos:

$$y[n] = \sum_{k=0}^{N-1} b_k . x[n-k]$$

Los filtros adaptativos buscan hallar los valores más adecuados de los coeficientes w_k, de manera de minimizar la salida de error $e[n]$ del sistema adaptativo. Estos coeficientes se actualizan a medida que va cambiando la entrada al sistema. En la Fig. 68 se muestra un esquema del Filtro Adaptativo FIR. La salida del filtro $y[n]$ se expresa mediante la siguiente ecuación:

$y[n] = \sum_{i=0}^{N} w_i[n] . x[n-i] = w^T[n] . x[n]$

Siendo $x[n] = \left[x[n],\ x[n-1], \ldots \ldots . x[n-N]\right]^T$

$w[n] = \left[w_0[n],\ w_1[n], \ldots \ldots . w_N[n]\right]^T$

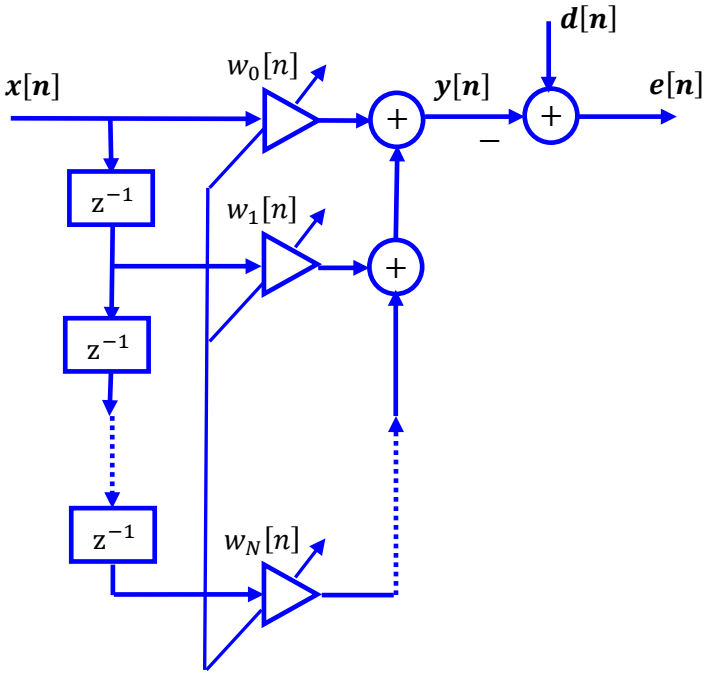


Fig. 138 – Filtro Adaptativo FIR

La señal de entrada es $x[n]$. La señal de error es $e[n]$ resulta de la resta entre la señal deseada $d[n]$ y la salida del filtro $y[n]$.

La salida del filtro se obtiene mediante:

$$y[n] = w^T[n].x[n] \qquad (109)$$

Siendo $w[n]$ los coeficientes del filtro

Se obtiene la Ecuación para el cálculo de la señal de error:

$$e[n] = d[n] - y[n] \qquad (110)$$

Generalmente se trata de que la señal $e[n]$ sea cero, para esto a partir de la señal $x[n]$ se genera $y[n]$ y se van ajustando los coeficientes del filtro para que $y[n]$ sea igual a la señal esperada $d[n]$.

Se disponen de distintos métodos pata minimizar la señal de error $e[n]$. Por ejemplo, en caso de conocer la estadística de las señales, se puede minimizar la siguiente función: $J = E\{|e[n]|^2\}$. O en caso de no conocer la estadística, se puede simplificar minimizando la función: $J_s = |e[n]|^2$. Mediante el algoritmo de gradiente descendente, se obtiene:

$$w[n+1] = w[n] - \frac{\mu}{2}.\nabla J \qquad (111)$$

Siendo μ una constante que modifica la velocidad de convergencia del algoritmo.

Mediante el método LMS (Least Mean Square), se tiene la siguiente ecuación:

$$w[n+1] = w[n] + \mu.e[n].x[n] \qquad (112)$$

En cada instante n se tienen los siguientes tamaños de vectores:

$$w[n+1] \quad \rightarrow \quad Nx1$$

$$w[n] \quad \rightarrow \quad Nx1$$

$$e[n] \rightarrow 1x1$$
$$x[n] \rightarrow Nx1$$

A modo de ejemplo, para $\mu = 0{,}01$ y filtro de un solo coeficiente se tiene: $w[n+1] = w[n] + 0{,}01.x[n].e[n]$;

Siendo $y[n] = w[n].x[n]$; $e[n] = d[n] - y[n]$

Método de gradiente estocástico descendente

El enfoque de gradiente estocástico utiliza una línea de retardo con derivación, o filtro FIR, como base estructural para implementar el filtro adaptativo lineal. Para el caso de señales estacionarias, la función de costo J, también conocida como índice de desempeño, se define como el error cuadrático medio (es decir, el valor cuadrático medio de la diferencia entre la respuesta deseada y la salida del filtro FIR). Esta función de costo es precisamente una función de segundo orden de los pesos de los bloques del filtro FIR. La dependencia del error cuadrático medio de los pesos desconocidos de los bloques puede verse en forma de un paraboloide multidimensional (es decir, un "tazón") con un fondo definido de forma única o un punto mínimo. Como se mencionó anteriormente, nos referimos a este paraboloide como la superficie de comportamiento del error; los pesos de los bloques correspondientes al punto mínimo de la superficie definen la solución óptima de Wiener.

Para desarrollar un algoritmo recursivo con el fin de actualizar los pesos del filtro FIR adaptativo utilizando el enfoque de gradiente estocástico, como su nombre lo implica, debemos comenzar con una función de costo estocástico. Para tal función, por ejemplo, podemos usar el valor instantáneo al cuadrado de la señal de error, definido como la diferencia entre la respuesta deseada suministrada externamente y la respuesta real del filtro FIR a la señal de entrada.

$$J_s = |e[n]|^2$$

Luego, al derivar esta función de costo estocástico con respecto al vector de pesos de del filtro, obtenemos un vector de gradiente que es naturalmente estocástico. Con el deseo de avanzar hacia la optimización, la adaptación se realiza a lo largo de la dirección "negativa" del vector de gradiente.

En la Fig. 139 se muestra la función de error cuadrático medio J en función de w para el caso 2 coeficientes.

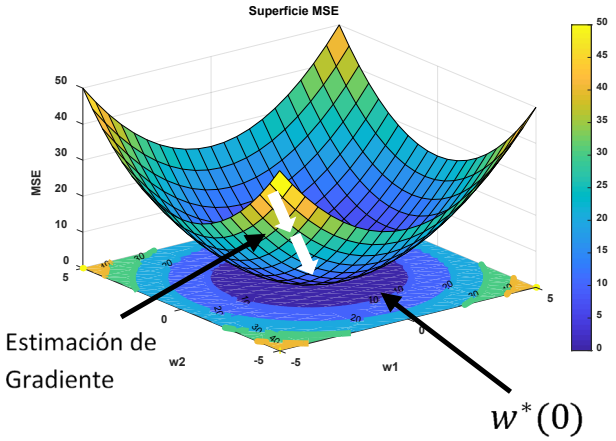


Fig. 139 – Función de error cuadrático medio J en función de w.

Algoritmo LMS

El algoritmo de mínimos cuadrados medios LMS (en inglés Least Mean Squares) fue desarrollado por Widrow y Hoff (Widrow &Hoff, 1960). Este algoritmo de filtrado adaptativo resulta del enfoque de pasos descendentes, puede expresarse en palabras de la siguiente manera:

[Vector de pesos actualizados] = [Vector de pesos anterior] + [Parámetro de tasa de aprendizaje] x [Vector de entrada] x [señal de error]

El parámetro de tasa de aprendizaje determina la tasa a la que se realiza la adaptación. El algoritmo recursivo así descrito se denomina LMS, que es simple en términos computacionales y eficaz en rendimiento. Sin embargo, su comportamiento de convergencia es lento y difícil de estudiar matemáticamente.

Procesos básicos del Algoritmo LMS:

- Proceso de Filtrado
- Calcular la salida del filtro. En caso de filtro FIR convolucionar la entrada con los coeficientes
- Estimar el error
- Proceso de adaptación
- Ajustar los pesos

A continuación, se muestra el cálculo del error cuadrático medio (MSE).

$$y[n] = w^T[n].x[n] \qquad (113)$$

$$e[n] = d[n] - y[n] = d[n] - w^T[n].x[n] \qquad (114)$$

$$J = e^2[n] = (d[n] - w^T[n].x[n])^2 \qquad (115)$$

$$e^2[n] = d^2[n] - 2.d[n].w^T[n].x[n] + w^T[n].x[n].x^T[n].w[n] \qquad (116)$$

Aplicamos en ambos miembros el operador $E\{\ \}$ correspondiente al valor esperado

$$E\{e^2[n]\} = E\{d^2[n] - 2.d[n].w^T[n].x[n] + w^T[n].x[n].x^T[n].w[n]\} \qquad (117)$$

$$J = E\{e^2[n]\} = E\{d^2[n]\} - 2.E\{d[n].w^T[n].x[n]\} + E\{w^T[n].x[n].x^T[n].w[n]\} \qquad (118)$$

Para un filtro con coeficientes fijos ($w = w[n]$), la función MSE para un entorno estacionario se puede reescribir:

$$J = E\{d^2[n]\} - 2.w^T.E\{d[n].x[n]\} + w^T.E\{x[n].x^T[n]\}.w \qquad (119)$$

Teniendo en cuenta:

$J = E\{e^2[n]\}$: MSE, es el error cuadrático medio

$\sigma^2 = E\{d^2[n]\}$: autocorrelación de la señal deseada

$p = E\{d[n].x[n]\}$: correlación cruzada entre $d[n]$ y $x[n]$

$R = E\{x[n].x^T[n]\}$: autocorrelación de la entrada $x[n]$

Obtenemos:

$$J = E\{d^2[n]\} - 2.w^T.p + w^T.R.w \qquad (120)$$

$$\boxed{J = \sigma^2 - 2.w^T.p + w^T.R.w} \qquad (121)$$

Se desea obtener el valor del peso ideal, es decir el valor de w que minimice el error J. Para el caso de un coeficiente, se calcula la derivada de J y se iguala a 0.

$$\nabla J_w = \frac{\partial J}{dw} = -2.p + 2.R.w \qquad (122)$$

$$\frac{\partial J}{dw} = -2.p + 2.R.w = 0 \qquad (123)$$

$$R.w = p \qquad (124)$$

$$w_o = w_{optima} = R^{-1}.p \qquad (125)$$

Se obtuvo w_{optima} que es la solución óptima de Wiener

En la Fig. 140 se muestra un ejemplo del error cuadrático medio J para al algoritmo de pasos descendentes con 1 coeficiente. Notar que para $\frac{dJ}{dw} > 0$ el coeficiente decrece y viceversa

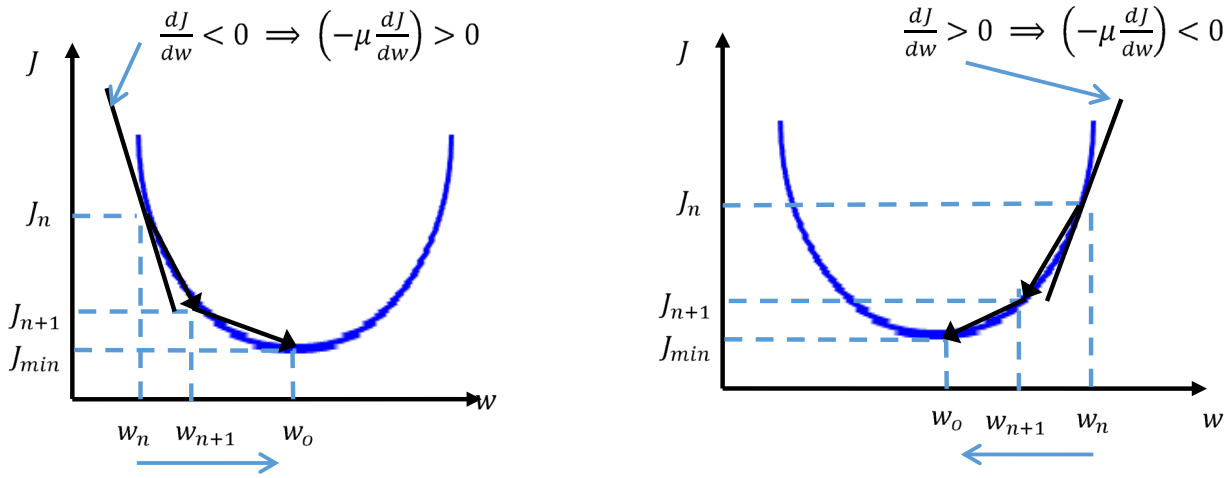


Fig. 140 – Ejemplo de error cuadrático medio J para al algoritmo de pasos descendentes con un coeficiente. Izquierda) $\frac{dJ}{dw} < 0$ *y derecha)* $\frac{dJ}{dw} > 0$

Los pesos se actualizan en cada iteración, el nuevo w, es decir $w[n+1]$ debe ser el w anterior: $w[n]$ al cual se le resta/suma un porcentaje de $\hat{g}_w(n)$ que lleve a la función al valor mínimo de J. Siendo μ una constante que modifica la velocidad de convergencia del algoritmo

$$w[n+1] = w[n] - \mu.\hat{g}_w[n] \qquad (126)$$

$$\hat{g}_w[n] = \hat{\nabla} J_w$$

Teníamos: $\nabla \mathrm{J}_w = \frac{\partial J}{dw} = -2.p + 2.R.w$

Ahora $\hat{g}_w[n]$ es una estimación del gradiente

$$\hat{g}_w[n] = \hat{\nabla} J_w = -2.\hat{p}[n] + 2.\hat{R}[n].w[n] \qquad (127)$$

Donde $\hat{R}$ indica una buena estimación de R y $\hat{p}$ indica una buena estimación de p

Reemplazando ecuación (127) en la ecuación (126) se obtiene:

$$w[n+1] = w[n] + 2.\mu.\left(\hat{p}[n] - \hat{R}[n].w[n]\right)$$

Una posible solución es tomar las siguientes estimaciones:

$$\hat{p}[n] = d[n].x[n]$$

$$\hat{R}[n] = x[n].x^T[n]$$

Recordando los valores de p y R:

$p = E\{d[n].x[n]\}$: correlación cruzada entre $d[n]$ y $x[n]$

$R = E\{x[n].x^T[n]\}$: autocorrelación de la entrada $x[n]$

Entonces el gradiente estimado queda:

$$\hat{g}_w[n] = -2.\hat{p}[n] + 2.\hat{R}[n].w[n]$$

$$\hat{g}_w[n] = -2.d[n].x[n] + 2.x[n].x^T[n].w[n]$$

$$\hat{g}_w[n] = 2.x[n](-d[n] + x^T[n].w[n])$$

Reemplazando: $y[n] = w^T[n].x[n]$; $e[n] = d[n] - y[n]$, se obtiene:

$$\hat{g}_w[n] = -2.e[n].x[n]$$

Reemplazando en: $w[n+1] = w[n] - \mu.\hat{g}_w[n]$, obtenemos la siguiente solución conocida como LMS (Least Mean Square):

$$\boxed{w[n+1] = w[n] + \mu.2.e[n].x[n]} \qquad (128)$$

Recordar que el filtro tiene $N+1$ coeficientes

$y[n] = \sum_{i=0}^{N} w_i[n].x[n-i] = w^T[n].x[n]$

Siendo $x[n] = [x[n],\ x[n-1], \dots \dots . x[n-N]]^T$

$w[n] = [w_0[n],\ w_1[n], \dots \dots . w_N[n]]^T$

Si la función objetivo $J = E\{e^2[n]\}$ es reemplazada por el valor instantáneo del error cuadrático medio $e^2[n]$ se tiene el siguiente vector:

$$\hat{g}_w[n] = \frac{\partial e^2[n]}{\partial w} = \left[2.e[n].\frac{\partial e[n]}{\partial w_o[n]},\quad 2.e[n].\frac{\partial e[n]}{\partial w_1[n]},\quad \dots \dots, 2.e[n].\frac{\partial e[n]}{\partial w_N[n]}\right]^T$$

En la Fig. 141 se muestra un esquema del sistema adaptativo, detallando el filtro FIR

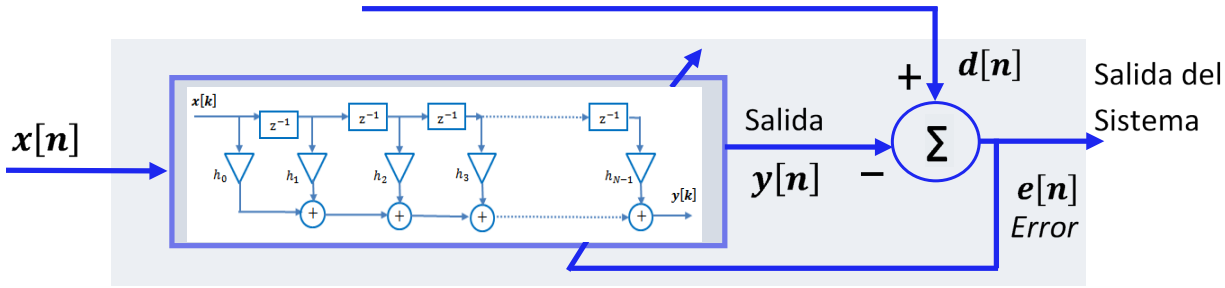


Fig. 141 – Sistema adaptativo con filtro FIR.

Otra nomenclatura

Partiendo de la ecuación utilizada en LMS para un filtro FIR se reescribe la ecuación de la siguiente forma:

$w_{n+1} = w_n + \mu.2.e[n].x[n]$

Para un filtro FIR de orden n, se extiende el algoritmo LMS de la siguiente forma:

$w_{n+1}[i] = w_n[i] + \mu.2.e[n].x[n-i]$

Siendo $i = 0,1, \dots \dots, N$; varios pesos

$$y[n] = w_n[0].x[n] + w_n[1].x[n-1] + \cdots \dots . + w_n[N-1].x[n-N+1]$$

Nota: Para hallar la última ecuación se tiene en cuenta la respuesta del filtro FIR: $y[n] = \sum_{k=0}^{N} b_k . x[n-k]$, pero para sistemas adaptativos se utilizan los coeficientes $w_n[i]$ en vez de b_k. Así se expresa que w_n varía en función de las iteraciones del algoritmo.

Factor de Convergencia LMS

Para un filtro FIR de orden n, se extiende el algoritmo LMS de la siguiente forma:

$w_{n+1}[i] = w_n[i] + \mu . 2 . e[n] . x[n-i]$

Siendo $i = 0,1, \dots \dots, N-1$;

El factor de convergencia μ debe cumplir:

$\boxed{0 < \mu < \frac{1}{N.P_x}}$; siendo P_x la potencia de la señal de entrada.

En la práctica, si tenemos un ADC que tiene datos de 16 bits, la amplitud máxima de la señal debe ser $A = 2^{15}$. Entonces la potencia de entrada máxima debe ser menor que:

$$P \leq (2^{15})^2 = 2^{30}$$

Por lo tanto, podemos hacer una selección del parámetro de convergencia como:

$0 < \mu < \frac{1}{N.2^{30}} \quad \Longrightarrow \quad 0 < \mu < \frac{9{,}3 . \ 10^{-10}}{N}$

Algoritmo LMS

Los pasos básicos para implementar un algoritmo LMS de un sistema adaptativo son:

a) Inicializar los pesos del filtro: $w(0), w(1), \dots \dots w(N-1)$, por ejemplo, con unos, o con valores arbitrarios
b) Asignar un valor adecuado de μ
c) Leer $d[n]$ y $x[n]$, y calcular la salida del filtro: $y[n]$
$$y[n] = w(0).x(n) + w(1).x(n-1) + \cdots \dots . . + w(N-1).x(n-N+1)$$
d) Calcular el error $e[n]$:
$$e[n] = d[n] - y[n]$$
e) Actualizar los nuevos coeficientes del filtro $w[n+1]$ con los valores $w[n]$ anteriores, el error y los datos de entrada:
Para: $i = 0,1, \dots \dots, N-1$
$$w[i] = w[i] + \mu . 2 . e[n] . x[n-i]$$
f) Repetir algunas veces el algoritmo desde el punto c) para minimizar el error y ajustar los coeficientes

Partiendo de la ecuación:

$w[n+1] = w[n] + \mu . 2 . e[n] . x[n]$

Podemos reescribir la ecuación en forma detallada indicando los coeficientes del filtro:

$w_i[n+1] = w_i[n] + \mu . 2 . e[n] . x[n]$

A partir de esta última ecuación y teniendo en cuenta la Fig. 68, se obtiene la Fig. 142 correspondiente a un filtro FIR con algoritmos LMS

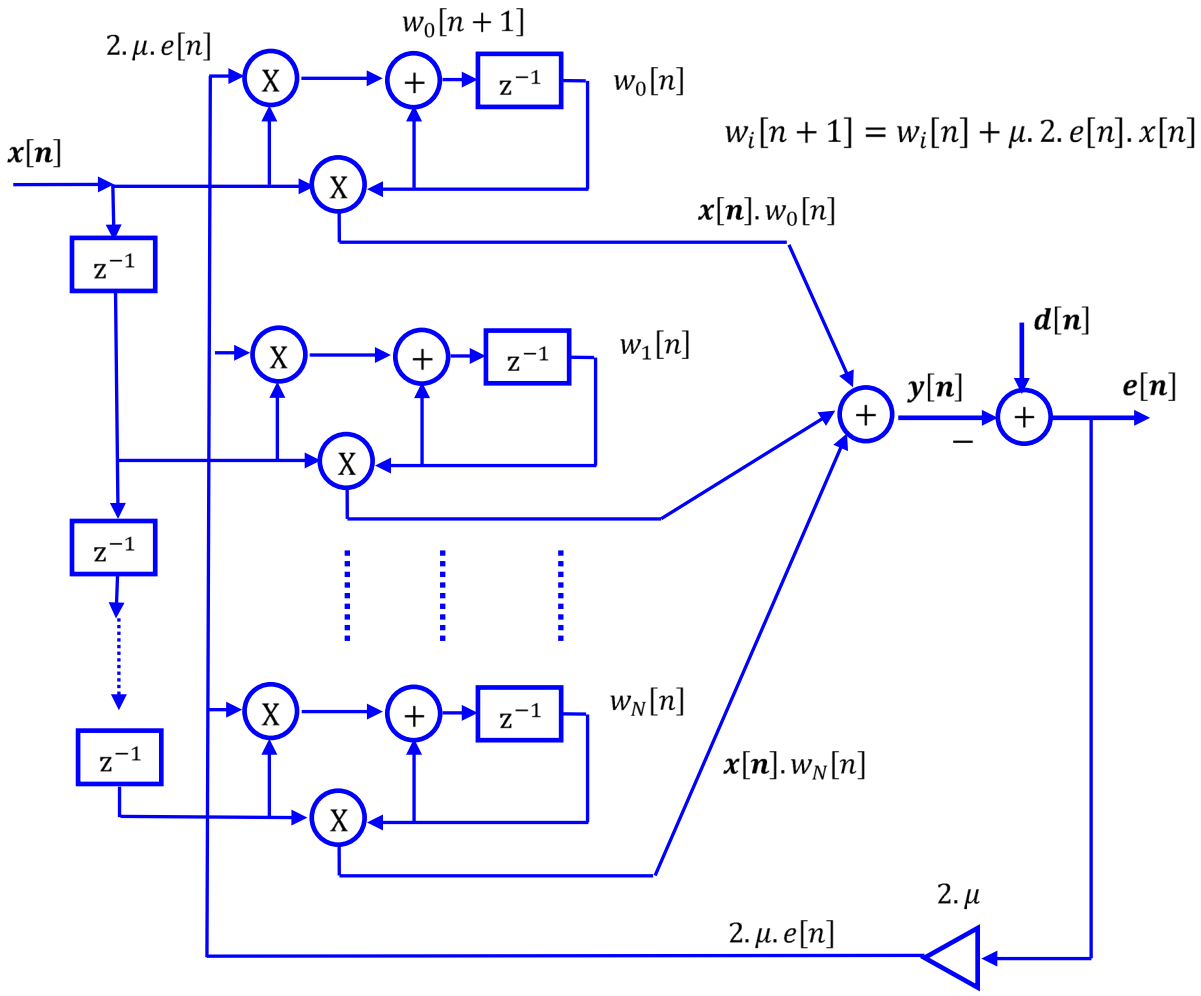


Fig. 142 – Filtro Adaptativo LMS FIR

Filtro de Wiener

Dado un conjunto L de coeficientes correspondientes a las Respuesta de un filtro FIR:

$$w[n] = \{w_0[n], \quad w_1[n], \dots\dots, w_{L-1}[n]\}^T \tag{129}$$

Siendo el conjunto de señales de entrada:

$$x[n] = \{x[n], \quad x[n-1], \dots\dots, x[n-L+1]\}^T \tag{130}$$

La Salida del filtro se calcula con la siguiente expresión:

$$y[n] = \sum_{i=0}^{L-1} w_i[n].x[n-i] \tag{131}$$

Expresado en vectores:

$$y[n] = w^T[n].x[n] = x^T[n].w[n] \tag{132}$$

Generalmente la salida $y[n]$ obtenida en esta última expresión se compara con la señal deseada $d[n]$. Donde la señal de error es la resta de ambas señales. Junto con esta definición de $e[n]$ y la ecuación (132) obtenemos:

$$e[n] = d[n] - y[n] = d[n] - w^T[n].x[n] \quad (133)$$

Mediante los algoritmos adaptativos, se modifican los coeficientes del filtro para minimizar este error residual

Filtros de Wiener

El filtro de Wiener es un filtro propuesto en la década de 1940 por Norbet Wiener. Mediante métodos estadísticos reduce lo más posible el error cuadrático medio (MSE) entre la señal deseada del filtro y la señal de entrada. Kolmogorov en forma independiente publicó en 1941 resultados equivalentes en tiempo discreto. Por esto, la teoría es a veces se denomina teoría de filtrado de Wiener-Kolmogorov.

Se buscan los valores óptimos de los coeficientes mediante la ecuación de Wiener-Hopf.

Este método presenta la desventaja que se deben conocer las propiedades estadísticas de la señal deseada y sus relaciones con la señal de entrada. Los filtros de Wiener se puede decir que no aprenden.

Algoritmo de error cuadrático medio (MSE)

En este algoritmo las señales $x[n]$ y $d[n]$ tienen que ser estacionarias. Para reducir el error $e[n]$ mediante un algoritmo adaptativo hay que ajustar en forma progresiva los coeficientes o pesos del filtro. Para aplicar estos algoritmos hay que tener en cuenta la superficie de *error cuadrático medio* (MSE), esto resulta muy importante en los algoritmos de adaptación, donde los coeficientes del filtro dependen de esta MSE.

El MSE está dado por la siguiente ecuación:

$$J = \xi[n] = E\{|e[n]|^2\} \quad (134)$$

Para un primer análisis simplificado, que se cumple bajos ciertas condiciones, se utiliza la siguiente expresión:

$$\xi[n] = E\{e^2[n]\} \quad (135)$$

$E\{\ \ \}$ representa la esperanza matemática o valor esperado

Si sustituimos la ecuación (133) en (135)

$$\xi[n] = E\{e^2[n]\} = E\{\ (d[n] - w^T[n].x[n]).(d[n] - w^T[n].x[n])\ \ \} \quad (136)$$

$$\xi[n] = E\{\ d^2[n]\ - 2.d[n].w^T[n].x[n]\ \ + w^T[n].x[n].w^T[n].x[n]\ \} \quad (137)$$

$$\xi[n] = E\{e^2[n]\} = E\{\ d^2[n]\ \} - 2.w^T[n].E\{d[n].x[n]\} + w^T[n].E\{x[n].x[n]\ \}.w[n] \quad (138)$$

Reemplazando p y R en la ecuación anterior, se obtiene:

$$\xi[n] = E\{e^2[n]\} = E\{\ d^2[n]\ - 2.p^T.w[n]\ + w^T[n].R.w[n]\ \}$$

$$\boxed{\xi[n] = E\{e^2[n]\} = E\{\ d^2[n]\ \}\ - 2.p^T.w[n]\ + w^T[n].R.w[n]} \quad (140)$$

Siendo p la función de correlación entre $d[n]$ y $x[n]$:

$$\boxed{p = E\{\ d[n].x[n]\ \} = [r_{dx}(0),\ r_{dx}(1),\ \ \ \ \ \ \ \ \ldots\ldots, r_{dx}(L-1)]^T} \quad (141)$$

Con $r_{dx}(k) = E\{\ d[n].x[n-k]\ \}$ (142)

Por otro lado R es la matriz de autocorrelación de $x[n]$:

$$R = E\{\ x[n].x^T[n]\ \} \quad (143)$$

$$R = \begin{bmatrix} r_{xx}(0) & r_{xx}(1) & r_{xx}(L-1) \\ r_{xx}(1) & r_{xx}(0) & r_{xx}(L-2) \\ \ldots. & & \ldots\ldots\ldots\ldots \\ r_{xx}(L-1) & r_{xx}(L-2) & r_{xx}(0) \end{bmatrix} \quad (144)$$

Con $r_{xx}(k) = E\{\ x[n].x[n-k]\ \}$ (145)

El filtro óptimo $w^o[n]$ minimiza la función $\xi[n]$

Para obtener dicha función, calculamos el gradiente respecto de $w[n]$ e igualamos a 0

Utilizamos la ecuación (140): $\xi[n] = E\{e^2[n]\} = E\{\ d^2[n]\ \}\ - 2.p^T.w[n]\ + w^T[n].R.w[n]$

$$\nabla\xi[n] = -2.p\ + 2.w[n] = 0 \quad (146)$$

$$\boxed{R.w^o[n] = p} \quad (147)$$

Obtenemos la ecuación de Ecuación de Wiener-Hopf para el cálculo de los valores óptimos de los coeficientes:

Ecuación de Wiener-Hopf: $\boxed{w^o[n] = p.R^{-1}}$ (148)

Se tiene una solución para el filtrado adaptativo, sin embargo, los vectores y matrices de correlación se deben calcular constantemente. Teniendo en cuenta que la señales pueden ser no estacionarias, se necesita mucha carga computacional para calcular las correlaciones R y p, si la matriz R es grande se necesitan realizar muchos cálculos.

Cálculo del valor mínimo de MSE:

Partiendo de la ecuación (140): $\xi[n] = E\{e^2[n]\} = E\{\ d^2[n]\ \}\ - 2.p^T.w[n]\ + w^T[n].R.w[n]$

Reemplazamos el valor óptimo de los coeficientes $w^o[n]$, para obtener el mínimo MSE:

$\xi_{min} = E\{\ d^2[n]\ \}\ - 2.p^T.w^o[n]\ + w^{oT}[n].R.w^o[n]$; reemplazamos $R.w^o[n] = p$

$\xi_{min} = E\{\ d^2[n]\ \}\ - 2.p^T.w^o[n]\ + w^{oT}[n].p$

Pues el tamaño de $w^{oT}[n].p$ en el segundo miembro de la ecuación anterior es de 1x1 (es decir es un número) con lo cual es igual a su traspuesta (para las matrices A de 1x1 vale $A = A^T$)

Luego $w^{oT}[n].p$=$(w^{oT}[n].p)^T$=$p^T(w^{oT}[n])^T = p^T w^o[n]$ donde se usó que $(A.B)^T = B^T.A^T$ y $A^{T^T} = A$

Es decir: $w^{oT}[n].p = p^T w^o[n]$, y con este último resultado reemplazamos en ξ_{min}.

Obtenemos el mínimo MSE:

$$\boxed{\xi_{min} = E\{\ d^2[n]\ \} - p^T.w^o[n]} \quad (149)$$

Partiendo de la ecuación (140): $\xi[n] = E\{\ d^2[n]\ \} - 2.p^T.w[n] + w^T[n].R.w[n]$

$\xi_{min} = E\{\ d^2[n]\ \} - p^T.w^o[n]$; entonces: $E\{\ d^2[n]\ \} = \xi_{min} + p^T.w^o[n]$; Reemplazamos en (6)

$\xi[n] = \xi_{min} + p^T.w^o[n] - 2.p^T.w[n] + w^T[n].R.w[n]$;

Siendo $\boxed{p = R.w^o[n]}$; Donde se usó que $p^T = (R.w^o[n])^T = w^{oT}[n].R^T$; Reemplazamos $p^T = w^{oT}[n].R^T$

$\xi[n] = \xi_{min} + w^{oT}[n].R^T.w^o[n] - 2.w^{oT}[n].R^T.w[n] + w^T[n].R.w[n]$

Como R es una matriz simétrica, se cumple: $R = R^T$

$\xi[n] = \xi_{min} + w^{oT}[n].R.w^o[n] - 2.w^{oT}[n].R.w[n] + w^T[n].R.w[n]$

$\xi[n] = \xi_{min} + w^{oT}[n].R.w^o[n] - w^{oT}[n].R.w[n] - w^{oT}[n].R.w[n] + w^T[n].R.w[n]$ (*)

Tomamos este miembro:

$w^{oT}[n].R.w[n] = w^{oT}[n].R^T.(w^T[n])^T = w^{oT}[n].(w^T[n].R)^T = (w^T[n].R.w^o[n])^T = w^T[n].R.w^o[n]$

$(A.B)^T = B^T.A^T$; Se toma: $(w^T[n].R.w^o[n])^T = w^T[n].R.w^o[n]$ por ser un número 1x1

$w^{oT}[n].R.w[n] = w^T[n].R.w^o[n]$ reemplazamos en (*)

$\xi[n] = \xi_{min} + w^{oT}[n].R.w^o[n] - w^{oT}[n].R.w[n] - w^T[n].R.w^o[n] + w^T[n].R.w[n]$

Sacando factor común $R.w[n]$ y $R.w^o[n]$

$\xi[n] = \xi_{min} + (w^{oT}[n] - w^T[n]).R.w^o[n] + (-w^{oT}[n] + w^T[n]).R.w[n]$

$\xi[n] = \xi_{min} + (w^{oT}[n] - w^T[n]).R.w^o[n] - (w^{oT}[n] - w^T[n]).R.w[n]$

$\xi[n] = \xi_{min} + (w^{oT}[n] - w^T[n]).(R.w^o[n] - R.w[n])$; sacamos factor común R, multiplicamos por (-1).(-1)

$\xi[n] = \xi_{min} + (w^T[n] - w^{oT}[n]).R.(w[n] - w^o[n])$; $A^T - B^T = (A - B)^T$

$$\xi[n] = \xi_{min} + \{w[n] - w^o[n]\}^T.R.\{w[n] - w^o[n]\} \quad (150)$$

Siendo $v[n]$ el vector de desviación de coeficientes se calcula como la diferencia entre los coeficientes del filtro adaptativo y la solución óptima:

$$\boxed{v[n] = w[n] - w^o[n]} \quad (151)$$

$$\boxed{\xi[n] = \xi_{min} + v^T[n].R.v[n]} \quad (152)$$

Para cada valor del vector de coeficientes $w[n]$ existe un valor escalar correspondiente del MSE, estos valores asociados con $w[n]$ forman un espacio de dimensión L+1 denominado superficie de desempeño.

En la Fig. 143 se muestra la superficie de desempeño para L=2

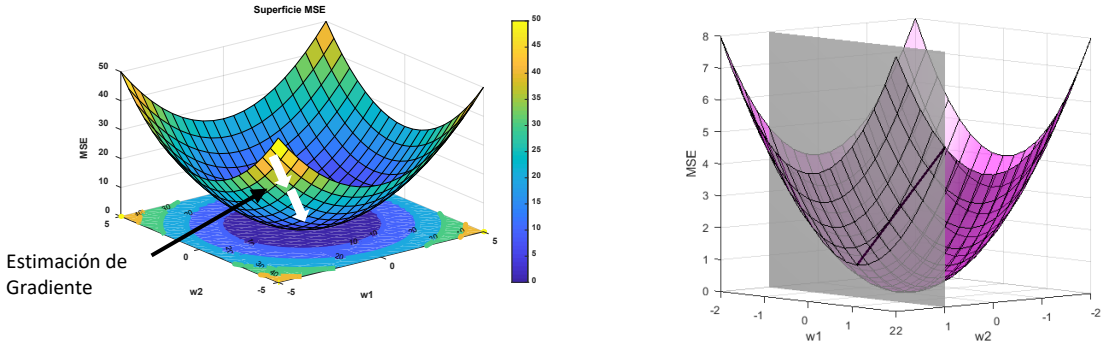


Fig. 143 - Superficie de desempeño - MSE para el caso L = 2. Izquierda: grafica en colores, Derecha: corte vertical

La figura de ejemplo corresponde a la función $MSE = f(x, y) = x^2 + y^2$ se obtiene con el siguiente código en Octave / Matlab®:

```
vector_x = [-2:  0.25  :2];
vector_y = vector_x;
[x, y] = meshgrid(vector_x, vector_x);
z = x.^2 + y.^2;
surfc(x, y, z)
xlabel('w1'); ylabel('w2')
zlabel('MSE'); title('Superficie MSE')
colorbar
```

En la Fig. 144 se muestra un filtro FIR utilizado en filtro de Wiener

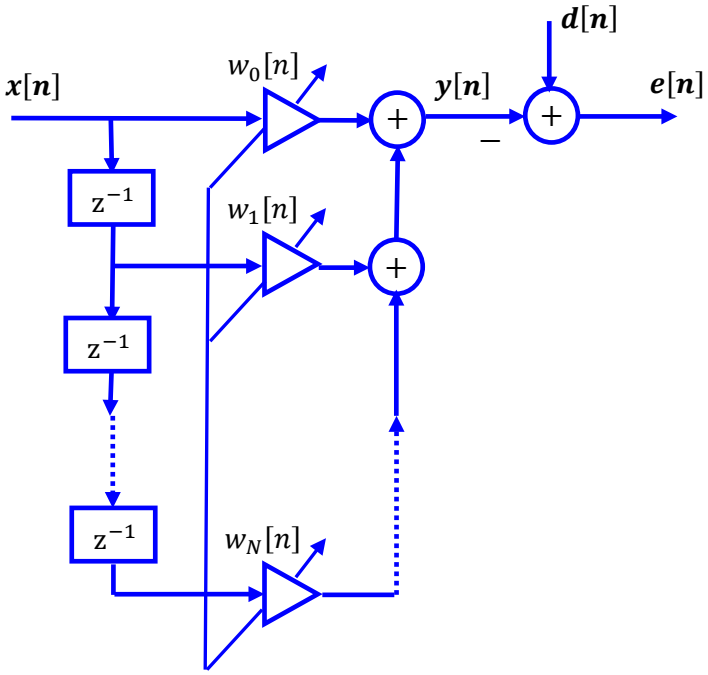


Fig. 144 - Filtro FIR utilizado en filtro de Wiener

Algoritmo del descenso más pronunciado

El algoritmo del descenso más pronunciado se basa en el método de pasos descendentes (MSD: method of steepest descend). MSD analiza el descenso más pronunciado. En la Fig. 143 se muestra un ejemplo de error cuadrático medio (MSE), la función de error resulta cuadrática respecto de los coeficientes w. Se realiza un ajuste de coeficientes en varios pasos para minimizar el error, en la figura se observa una especie de "tazón", en cada paso se desciende en la superficie del tazón hasta llegar al error mínimo ξ_{min} que corresponde a los valores óptimos de $w[n]$, denominados $w^o[n]$

Mediante MSD se eligen valores iniciales arbitrarios de $w[n]$ que corresponde a un error inicial $\xi[0]$ correspondiente al tiempo inicial $n = 0$. Posteriormente se desplaza en pasos por la superficie en dirección tangente a sus puntos, se desciende al fondo del tazón hasta llegar al error mínimo ξ_{min}. Los coeficientes del filtro se actualizan en dirección negativa al gradiente de la superficie de error. Para cada punto de la superficie se tiene una configuración de filtro y un error dado por el conjunto $\{w[n],\ \xi[n]\}$. Para cada punto de la superficie, existen derivadas direccionales que tienden al error mínimo. Es decir, existen pendientes a la superficie correspondientes a cada eje w_i, siendo w_i las componentes de $w[n]$. Estas pendientes están definidas por las derivadas direccionales $\partial\xi[n]/\partial w_i$. El vector de estas derivadas direccionales es el gradiente de la superficie de error $\nabla\xi[n]$.

Siendo $w[n+1]$ el valor posterior a $w[n]$, el descenso más corto se expresa de la siguiente manera:

$$\boxed{w[n+1] = w[n] - \frac{\mu}{2}.\nabla\xi[n]} \tag{153}$$

El vector $\nabla\xi[n]$ corresponde al gradiente de error y el signo negativo orienta al vector de coeficientes $w[n]$ en dirección negativa al gradiente. El factor de convergencia es μ, indica la velocidad de descenso en la curva y controla la estabilidad, mientras más grande desciende más rápido

Partiendo de la ecuación (140): $\xi[n] = E\{\ d^2[n]\ \} \ - 2.p^T.w[n] \ + w^T[n].R.w[n]$

Siendo su gradiente $\nabla\xi[n] = -2.p + 2.R.w[n]$, reemplazamos en ecuación (153), obtenemos la expresión del algoritmo de pasos descendentes:

$$\boxed{w[n+1] = w[n] + \mu.(p - R.w[n])} \tag{154}$$

El vector $\nabla\xi[n]$ vale cero cuando $w[n]$ llega al valor óptimo $w^o[n]$

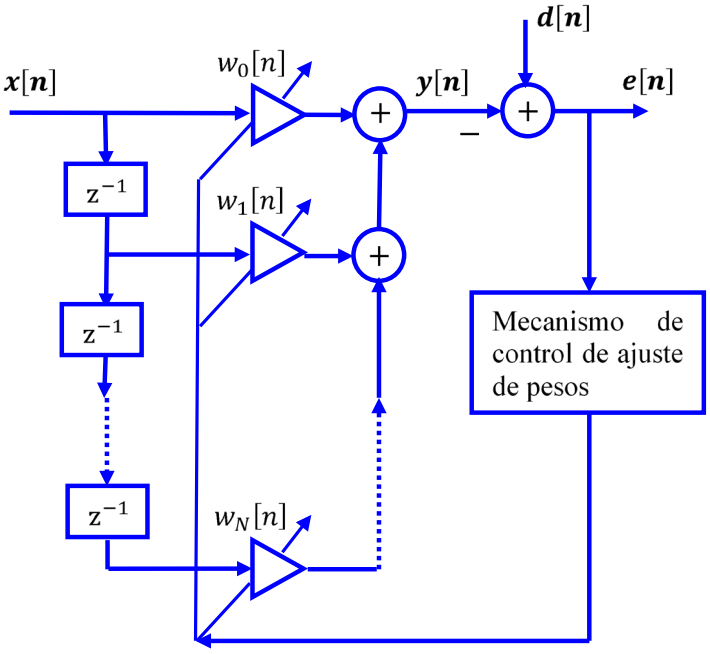


Fig. 145 - Estructura de un Filtro Adaptativo tipo FIR

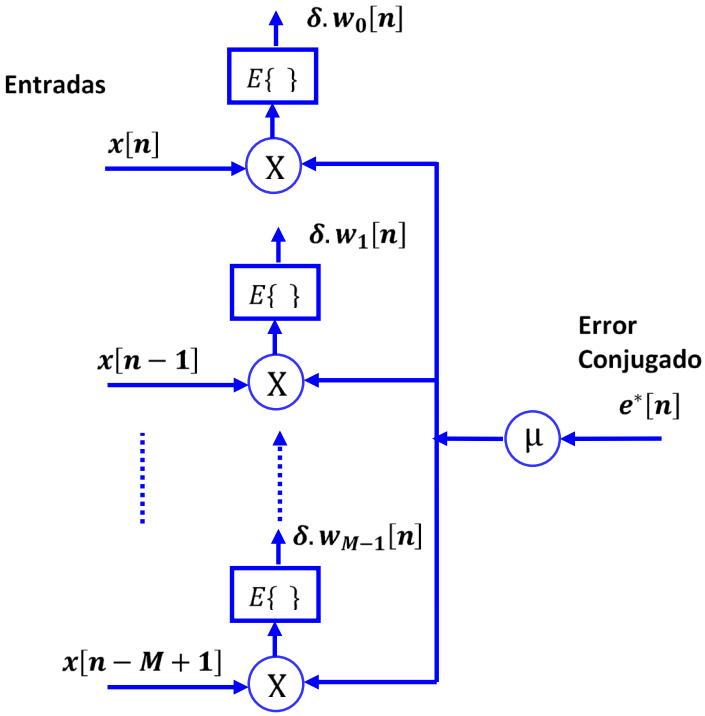


Fig. 146 - Banco de bloques de correlación para calcular la corrección de los pesos del filtro para un determinado tiempo n

La siguiente ecuación describe la formulación matemática del algoritmo de descenso más pronunciado aplicado al filtrado de Wiener.

$w[n+1] = w[n] + \mu.(p - R.w[n])$ con $n = 0,1,2$ ....

Estabilidad del Algoritmo de descenso más pronunciado

El algoritmo del descenso más pronunciado (o pasos descendentes) utiliza realimentación. Esta realimentación puede generar inestabilidad en el sistema. Esta inestabilidad está determinada por 2 factores: a) Factor μ que determina el tamaño del paso y b) la matriz de autocorrelación de la entrada denominada $\boldsymbol{R}$.

Algoritmo LMS

En 1960 Bernard Widrow y Edward Hopf desarrollaron el algoritmo LMS (Least Mean Squares Algorithm) basado en el método de pasos descendentes, para estimar una función variable en el tiempo. Con este método se utiliza en forma iterativa la dirección descendente del gradiente para reducir el error cuadrático medio (MSE) y obtener los coeficientes $w[n]$ más adecuados del filtro adptativo. En la ecuación (153) se observa que se realiza una suma en dirección contraria al gradiente, los coeficientes seguirán un camino de mayor descenso a través de la superficie de error. En muchas aplicaciones se desconocen las estadísticas de $x[n]$ o $d[n]$, por lo cual el método de pasos descendentes no se puede utilizar en forma directa, ya que se necesita conocer el gradiente en cada operación. Para solucionar este problema, Widrow propuso usar el error cuadrático instantáneo $e^2[n]$, para obtener el error cuadrático medio mediante la siguiente ecuación:

$$\hat{\xi}[n] = e^2[n] \tag{155}$$

Es decir que se utiliza esta ecuación, en vez de la ecuación (140): $\xi[n] = E\{e^2[n]\}$

Tomado la ecuación (155), la estimación del gradiente para el algoritmo LMS es el gradiente instantáneo de una muestra del error cuadrático:

$$\nabla\hat{\xi}[n] = 2.\nabla\{e[n]\}.e[n] \tag{156}$$

Utilizando la ecuación (133): $e[n] = d[n] - w^T[n].x[n]$, el gradiente resulta:

$$\nabla e[n] = -x[n] \tag{157}$$

Reemplazando en ecuación anterior se obtiene la estimación del gradiente:

$$\boxed{\nabla\hat{\xi}[n] = -2.x[n].e[n]} \tag{158}$$

Reemplazando en ecuación (153): $w[n+1] = w[n] - \frac{\mu}{2}.\nabla\xi[n]$, donde utilizamos $\nabla\hat{\xi}[n]$ en vez de $\nabla\xi[n]$

$$\boxed{w[n+1] = w[n] + \mu.x[n].e[n]} \tag{159}$$

Características del Algoritmo LMS

Este algoritmo es una implementación simple del método de pasos descendentes (MSD), su complejidad computacional escala linealmente con la dimensión del filtro de respuesta finita FIR.

A diferencia del filtro de Wiener, no se requiere conocimiento de las estadísticas del entorno en que trabaja. Resulta robusto ante perturbaciones ambientales desconocidas

Otra característica importante es que no requiere invertir la matriz de correlación del regresor (por ejemplo, de la entrada), lo que simplifica las cuentas respecto del algoritmo RLS.

Debido a todas estas características, este algoritmo es uno de los más utilizados en filtros adaptativos.

Estructura del Algoritmo LMS

En la Fig. 147 se muestra un diagrama general del Algoritmo LMS. Se observan 3 componentes:

a) Filtro FIR que opera, en este caso sobre la entrada $x[n]$, para generar una estimación de la respuesta deseada $\hat{d}[n]$. Este filtro utiliza los coeficientes $w[n]$ que se actualizan en forma iterativa.
b) Un comparador, que realiza la resta entre respuesta deseada $d[n]$ y la estimación de la respuesta deseada $\hat{d}[n]$, generando la función de error o también llamada señal de error $e[n]$.
c) Un mecanismo de control de pesos adaptativo que ajusta cada uno de los valores $w[n]$ utilizados en el filtro FIR. Realiza los cálculos en base al vector de pesos anterior $w[n-1]$, la señal de error $e[n]$ y la entrada $x[n]$

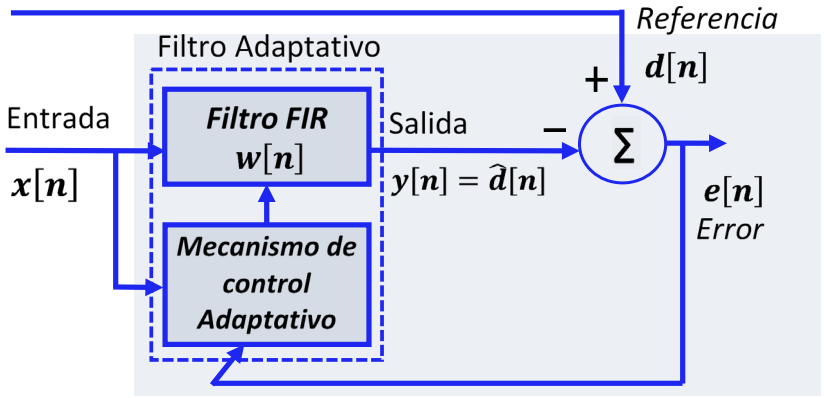


Fig. 147 – Esquema general del Algoritmo LMS con filtro FIR

El error cuadrático medio (MSE) del método de descenso más pronunciado se obtiene con la siguiente ecuación:

$$J = \xi[n] = E\{|e[n]|^2\} \tag{160}$$

Al comparar el mecanismo de control de la Fig. 148 para el algoritmo LMS con respecto al método de descenso más pronunciado de la Fig. 146, vemos que el algoritmo LMS usa el producto $x[n-k].e^*[k]$ como una estimación del elemento k en el vector gradiente $\nabla J[n]$ que caracteriza el método de descenso más pronunciado. Es decir que los bloques de Esperanza (o expectativa) se eliminan por

completo. Por consiguiente, el cálculo recursivo de cada peso de derivación en el algoritmo LMS sufre de ruido de gradiente.

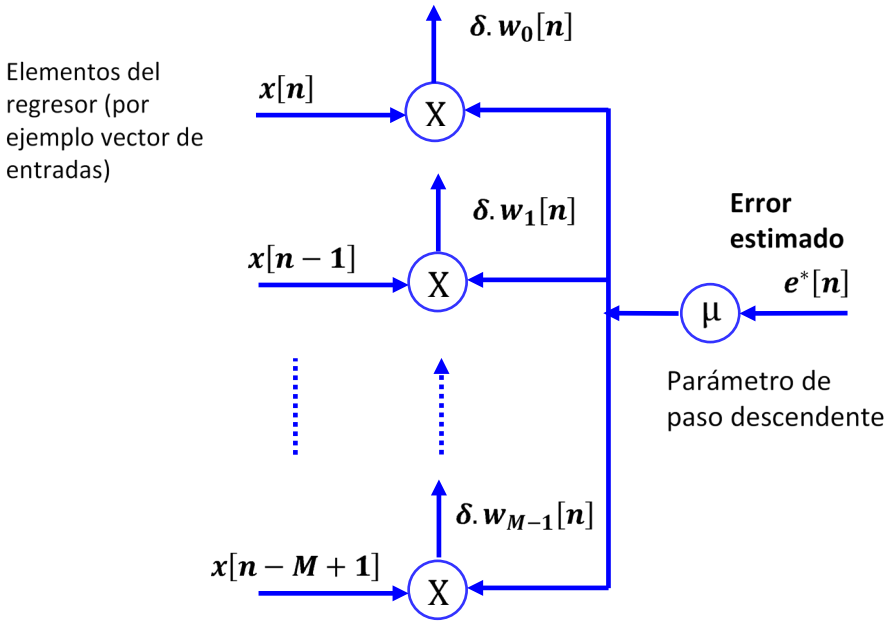


Fig. 148 – Detalles del Mecanismo de control de pesos adaptativos

Algoritmos Basados en LMS

Existe una serie de algoritmos alternativos basados en LMS. Estos algoritmos tienen como objetivo reducir la complejidad computacional y/o el tiempo de convergencia. A continuación, se muestran los algoritmos más utilizados:

LMS Básico

Algoritmo de Error cuantificado

Algoritmo LMS–Newton

Algoritmo LMS Normalizado

Algoritmo en Dominio de frecuencia (o dominio transformado) LMS

Algoritmo de proyección afín

Los algoritmos de error cuantificado (Quantized-error algorithms) reducen la complejidad computacional de los algoritmos LMS al representar la señal de error con una longitud de palabra corta o con un simple número de potencia de dos. Siendo $\mathbb{Q}$ una operación de cuantificación, se tiene la siguiente expresión:

$$w[k+1] = w[k] + 2.\mu.\mathbb{Q}(e[k]).x[k]$$

A modo de ejemplo la cuantificación puede estar dada por la función signo:

$$sgn(a) = \begin{cases} 1, & a > 0 \\ 0, & a = 0 \\ -1, & a < 0 \end{cases}$$

$$w[k+1] = w[k] + 2.\mu.sgn(e[k]).x[k] \quad \text{Algoritmo signo del Error}$$

$$w[k+1] = w[k] + 2.\mu.e[k].sgn(x[k]) \quad \text{Algoritmo signo de los datos}$$

El algoritmo LMS-Newton incorpora estimaciones de las estadísticas de segundo orden de las señales. El objetivo del algoritmo es evitar la convergencia lenta del algoritmo LMS en los casos de tener señal de entrada altamente correlacionada. La mejora en la tasa de convergencia se logra a expensas de una mayor complejidad computacional. La realización no recursiva del filtro adaptativo conduce a una superficie MSE que es una función cuadrática de los coeficientes del filtro. En la estructura FIR de forma directa, el MSE se puede describir mediante:

$$\varepsilon[k+1] = \varepsilon[k] + {g_w}^T[k].(w[k+1] - w[k]) + (w[k+1] - w[k])^T.\mathrm{R}.(w[k+1] - w[k])$$

La velocidad de convergencia en el algoritmo LMS-Newton es independiente de la dispersión de los autovalores de la matriz de correlación de las señales de entrada. Esta mejora se logra mediante el uso de una estimación de la inversa de la matriz de correlación de señales de cinco entradas, lo que conduce a un aumento sustancial de la complejidad computacional.

$$w[k+1] = w[k] + 2.\mu.e[k].\hat{R}^{-1}[k].x[k]$$

El algoritmo LMS normalizado contiene un factor de convergencia variable que minimiza el error instantáneo. Tal factor de convergencia generalmente reduce el tiempo de convergencia, pero aumenta el desajuste.

$$w[k+1] = w[k] + 2.\mu_k.e[k].x[k] = w[k] + \Delta\hat{w}[k]$$

En el algoritmo de Transformada del dominio (Frequency-domain or transform-domain), se aplica una transformada a la señal de entrada. Esto permite reducir la dispersión de los autovalores de la matriz de correlación de la señal transformada en comparación con la dispersión de los autovalores de la matriz de correlación de la señal de entrada. El algoritmo LMS aplicado a la señal transformada mejor acondicionada logra una convergencia más rápida.

Se aplica una Transformación ortonormal:

$$s[k] = T.x[k]$$

$$T.T^T = I$$

El algoritmo de proyección afín reutiliza datos antiguos, lo que da como resultado una convergencia rápida cuando la señal de entrada está altamente correlacionada, lo que lleva a una familia de algoritmos que pueden compensar la complejidad computacional.

Análisis de la estabilidad y performance del LMS.

Suponemos que un filtro desconocido del tipo FIR con coeficientes w_0 es identificado por un sistema adaptativo con algoritmo LMS con filtro FIR de mismo orden. A la salida del sistema desconocido tenemos ruido blanco $n[k]$ agregado con media cero y varianza σ_n^2. El error en los coeficientes del filtro adaptativo en relación con el vector de coeficiente ideal w_0, en cada iteración, se describe mediante un vector de longitud $N+1$ dado por:

$$\Delta w[k] = w[k] - w_0$$

Mediante esta ecuación, usando $\Delta w[k]$, el algoritmo LMS se puede escribir en forma alternativa de la siguiente forma:

$$\Delta w[k+1] = \Delta w[k] + 2.\mu.e[k].x[k]$$

Reemplazamos $e[k] = x^T[k].w_0 + n[k] - x^T[k].w[k]$

$$\Delta w[k+1] = \Delta w[k] + 2.\mu.x[k].\ \{x^T[k].w_0 + n[k] - x^T[k].w[k]\}$$

$$\Delta w[k+1] = \Delta w[k] + 2.\mu.x[k].\ \{n[k] + x^T[k].w_0 - x^T[k].w[k]\}$$

Reemplazamos: $e_0[k] = n[k]\ ;\quad \Delta w[k] = w[k] - w_0$

$$\Delta w[k+1] = \Delta w[k] + 2.\mu.x[k].\ \{e_0[k] - x^T[k].\Delta w[k]\}$$

$$\Delta w[k+1] = \Delta w[k] - 2.\mu.x[k].x^T[k].\Delta w[k] +\ \ 2.\mu..e_0[k].x[k]$$

$$\Delta w[k+1] = \{I - 2.\mu.x[k].x^T[k]\}.\Delta w[k] +\ \ 2.\mu.e_0[k].x[k]$$

Donde $e_0[k]$ es el error óptimo dado por:

$$e_0[k] = d[k] - w_0^T.x[k]$$

$$e_0[k] = w_0^T.x[k] + n[k] - w_0^T.x[k]$$

$$e_0[k] = n[k]$$

Aplicamos Valor esperado en ambos miembros:

$$E\{\Delta w[k+1]\} =\ E\{\{I -\ 2.\mu.x[k].x^T[k]\}.\Delta w[k]\} +\ \ 2.\mu.E\{e_0[k].x[k]\}$$

Asumiendo que los valores de $x[k]$ estadísticamente son independientes de los valores de $\Delta w[k]$ y ortogonales a $e_0[k]$. Desaparece segundo término del lado derecho, se realizan las siguientes simplificaciones:

$$E\{\Delta w[k+1]\} = (I - 2.\mu.E\{x[k].x^T[k]\}).E\{\Delta w[k]\}$$

$$E\{\Delta w[k+1]\} = (I - 2.\mu.R).E\{\Delta w[k]\}$$

La primera suposición se justifica asumiendo que la desviación en los parámetros depende únicamente de los vectores de señal de entrada anteriores, mientras que la segunda suposición consideramos que la señal de error en la solución óptima es ortogonal a los elementos del vector de señal de entrada.

La expresión anterior conduce a:

$$E\{\Delta w[k+1]\} = (I - 2.\mu.R)^{k+1}.E\{\Delta w[0]\}$$

Premultiplicamos la ecuación: $E\{\Delta w[k+1]\} = (I - 2.\mu.R).E\{\Delta w[k]\}$ por Q^T, donde Q es la matriz unitaria que diagonaliza R generando una transformación similar. Al ser matriz unitaria cumple $Q.Q^* = I$, Siendo Q^* el traspuesto conjugado, también llamado Hermitiano. Obtenemos:

$$E\{Q^T.\Delta w[k+1]\} = (I - 2.\mu.Q^T R.Q).E\{Q^T.\Delta w[k]\}$$

Donde $\Delta w'[k+1] = Q^T.\Delta w[k+1]$ es el vector de error de coeficientes rotado

$$E\{Q^T.\Delta w[k+1]\} = E\{\Delta w'[k+1]\} = (I - 2.\mu.Q^T R.Q).E\{\Delta w'[k]\}$$

Siendo $\Lambda = Q^T R.Q$

$$E\{Q^T.\Delta w[k+1]\} = E\{\Delta w'[k+1]\} = (I - 2.\mu.\Lambda).E\{\Delta w'[k]\}$$

$$= (I - 2.\mu.\Lambda).E\{\Delta w'[k]\}$$

$$E\{\Delta w'[k+1]\} = \begin{vmatrix} 1-2.\mu.\lambda_0 & 0 & \dots. & 0 \\ 0 & 1-2.\mu.\lambda_1 & & \\ & & & \\ 0 & 0 & & 1-2.\mu.\lambda_N \end{vmatrix} . E\{\Delta w'[k]\}$$

Aplicando la rotación, se obtiene un análisis más simple del comportamiento del error.

Utilizando esta ecuación: $E\{\Delta w[k+1]\} = (I-2.\mu.R)^{k+1}.E\{\Delta w[0]\}$, en forma alternativa, se puede escribir la siguiente expresión:

$$E\{\Delta w'[k+1]\} = (I-2.\mu.\Lambda)^{k+1}.E\{\Delta w'[0]\}$$

$$E\{\Delta w'[k+1]\} = \begin{vmatrix} (1-2.\mu.\lambda_0)^{k+1} & 0 & \dots. & 0 \\ 0 & (1-2.\mu.\lambda_1)^{k+1} & & \\ & & & \\ 0 & & & (1-2.\mu.\lambda_N)^{k+1} \end{vmatrix} . E\{\Delta w'[0]\}$$

Analizando cada término de la matriz: $(1-2.\mu.\lambda_i)^{k+1}$, se tienen potencias de $k+1$

Se puede comparar con la serie geométrica: $\sum_{i=0}^{\infty} q^n = \frac{1}{1-q}$; $|q| < 1$ para que converja.

Es decir que se debe cumplir $\left|(1-2.\mu.\lambda_i)^{k+1}\right| < 1$

Sabiendo que los autovalores de la matriz de correlación R son siempre reales y positivos

Entonces: $2.\mu.\lambda_i < 2$, despejamos μ y tomamos λ_i su valor máximo. Entonces, para poder garantizar la convergencia de los coeficientes en la media, el factor de convergencia del algoritmo LMS debe elegirse en el siguiente rango:

$\boxed{0 < \mu < \frac{1}{\lambda_{max}}}$; Partiendo de esta ecuación: $\Delta w[k+1] = \Delta w[k] + 2.\mu.e[k].x[k]$

En cambio, si se utiliza el factor μ según la siguiente ecuación: $\Delta w[k+1] = \Delta w[k] + \mu.e[k].x[k]$, se obtiene:

$$\boxed{0 < \mu < \frac{2}{\lambda_{max}}}$$

Siendo λ_{max} el máximo autovalor de la matriz de correlación R. Los valores de μ elegidos garantizan que todos los elementos de la matriz diagonal: $(1-2.\mu.\lambda_i)^{k+1}$ tiendan a 0 cuando con $k \to \infty$. Así también $E\{\Delta w'[k+1]\}$ tenderá a 0 para valores grandes de k

En el análisis anterior se utiliza la teoría de la independencia, que considera todos los vectores $x[k]$, estadísticamente independientes. Esta suposición nos permitió considerar independientes $\Delta w[k]$ de $x[k].x^T[k]$. Tal suposición, a pesar de no ser rigurosamente válida, especialmente cuando se compone de los elementos de una línea de retardo, conduce a resultados teóricos correctos que están de acuerdo con los resultados experimentales.

La condición $0 < \mu < \frac{2}{\lambda_{max}}$ para que el algoritmo LMS converja requiere conocer el valor máximo del autovalor: λ_{max} de la matriz de correlación R. Generalmente en los algoritmos LMS el valor de λ_{max} no está disponible. En la práctica, para solucionar esta dificultad, se toma la traza de R como un valor conservativo. Entonces se reformula la condición de la siguiente manera:

$$0 < \mu < \frac{2}{tr[R]}$$

Se tiene en cuenta que la matriz Res positiva y Toeplitz con todos los elementos en su diagonal principal igual a $r[0]$. Siendo $r[0]$ el valor cuadrático medio de cada una de las M entradas. La traza

de una matriz se obtiene con la suma de los elementos correspondientes a la diagonal principal, entonces:

$$tr[R] = M.r[0] = \sum_{k=0}^{M-1} E\{|x[n-k]|^2\}$$

Entonces se usa la suma de los valores cuadráticos medios de las entradas $x[n],\ x[n-1], \ldots\ x[n-M+1]$, denominada en inglés: "tap-input power". Se escribe nuevamente la condición de convergencia del algoritmo LMS:

$$0 < \mu < \frac{2}{tr[R]} = \frac{2}{\text{tap input power}}$$

A su vez, tomando un valor más conservativo, para simplificar cuentas se puede utilizar: $0 < \mu < \frac{2}{M.Potencia}$

Implementación del algoritmo LMS

Parámetros: M cantidad de coeficientes del filtro

μ: parámetro de paso

$0 < \mu < \frac{2}{tr[R]} = \frac{2}{\text{tap input power}}$

$\text{tap input power} = \sum_{k=0}^{M-1} E\{|x[n-k]|^2\}$

Inicialización: si se conoce un valor estimado de $\widehat{w}[n]$ se utiliza para inicializar $\widehat{w}[0]$. Sino se inicializa con $\widehat{w}[0] = 0$

Se adquiere vector de datos $x[n]$ de tamaño $Mx1$ en el tiempo n

$d[n]$ es la entrada esperada para tiempo n

Se calcula $\widehat{w}[n+1]$, vector estimado de pesos para el tiempo $n+1$

Se calcula para $n = 0,1,2$ ….

$$e[n] = d[n] - \widehat{w}^H[n].x[n]$$
$$\widehat{w}[n+1] = \widehat{w}[n] + \mu.x[n].e^*[n]$$

Conclusiones

1 - Los filtros adaptativos se pueden utilizar a entornos donde hay cambios de señal, superposición espectral entre ruido y señal y ruidos desconocidos o variables en el tiempo y en identificación de sistemas

2. La teoría del filtro de Wiener proporciona soluciones de peso óptimas basadas en estadísticas. Implica la recopilación de un gran bloque de datos, el cálculo de una matriz de autocorrelación y una matriz de correlación cruzada, y la inversión de la matriz de autocorrelación de gran tamaño

3. El algoritmo de descenso puede encontrar la solución de peso óptima utilizando un método iterativo, por lo que no se necesita invertir una matriz grande. Igualmente requiere calcular una matriz de autocorrelación y correlación cruzada.

4. El LMS es un algoritmo basado en muestras, que no necesita el cálculo de estadísticas y no implica la inversión de matrices.

5. El factor de convergencia del algoritmo LMS está limitado por el recíproco del producto del número de coeficientes de filtro y la potencia de su señal de entrada.

Algoritmo recursivo de los mínimos cuadrados (RLS)

El segundo enfoque para desarrollo de algoritmos de filtrado adaptativo lineal se basa en el método de recursivo de los mínimos cuadrados (RLS: Recursive Least Squares). De acuerdo con este método, se minimizan la suma de los errores cuadráticos de las diferencias entre la respuesta deseada y la salida real del filtro.

Resulta diferente al método del gradiente estocástico, esta minimización se logra mediante manipulaciones de matrices algebraicas, lo que resulta en una regla de actualización que puede expresarse en palabras, de la siguiente manera:

[Vector de pesos actualizados] = [Vector de pesos anterior] + [Vector de ganancia] x [Innovación]

Donde la innovación es información "nueva" puesta en el proceso de filtrado en el momento de su actualización. El algoritmo de filtrado adaptativo así descrito se denomina algoritmo de *mínimos cuadrados recursivos (RLS)*. Una propiedad distintiva de este segundo algoritmo es su rápida tasa de convergencia, que se logra a expensas de una mayor complejidad computacional.

Cuando se reciben muestras nuevas de las señales entrantes en cada iteración, la solución para el problema de mínimos cuadrados se puede calcular en forma recursiva, lo que da como resultado los algoritmos de mínimos cuadrados recursivos (RLS). Se sabe que los algoritmos RLS persiguen una convergencia rápida incluso cuando la dispersión de los valores propios de la matriz de correlación correspondiente a la señal de entrada es grande. Los algoritmos RLS pueden tener un rendimiento competitivo cuando se trabaja en entornos que varían en el tiempo, según modelos de parámetros desconocidos. Todas estas ventajas tienen el costo de una mayor complejidad computacional y algunos problemas de estabilidad, que no son tan críticos en los algoritmos basados en LMS.

Los coeficientes se actualizan a medida que va cambiando la entrada al sistema. Salida del filtro:

$y[k] = \sum_{i=0}^{N} w_i[k].x[k-i] = w^T[k].x[k]$ *FIR*

Siendo $x[k] = \left[x[k],\ x[k-1],\ \ldots\ldots.x[k-N]\right]^T$

$w[k] = \left[w_0[k],\ w_1[k],\ \ldots\ldots.w_N[k]\right]^T$

N es el orden del Filtro

El proceso de minimización de error requiere la información de la señal de entrada disponible hasta el momento. Además, la función objetivo que buscamos minimizar es determinística: $\varepsilon^d[k]$

$$\varepsilon^d[k] = \sum_{i=0}^{k} \lambda^{k-i}.\varepsilon^2[i]$$

$$\varepsilon^d[k] = \sum_{i=0}^{k} \lambda^{k-i}.\ \left[d[i] - x^T[i].w[k]\right]^2$$

$w[k]$ son los coeficientes del filtro adaptativo

$\varepsilon[i]$ es el error «a posteriori» en instante i

λ es un factor de pesos exponencial: $0 \ll \lambda < 1$

λ también se llama factor de olvido

$\varepsilon[k]$ es el error «a posteriori»

$\varepsilon^d[k] = \sum_{i=0}^{k} \lambda^{k-i}. \left[d[i] - x^T[i].w[k]\right]^2$

Como se puede notar, «cada error» consiste en la diferencia de la salida deseada y la salida del filtro, usando los coeficientes más recientes $w[k]$

Calculamos la derivada respecto de $w[k]$

$$\frac{\partial \varepsilon^d[k]}{\partial w[k]} = -2.\sum_{i=0}^{k} \lambda^{k-i}.x[i].\ \{d[i] - x^T[i].w[k]\}$$

Igualamos a 0 para buscar el valor mínimo:

$$\sum_{i=0}^{k} \lambda^{k-i}.x[i].\ \{d[i] - x^T[i].w[k]\} = \begin{bmatrix}0\\0\\..\\0\end{bmatrix}$$

$$\sum_{i=0}^{k} \lambda^{k-i}.x[i].d[i] - \sum_{i=0}^{k} \lambda^{k-i}.x[i].x^T[i].w[k] = \begin{bmatrix}0\\0\\..\\0\end{bmatrix}$$

$$\sum_{i=0}^{k} \lambda^{k-i}.x[i].d[i] = \sum_{i=0}^{k} \lambda^{k-i}.x[i].x^T[i].w[k]$$

$$w[k] = \left[\sum_{i=0}^{k} \lambda^{k-i}.x[i].x^T[i]\right]^{-1}.\sum_{i=0}^{k} \lambda^{k-i}.x[i].d[i]$$

$$w[k] = {R_D}^{-1}[k].p_D[k]$$

Al minimizar el error determinístico, obtuvimos:

$$w[k] = \left[\sum_{i=0}^{k} \lambda^{k-i}.x[i].x^T[i]\right]^{-1}.\sum_{i=0}^{k} \lambda^{k-i}.x[i].d[i] = {R_D}^{-1}[k].p_D[k]$$

${R_D}^{-1}[k] = \left[\sum_{i=0}^{k} \lambda^{k-i}.x[i].x^T[i]\right]^{-1}$; $\boxed{p_D[k] = \sum_{i=0}^{k} \lambda^{k-i}.x[i].d[i]}$

Se utiliza la información previa, teniendo una matriz de correlación determinística $R_D[k]$

${R_D}^{-1}[k]$: Matriz Inversa de correlación determinística

$p_D[k]$: matriz de correlación cruzada entre x y d

λ: forgetting factor (factor de olvido)

Mediante el «Lema de Inversión de Matrices» se estima ${\mathrm{R_D}}^{-1}[\mathrm{k}]$ de la siguiente forma:

$$\boxed{S_D[k] = {R_D}^{-1}[k] = \frac{1}{\lambda}.\left[S_D[k-1] - \frac{S_D[k-1].x[k].x^T[k].S_D[k-1]}{\lambda + x^T(k).S_D[k-1].x[k]}\right]}$$

La Matriz $S_D[k]$ se inicializa como matriz diagonal. En varias iteraciones se va modificando dicha matriz para reducir el error

Una forma práctica de estimar la matriz R es tomar valores anteriores de $\hat{R}$ multiplicados por un factor α pequeño

$$\boxed{\hat{R}[k] = \alpha . x[k]x^T[k] + (1-\alpha).\hat{R}[k-1]}$$

Aplicamos misma regla, Reemplazamos $\hat{R}[k-1]$

$$\hat{R}[k] = \alpha . x[k]x^T[k] + (1-\alpha).\{\alpha . x[k-1]x^T[k-1] + (1-\alpha).\hat{R}[k-2]\}$$

$$\hat{R}[k] = \alpha . x[k]x^T[k] + (1-\alpha).\alpha . x[k-1]x^T[k-1] + (1-\alpha)^2.\hat{R}[k-2]$$

Reemplazamos: $\hat{R}[k-2]$ …. Luego $\hat{R}[k-3]$ se vuelve RECURSIVO.

Obtenemos:

$$\boxed{\hat{R}[k] = \alpha . x[k]x^T[k] + \alpha . \sum_{i=0}^{k-1}(1-\alpha)^{k-i}.x[i]x^T[i]}$$

Tomando la última ecuación, si aplicamos$E\{\quad\}$en ambos lados y $k \to \infty$

$E\{\hat{R}[k]\} = \alpha .\ \sum_{i=0}^{k}(1-\alpha)^{k-i}.E\{x[i]x^T[i]\} = R\ \ ;\quad k \to \infty$

Es decir que para valores muy grandes de k obtenemos R

Para invertir $\hat{R}[k]$ utilizamos lema de inversión de matriz (matrix inversion lemma), que dice:

$$\boxed{[A+BCD]^{-1} = A^{-1} - A^{-1}B[DA^{-1}B + C^{-1}]^{-1}DA^{-1}}$$

Siendo *A,B,C,D* matrices de tamaño apropiado. A y C no singulares

Tomamos: $\hat{R}[k] = \alpha . x[k]x^T[k] + (1-\alpha).\hat{R}[k-1]$

Reemplazamos: $A = (1-\alpha).\hat{R}[k-1]\quad ; B = D^T = x[k]\quad ;\ \ C = \alpha$

Operando se obtiene:

$$\boxed{S_D[k] = {R_D}^{-1}[k] = \frac{1}{\lambda}.\left[S_D[k-1] - \frac{S_D[k-1].x[k].x^T[k].S_D[k-1]}{\lambda + x^T(k).S_D[k-1].x[k]}\right]}$$

${R_D}^{-1}[k]$ resulta menos compleja de calcular (de orden N^2 multiplicaciones) que la inversión directa de $\hat{R}[k]$ (de orden N^3 multiplicaciones)

Resumiendo, tenemos las siguientes ecuaciones para el algoritmo RLS:

$$\boxed{p_D[k] = \sum_{i=0}^{k}\lambda^{k-i}.x[i].d[i]}$$

$$\boxed{S_D[k] = {R_D}^{-1}[k] = \frac{1}{\lambda}.\left[S_D[k-1] - \frac{S_D[k-1].x[k].x^T[k].S_D[k-1]}{\lambda + x^T(k).S_D[k-1].x[k]}\right]}$$

$$\boxed{w[k] = {R_D}^{-1}[k].p_D[k]}$$

En los resultados anteriores se utiliza error a posteriori

También se puede utilizar el error a priori, que corresponde al algoritmo RLS Alternativo

$$e[k] = d[k] - x^T[k].w[k-1]$$

Manipulando matrices se obtiene la siguiente expresión:

$$w[k] = w[k-1] + e[k].S_D[k].x[k]$$

En el método RLS la Innovación es información "nueva" puesta en el proceso de filtrado en el momento de su actualización de coeficientes.

RLS presenta las siguientes ventajas: Rápida tasa de convergencia. Incluso con dispersiones grandes de matriz de correlación. Buen rendimiento en entornos que varían con el tiempo con parámetros desconocidos

Desventajas de RLS: Mayor complejidad computacional. Algunos problemas de estabilidad (LMS no es tan crítico). Es importante tener muestras nuevas de las señales entrantes en cada iteración

Algoritmos de filtrado adaptativo lineal: LMS y RLS

Los algoritmos LMS y RLS constituyen dos algoritmos básicos, alrededor de cada uno de ellos se formulan conjuntos de algoritmos. Dentro de cada conjunto, los algoritmos de filtrado adaptativo se diferencian entre sí en la forma en que se configura la estructura de filtrado. Sin embargo, independientemente de la estructura de filtrado utilizada alrededor de la cual se realice la adaptación de los parámetros, los algoritmos dentro de cada conjunto heredan ciertas propiedades arraigadas en los algoritmos LMS y RLS. Específicamente:

- Los algoritmos basados en LMS son independientes del modelo, en el sentido de que no se realizan supuestos estadísticos para derivarlos.
- Por otro lado, los algoritmos basados en RLS dependen del modelo, ya que sus derivadas suponen el uso de un modelo gaussiano multivariado.

La diferenciación que hemos hecho aquí tiene un impacto profundo en la tasa de convergencia, seguimiento y robustez del algoritmo.

Análisis de Redes Neuronales y Sistemas adaptativos

Vamos a comparar una red neuronal fija (RNF) y un sistema adaptativo

a) En un sistema adaptativo los pesos de los coeficientes del filtro se actualizan constantemente, mientras en una RNF solo se modifican para un conjunto de entrenamiento, luego de entrenada la red fija no se modifican.
b) Los sistemas adaptativos admiten cambios en la señal de entrada, esto no ocurre en las redes fijas
c) Las RNF olvidan los datos antiguos, con lo cual si aparece ruido se pueden desviar notablemente sus parámetros óptimos. Mientras los sistemas adaptativos se comportan como un sistema de memoria larga.

Para tener un buen sistema se utiliza información de estado que se comporta como memoria de larga duración. De esta manera, cuando el sistema no puede predecir la entrada, actualiza la salida a partir de estos estados.

En un sistema adaptativo, cada peso de los coeficientes del filtro converge a un valor mínimo con una velocidad que resulta inversamente al autovalor del peso. A su vez, entre una iteración y otra se tiene un paso, si este paso realiza un ajuste grande, la velocidad de adaptación será mayor, donde el paso máximo está limitado por el mayor autovalor, por tanto, la dispersión en los autovalores de la matriz de correlación correspondiente a las entradas define el comportamiento que tendrá el algoritmo.

La elección del paso termina siendo un compromiso entre la velocidad de adaptación y la precisión, por ejemplo, para un desajuste de un 15 %, el algoritmo converge en una cantidad de pasos que resulta aproximadamente 15 veces el valor de sus pesos. En muchas aplicaciones con estas velocidades de convergencia se pueden implementar sistemas en tiempo real.

Análisis de convergencia RLS

RLS converge a la solución de mínimos cuadrados aproximadamente con 2M interacciones, independientemente de que se produzca en un entorno ergódico (con factor de olvido unitario). Notamos que este algoritmo no se ve afectado por la dispersión de autovalores de la matriz de autocorrelación, pero si puede presentar inestabilidad numérica por el lema de inversión matricial.

En la Tabla III se muestra una comparación de los algoritmos LMS, NLMS y RLS

Tabla III – Comparación de algoritmos LMS, NLMS y RLS

Algoritmo	**Tamaño de paso**	**Observaciones**
LMS	μ pequeño	Necesita muchas muestras para que el sistema converja
	μ grande	Velocidad de convergencia mayor y más eficiente, pero aumenta el ruido
NLMS	μ pequeño	Necesita muchas muestras para que el sistema converja. Es más estable que el LMS
	μ grande	La velocidad de convergencia es mayor, sin embargo, el ruido aumenta. Óptimo para aplicaciones de cancelaciones de ruido.
RLS	μ pequeño	La velocidad de convergencia es mayor, pero es más inestable y puede no converger
	μ grande	Disminuye la velocidad de convergencia. Puede no converger

Filtrado adaptativo en el dominio de la frecuencia

En los algoritmos tradicionales y normalizados de mínimos cuadrados medios (LMS) descritos, los pesos del filtro de respuesta al impulso finita (FIR) se adaptan en el dominio del tiempo. Reconociendo que la transformada de Fourier mapea las señales del dominio del tiempo en el dominio de la frecuencia y que la transformada de Fourier inversa proporciona el mapeo inverso que nos lleva de regreso al dominio temporal, vemos que es igualmente factible realizar la adaptación de los

parámetros de los filtros analizados en el dominio frecuencial. En tal caso, hablamos de filtrado adaptativo en el dominio de la frecuencia (FDAF), cuyo origen puede remontarse a un artículo de Walzman y Schwartz (1973).

Hay dos motivaciones para buscar la adaptación en el dominio de la frecuencia:

Motivación 1. En determinadas aplicaciones, como la cancelación de eco acústico en teleconferencias, se requiere que el filtro adaptativo mantenga una respuesta impulsional muy larga (es decir, memoria larga) para hacer frente a una duración de eco igualmente larga. Cuando el algoritmo LMS se adapta en el dominio del tiempo, encontramos que el requisito de una memoria larga da como resultado un aumento significativo en la complejidad computacional del algoritmo. Los filtros adaptativos en el dominio de la frecuencia presentan una posible solución al problema de complejidad computacional.

Motivación 2. se logra una tasa de convergencia más uniforme mediante la explotación de las propiedades de ortogonalidad que tiene la transformada del coseno discreta (DCT). El filtrado adaptativo que emplea la ortogonalización, implementado de una forma diferente a la descripta en la Motivación 1, se utiliza para mejorar el rendimiento de convergencia del algoritmo LMS tradicional.

También se puede realizar un filtro adaptativo de sub-banda, que es diferente de un filtro adaptativo auto-ortogonalizable. En un filtro adaptativo de sub-banda, los filtros elimina bandas con alto rechazo de la banda de supresión se utilizan para dividir en banda la señal de entrada, por tanto, posiblemente proporcionar una mejora en la convergencia sobre el filtro adaptativo en el dominio de la frecuencia.

Además, reduciendo las señales de sub-banda (por ejemplo, reduciendo el muestreo a una tasa más baja), es posible lograr una reducción importante en la complejidad computacional. Así, un filtro adaptativo de sub-banda ofrece una alternativa atractiva en el dominio de la frecuencia del filtro adaptativo.

En el filtrado adaptativo en bloques se utilizan las siguientes técnicas:

- Filtrado adaptativo de dominio de frecuencia (FDAF)
- Filtrado adaptativo auto-ortogonalizable
- Filtrado adaptativo de sub-banda

Se estudiarán datos con valores reales para simplificar la presentación.

Filtros adaptativos en bloque.

En el filtrado adaptativo en bloques, dada una secuencia $x[n]$ se secciona en bloques de tamaño L mediante un convertidor de serie a paralelo, que se utilizan como entrada a un filtro de respuesta al impulso de duración finita (FIR) de longitud M, ingresa un bloque a la vez. De modo que la adaptación del filtro se realiza para cada bloque en lugar de muestra por muestra como en el algoritmo LMS tradicional (Clark et al., 1981; Shynk, 1992).

Tenemos las siguientes ecuaciones correspondientes a la entrada y a los pesos del filtro para el tiempo n:

$$x[n] = \left[x[n], x[n-1], \ldots . x[n-M+1]\right]^T$$

$$\widehat{w}[n] = \left[\widehat{w}_0[n], \widehat{w}_1[n], \ldots . \widehat{w}_{M-1}[n]\right]^T$$

Utilizamos k para indicar el índice del bloque, reescribimos

$n = k.L + i$, con $i = 0,1,2 \ldots . , L-1$ y $k = 1,2, \ldots .$

Siendo L el largo del bloque. Los datos de entrada para el bloque k del conjunto $\{x[k.L+i]\}_{i=0}^{L-1}$ se escriben mediante la matriz A

$$A^T[k] = \left[x[k.L], x[k.L+1], \ldots . x[k.L+L-1]\right]^T$$

En la Tabla IV se muestra un ejemplo de matriz $A[k]$ para un filtro de largo $M = 6$ y bloque de largo $L = 3$

Tabla IV – Matriz de datos para procesamiento en bloques

Matriz de Datos $A[k]$						
$x^T[3.k]$	10	9	8	7	6	5
$x^T[3.k+1]$	11	10	9	8	7	6
$x^T[3.k+2]$	12	11	10	9	8	T

Donde $x[3.k+2] = 12,\ x[3.k+1] = 11\ ,\ x[3.k] = 10$

Para un filtro en bloques LMS se tienen las siguientes ecuaciones

Entrada: $x[k.L+i]$

Salida del filtro:

$$y[k.L+i] = \widehat{w}^T[k].x[k.L+i] \qquad (161)$$

$$y[k.L+i] = \sum_{j=0}^{M-1} \widehat{w}_j[k].x[k.L+i-j] \quad ; \quad i = 0,1,2, \ldots . L-1$$

Respuesta deseada: $d[k.L+i]$

Error: $e[k.L+i] = d[k.L+i] - y[k.L+i]$

El error se calcula para cada bloque

A continuación se muestra un ejemplo para $L = 3$, donde se observa que la matriz de datos es Toeplitz para la diagonal principal

Bloque $k-1$: $\begin{bmatrix} x[3.k-3] & x[3.k-4] & x[3.k-5] \\ x[3.k-2] & x[3.k-3] & x[3.k-4] \\ x[3.k-1] & x[3.k-2] & x[3.k-3] \end{bmatrix} . \begin{bmatrix} w_0[k-1] \\ w_1[k-1] \\ w_2[k-1] \end{bmatrix} = \begin{bmatrix} y[3.k-3] \\ y[3.k-2] \\ y[3.k-1] \end{bmatrix}$

Bloque k: $\begin{bmatrix} x[3.k] & x[3.k-1] & x[3.k-2] \\ x[3.k+1] & x[3.k] & x[3.k-1] \\ x[3.k+2] & x[3.k+1] & x[3.k] \end{bmatrix} . \begin{bmatrix} w_0[k] \\ w_1[k] \\ w_2[k] \end{bmatrix} = \begin{bmatrix} y[3.k] \\ y[3.k+1] \\ y[3.k+2] \end{bmatrix}$

Bloque $k+1$: $\begin{bmatrix} x[3.k+3] & x[3.k+2] & x[3.k+1] \\ x[3.k+4] & x[3.k+3] & x[3.k+2] \\ x[3.k+5] & x[3.k+4] & x[3.k+3] \end{bmatrix} . \begin{bmatrix} w_0[k+1] \\ w_1[k+1] \\ w_2[k+1] \end{bmatrix} = \begin{bmatrix} y[3.k+3] \\ y[3.k+4] \\ y[3.k+5] \end{bmatrix}$

Para el algoritmo LMS en bloques con señales reales tenemos:

$$\widehat{w}[k+1] = \widehat{w}[k] + \mu . \sum_{i=0}^{L-1} x[k.L+i]\, e[k.L+i]$$

Definimos $\Phi[k]$ matriz de correlación cruzada de tamaño Mpor 1

$$\Phi[k] = \sum_{i=0}^{L-1} x[k.L+i]\, e[k.L+i] \qquad (162)$$

$$\Phi[k] = A^T[k].e[k]$$

Siendo $A[k]$ la matriz de datos de tamaño LxM

El vector $e[k]$ es la señal de error de tamaño $Lx1$

$$e[k] = \left[e[k.L], e[k.L+1], \dots\dots, e[k.L+L-1]\right]^T$$

Reemplazamos $\Phi[k]$ en el vector de pesos:

$$\widehat{w}[k+1] = \widehat{w}[k] + \mu.\Phi[k] \qquad (163)$$

Para el algoritmo en bloques LMS se incorpora un vector gradiente estimado:

$$\widehat{\nabla}[k] = -\frac{2}{L}.\sum_{i=0}^{L-1} x[k.L+i]\, e[k.L+i]$$

El factor 2 se incluye para consistencia con el algoritmo LMS tradicional, mientras que el factor $1/L$ se agrega para realizar un promedio temporal. Expresamos el algoritmo LMS en función del gradiente $\widehat{\nabla}[k]$ y de la constante nueva $\mu_B$definida como paso efectivo

$$\widehat{w}[k+1] = \widehat{w}[k] - \frac{1}{2}\mu_B.\widehat{\nabla}[k]$$

$$\mu_B = L.\mu$$

Elección del tamaño del bloque

Se tienen 3 posibles casos para elegir el largo del bloque:

a) $L = M$ es el valor óptimo respecto a la carga computacional

b) $L < M$ presenta la ventaja de tener un retardo reducido en el procesamiento

b) $L > M$ da lugar a operaciones redundantes en el proceso adaptativo, porque la estimación del vector de gradiente (calculado sobre L puntos) ahora usa más información que el propio filtro.

Limitaremos nuestra atención al caso de L = M, que es la opción más utilizada en el filtrado adaptativo por bloques.

Cuando hablamos de complejidad computacional, nos referimos al número de transformaciones rápidas de Fourier y el tamaño de cada transformación que se necesita para implementar el algoritmo.

Algoritmo en bloques rápido LMS

En el algoritmo en bloques rápido LMS se realiza una adaptación en frecuencias utilizando procesamiento en bloques y se utiliza el algoritmo de la Transformada Rápida de Fourier (FFT). Implementado por Clark et al. (1981, 1983) and Ferrara (1980).

Se busca una implementación eficiente del algoritmo en bloques LMS

Las ecuaciones (161) definen una convolución lineal entre la entrada y los pesos del filtro

$y[k.L+i] = \widehat{w}^T[k].x[k.L+i] = y[k.L+i] = \sum_{j=0}^{M-1} \widehat{w}_j[k].x[k.L+i-j] \quad ; \quad i = 0,1,2,\ldots..L-1$

La ecuación (162) define una correlación lineal entre la entrada y la señal de error

$$\Phi[k] = \sum_{i=0}^{L-1} x[k.L+i]\, e[k.L+i] = A^T[k].e[k]$$

El algoritmo de la transformada rápida de Fourier (conocido como FFT) es una herramienta muy potente para realizar convoluciones y correlaciones rápidas (Oppenheim 1989).

De la teoría del procesamiento de señales digitales, sabemos que el método de superposición y guardado (overlap-save) y el método de superposición y agregado (overlap-add) proporcionan dos procedimientos eficientes para calcular una convolución rápida lineal utilizando la transformada discreta de Fourier (Oppenheim y Schafer, 1989). El método de superposición y guardado es el más común de los dos para filtrado adaptativo. Además, el uso de 50 por ciento de superposición (es decir, tamaño de bloque igual al número de pesos) es el más eficiente. Utilizaremos este método. Los pesos del filtro de largo M se rellenan con un número igual de ceros, y se utiliza una FFT de N puntos para el cálculo, donde

$$N = 2.M$$

Los coeficientes $\widehat{W}[k]$ tiene un tamaño $Nx1$, el vector de ceros tiene tamaño $Mx1$. El vector de dominio frecuencial $\widehat{W}[k]$ es el doble de largo que el vector $\widehat{w}[k]$

$$\widehat{W}[k] = FFT\begin{bmatrix}\widehat{w}[k] \\ 0\end{bmatrix} \qquad (164)$$

Se muestra el vector transformado de la entrada correspondiente a 2 bloques, siendo una matriz diagonal

$$U[k] = diag\{FFT[x[k.M-M],\ldots..x[k.M-M-1],\ \ x[k.M],\ldots..x[k.M+M-1]]\}$$

Aplicamos el método overlap-save a la ecuación (161) de convolución:

$y[k.L+i] = \widehat{w}^T[k].x[k.L+i] = \sum_{j=0}^{M-1} \widehat{w}_j[k].x[k.L+i-j] \quad ; \quad i = 0,1,2,\ldots..L-1$

Obtenemos vector de tamaño $Mx1$:

$$y^T[k] = [y[k.M], y[k.M+1],\ldots..y[k.M+M-1]]$$
$$= \text{últimos } M \text{ elementos de } IFFT\left[X[k].\widehat{W}[k]\right]$$

Siendo $IFFT$ la transformada rápida de Fourier inversa. Solo se toman los últimos M elementos en la ecuación, debido a que los primeros elementos corresponden a la convolución circular.

Para la correlación lineal de la ecuación (162):

$\Phi[k] = \sum_{i=0}^{L-1} x[k.L+i]\, e[k.L+i]\ ; \quad \Phi[k] = A^T[k].e[k]$

Para el bloque k utilizamos vector de respuesta deseada y vector de error:

$$d[k] = [d[k.M], d[k.M+1],\ldots\ldots.d[k.M+M-1]]^T$$

$$e[k] = \left[e[k.M], e[k.M+1], \ldots \ldots . e[k.M+M-1]\right]^T$$
$$e[k] = d[k] - y[k]$$

Teniendo en cuenta que se descartan los primeros elementos de $y^T[k]$, se utiliza la siguiente transformada del vector de error:

$$E[k] = FFT\begin{bmatrix} 0 \\ e[k] \end{bmatrix}$$

Aplicando el método de ovelap-save para la correlación lineal de la ecuación (162), se obtiene

$$\Phi[k] = primeros\ M\ elementos\ de\ IFFT\left[X^H[k].E[k]\right]$$

Para la convolución lineal se descartan los primeros M elementos, mientras en la correlación lineal se descartan los últimos M elementos.

Para el vector de pesos de la ecuación (163): $\widehat{w}[k+1] = \widehat{w}[k] + \mu.\Phi[k]$obtenemos:

$$\widehat{W}[k+1] = \widehat{W}[k] + \mu.FFT\begin{bmatrix} \Phi[k] \\ 0 \end{bmatrix} \qquad (165)$$

Esta última ecuación y la ecuación (164) $\widehat{W}[k] = FFT\begin{bmatrix} \widehat{w}[k] \\ 0 \end{bmatrix}$ definen el algoritmo rápido en bloques LMS

Se analiza la carga computacional para el algoritmo LMS tradicional y el algoritmo rápido en bloques LMS

Para el algoritmo tradicional con M pesos para datos reales, se utilizan M multiplicaciones para calcular la salida y luego M multiplicaciones para calcular los pesos, es decir $2.M$ multiplicaciones para cada ciclo. Para un bloque de tamaño M, el número de multiplicaciones resulta $2.M^2$

Para el algoritmo rápido para cada punto de la FFT requiere $N.log_2 N$ multiplicaciones, siendo $N = 2.M$. Se necesitan 5 transformaciones en frecuencias. Además, se necesitan $4.N$ multiplicaciones para la salida en frecuencias y $4.N$ multiplicaciones para la correlación cruzada. Es decir, para este algoritmo se utilizan

$$5.N.log_2 N + 8.N = 10.M.log_2(2.M) + 16.M = 10.M.log_2(M) + 26.M$$

Obtenemos la relación entre los 2 algoritmos (Shynk 1992)

$Relación = \frac{10.M.log_2(M)+26.M}{2.M^2}$; $Relación = \frac{5.log_2(M)+13}{M}$

Por ejemplo para $M = 2048$ se tiene $Relación = \frac{5.log_2(2048)+13}{2048} = 0{,}0332$

Es decir que el algoritmo rápido LMS resulta 30 veces más rápido

Usando las fórmulas anteriores armamos la Tabla V.

Tabla V – Relación de carga computacional entre algoritmos

M	Tradicional LMS	Rápido LMS	Relación
1	2	26	0,1
2	8	72	0,1
10	200	592,1928	0,3
30	1800	2252,067	0,8
40	**3200**	**3168,771**	**1,0**
50	5000	4121,928	1,2
100	20000	9243,856	2,2
1000	2000000	125657,8	15,9
2048	8388608	278528	30,1
4096	33554432	598016	56,1

Para ambientes no estacionarios se realizan las siguientes modificaciones:

$\mu_i = \frac{\alpha}{P_i}; \qquad i = 0{,}1, \ldots . 2M-1$

α es constante

P_i Potencia promedio estimada para cada i

Se calcula la potencia de entrada como una suma exponencial con pesos ponderados:

$$P_i[k] = (1-\gamma).\sum_{l=0}^{\infty} \gamma^l . |X_i(k-l)|^2$$

Siendo γ un factor de olvido que controla la memoria del algoritmo.

$$\mu[k] = \alpha . D[k]$$

$$D[k] = diag\left[P_0^{-1}[k], P_1^{-1}[k], \ldots \ldots, P_{2M-1}^{-1}[k]\right]$$

$$\Phi[k] = primeros\ M\ elementos\ de\ IFFT\left[D[k].X^H[k].E[k]\right]$$

Algoritmo No contrastado

Una simplificación del algoritmo rápido en bloques LMS, se puede realizar reduciendo estas ecuaciones:

$$\Phi[k] = primeros\ M\ elementos\ de\ IFFT\left[X^H[k].E[k]\right]$$

$$\widehat{W}[k+1] = \widehat{W}[k] + \mu . FFT\begin{bmatrix}\Phi[k] \\ 0\end{bmatrix}$$

Se utiliza

$$\widehat{W}[k+1] = \widehat{W}[k] + \mu . X^H[k] . E[k]$$

Sin embargo, la estimación del vector de gradiente calculado aquí ya no corresponde a una correlación lineal; más bien, ahora tenemos una correlación circular. En consecuencia, encontramos que, en general, el algoritmo FDAF no contrastado se desvía del algoritmo LMS de bloque rápido, ya que el vector de peso de derivación ya no se acerca a la solución de Wiener cuando el número de ciclos de adaptación de bloque se acerca al infinito (Sommen et al., 1987; Lee & Un, 1989; Shynk, 1992).

Seguimiento de sistemas variables en el tiempo

Desafortunadamente, muchos de los entornos que se encuentran en la práctica no son estacionarios, por lo que la solución de Wiener adquiere una forma variable en el tiempo. En una aplicación del mundo real de este tipo, un algoritmo de filtrado adaptativo tiene una tarea adicional que realizar:

Realizar un seguimiento de la posición que varía continuamente en el tiempo del punto mínimo en la superficie de rendimiento del error.

En otras palabras, ahora se requiere que el algoritmo rastree continuamente las variaciones estadísticas del entorno, con la condición de que estas variaciones sean "lo suficientemente lentas" para que el rastreo algorítmico sea prácticamente factible.

Para que un filtro adaptativo ejerza su capacidad de seguimiento, primero debe pasar del modo transitorio al modo de funcionamiento de estado estable, y debe preverse un ajuste continuo de los parámetros libres del filtro. En general, la tasa de convergencia y la capacidad de seguimiento son dos propiedades diferentes del algoritmo. En particular, un algoritmo de filtrado adaptativo con buenas propiedades de convergencia no necesariamente posee una capacidad de seguimiento, y viceversa.

El seguimiento de un sistema variable en el tiempo no solo depende del tipo de algoritmo de filtrado adaptativo, sino también de las señales analizadas. Para un entorno no estacionario con énfasis en el seguimiento, podemos identificar dos cuestiones fundamentales:

1) Elección de Algoritmo LMS o RLS como base para el seguimiento

2) Ajuste automático del parámetro de adaptación del algoritmo de seguimiento para obtener mejor rendimiento

Punto 1: Elección de Algoritmo LMS o RLS como base para el seguimiento

Problema de Identificación del sistema

Proceso Markov de primer orden: $w_0[n+1] = \alpha . w_0[n+1] + \omega[n]$

$\omega[n]$: vector de ruido del proceso

Múltiple regresión: $d[n] = w_0^H[n] . u[n] + v[n]$

$v[n]$: error medido

$e[n] = d[n] - y[n]$

$e[n] = w_0^H[n] . u[n] + v[n] - \widehat{w}^H[n] . u[n]$

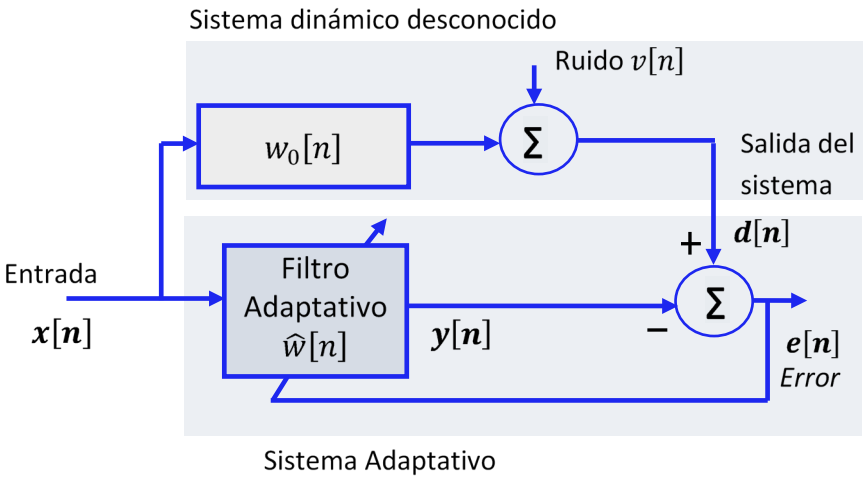


Fig. 149 –Identificación de sistemas en entorno no estacionario

Grado de no estacionariedad α para el modelo de Markov

$$\alpha = \frac{1}{\sigma_v}(tr[R_u.R_\omega])^{1/2} = \frac{1}{\sigma_v}(tr[R_\omega.R_u])^{1/2}$$

R_ω es la matriz de correlación del vector de ruido $\omega[n]$

${\sigma_v}^2$ es la varianza del ruido $v[n]$

El grado de no estacionariedad, α, tiene una relación con el desajuste $\mathcal{M}$ del filtro adaptativo.

Para el algoritmo LMS, μ_{opt} y $\mathcal{M}_{min}$ resultan:

$$\mu_{opt} \cong \frac{1}{\sigma_v}.\left(\frac{tr[R_\omega]}{tr[R_u]}\right)^{1/2}$$

$$\mathcal{M}_{min} \cong \frac{1}{\sigma_v}.(tr[R_u].tr[R_\omega])^{1/2}$$

Para el algoritmo RLS tenemos:

$$\lambda_{opt} \cong 1 - \frac{1}{\sigma_v}.\left(\frac{tr[R_\omega]}{tr[{R_u}^{-1}]}\right)^{1/2}$$

$$\mathcal{M}_{min} \cong \frac{1}{\sigma_v}.(M.tr[R_u.R_\omega])^{1/2}$$

Se puede comparar el rendimiento de seguimiento de algoritmos LMS con el del RLS dividiendo ambos coeficientes. Obtenemos la relación de desajuste $\mathcal{M}$ entre LMS y RLS

$$\frac{{\mathcal{M}_{min}}^{LMS}}{{\mathcal{M}_{min}}^{RLS}} \cong \left(\frac{tr[R_u].tr[R_\omega]}{M.tr[R_u.R_\omega]}\right)^{1/2}$$

Punto 2: Ajuste automático del parámetro de adaptación del algoritmo de seguimiento para obtener mejor rendimiento

Se utilizan algoritmos LMS y RLS modificados, donde cada algoritmo tiene 2 mecanismos de control, ver Fig. 150.

El mecanismo de control primario es impulsado por el error de estimación $e[n]$; el objetivo es controlar automáticamente los ajustes aplicados a los pesos del filtro FIR en el algoritmo de manera tradicional. El mecanismo de control secundario también es impulsado por el error de estimación, $e[n]$. Sin embargo, el objetivo es aplicar automáticamente los ajustes apropiados al parámetro de adaptación incorporado en el mecanismo de control primario.

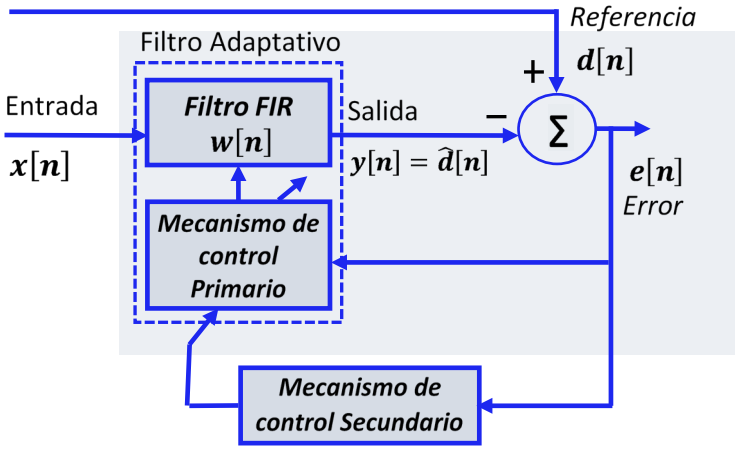


Fig. 150 - Sistema adaptativo con ajuste automático del parámetro de adaptación

Ejercicios en Matlab®

Ejercicio 4.1

Ejemplo de identificación con señales 4 QAM

La modulación de amplitud en cuadratura, también conocida como QAM (en inglés: Quadrature Amplitude Modulation) se utiliza en transmisión. En esta técnica se transmiten dos señales independientes de la señal base portadora, distintas en amplitud y en fase. Esto se logra mediante la modulación de una misma portadora, pero desfasada 90°. Una constelación 4 QAM tiene 4 puntos posibles como se muestra en la Fig. 151, estos puntos son: 0.7071 + 0.7071i, -0.7071 + 0.7071i, - 0.7071 - 0.7071i y -0.7071 - 0.7071i

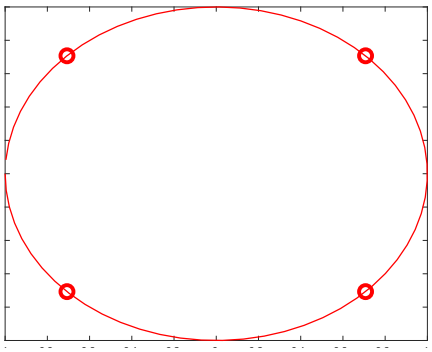


Fig. 151 – Modulación de amplitud en cuadratura para 4 QAM

Se desea implementar el sistema adaptativo de la Fig. 152

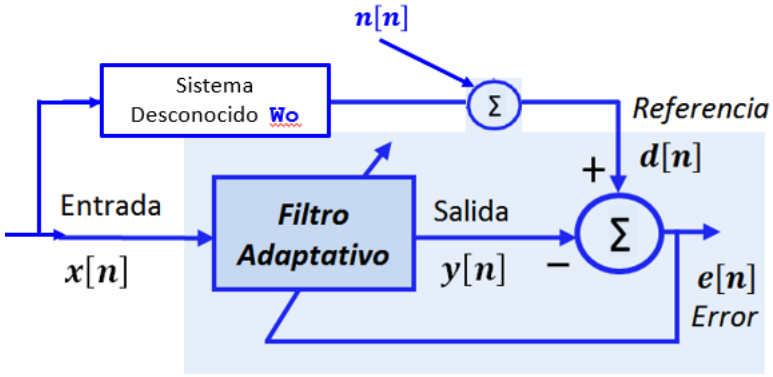


Fig. 152 – Sistema adaptativo de identificación.

Sistema Incógnito en formato Matlab®:

H = [0.31+0.24*j,-0.31+0.73*j,0.5-0.81*j,0.21+0.49*j].';

$d[n]$:Vector de Referencia de largo 600:

d(n) = (Wo'*X(:,1))+n(n);

Utilizamos el Wo conocido y agregamos Ruido

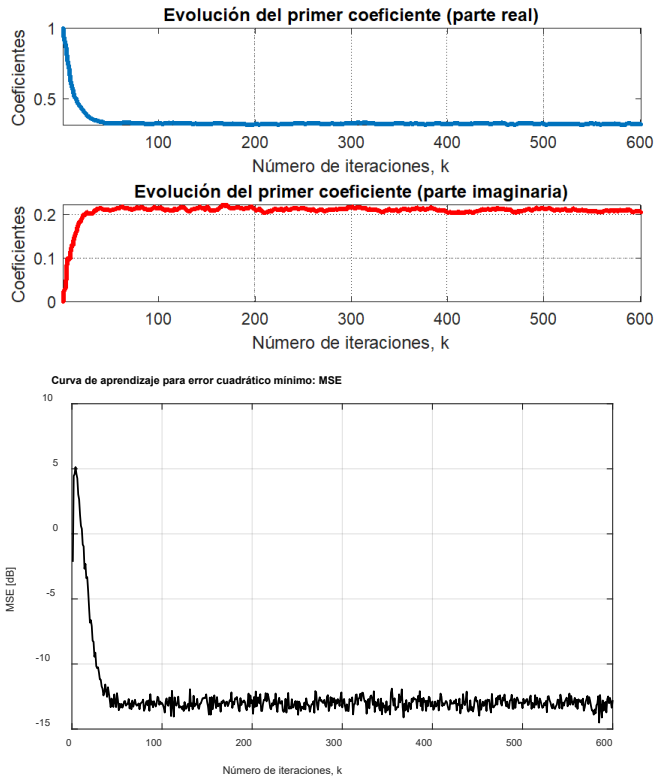
$x[n]$, es un vector que tiene los 4 puntos posibles QAM distribuidos en forma aleatoria.

Dado un $d[n]$ ruidoso y $x[n]$ aleatorio, el algoritmo LMS predice los parámetros W del filtro.

Se pide implementar el sistema en Matlab® mediante algoritmo LMS complejo, con 120 repeticiones (proceso con 120 realizaciones) y promediar las respuestas.

Resultados

Ejercicios correspondientes al seminario doctoral. Contactarse con el autor del libro para consultas y detalles de algoritmos. En la Fig. 153 se muestran los resultados del algoritmo



Solicite el código al autor

Fig. 153 – Evolución del primer coeficiente y curva de aprendizaje

Ejercicio 4.2

Ejemplo de comparación de algoritmos basados en LMS y RLS

Se dispone de un conjunto de algoritmos para la identificación de un sistema desconocido. Se utilizan para reconocer el siguiente sistema: Filtro FIR con coeficientes W0 = [0.31+0.22*j, -0.31+0.71*j, 0.49-0.8*j, 0.21+0.5*j].';

Algoritmos disponibles:

Algoritmo LMS con complejos

Algoritmo RLS

Algoritmo LMS Normalizado

Algoritmo LMS Newton

Algoritmo Transform-Domain LMS DCT (utiliza Discrete Cosine Transform: DCT)

Algoritmo Transform-Domain LMS DFT (utiliza Discrete Fourier Transform: DFT)

Se pide:

a) Analizar su funcionamiento.

b) Comparar el error cuadrático medio (MSE) para la iteración 32. Escribir en el script o en archivo word conclusiones breves. Seleccionar el algoritmo que considere mejor

Contactarse con el autor del libro para consultas y detalles de algoritmos.

En la Fig. 154 se muestran los gráficos obtenidos.

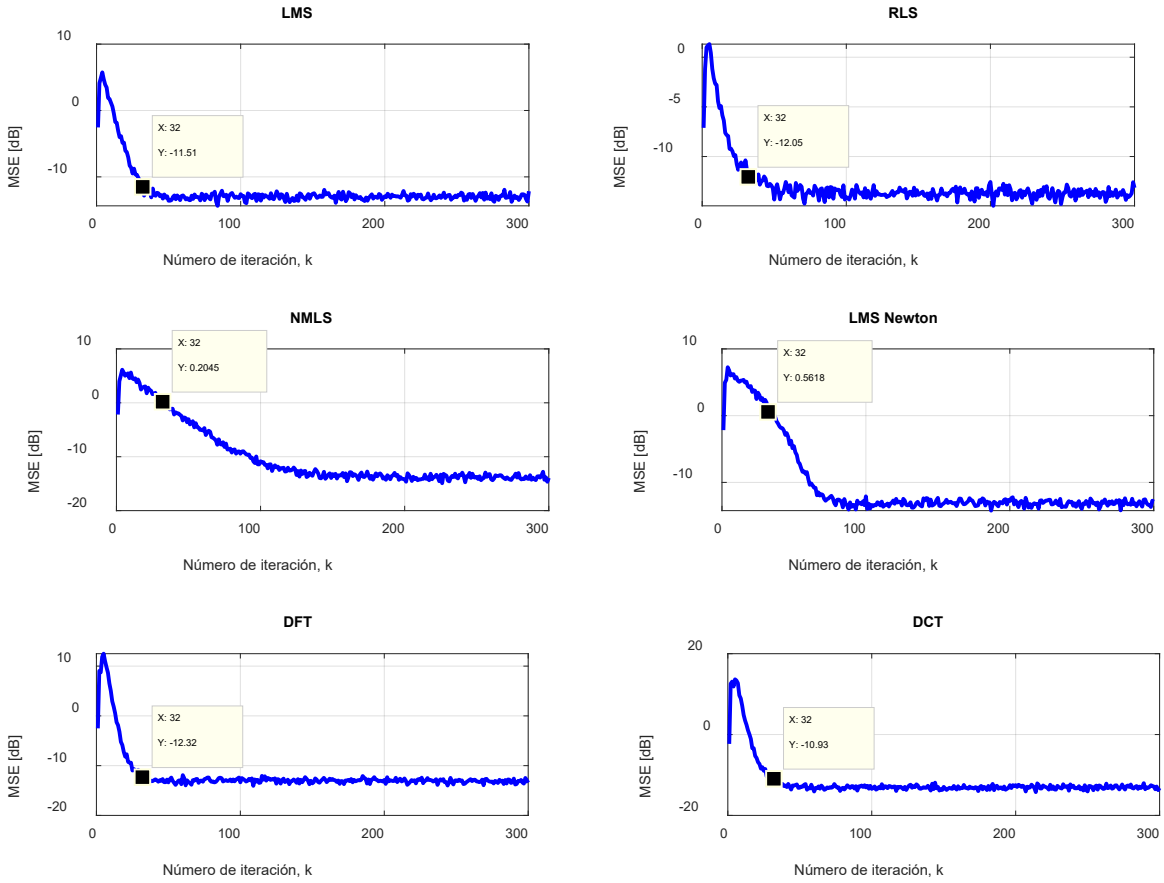


Fig. 154 – Error cuadrático medio (ECM) para distintos algoritmos de sistemas adaptativos

Solicite el código al autor

Ejercicio 4.3

Ejemplo Simple LMS para «Identificación» de Sistemas

Dado el siguiente código para identificación de sistemas, se pide:

a) Analizar su funcionamiento

b) Modificar secuencia de entrada. Comparar respuesta para: $\mu = 0{,}1 * 2$; $\mu = 0{,}01 * 2$; $\mu = 0{,}25 * 2$

c) Escribir conclusiones. Escribir que pasa con el error y la velocidad de convergencia

```
x=[2*ones(1,30)  3*ones(1,30)  1.6*ones(1,30)] ;
largo = length(x) ;
d=10*ones(1,largo) ;
mu=0.1  *2;
N= 1 ; % Cantidad de Coeficientes del filtro
W0=0 ;  % Coeficiente Inicial
S  =  struct('step',mu,'filterOrderNo',(N-1),'initialCoefficients',W0);
  %[y,e,W]  =  LMS(d,transpose(x),S);
[y,e,W]  =  LMS(d,x,S);  %  Solicitar al autor del libro librería LMS
%  Graficamos error MSE
figure,
plot(1:largo, e,'-k', 'linewidth',3); grid on
title('Curva de aprendizaje del error'); axis tight;
xlabel('Número de iteraciones, k'); ylabel('Error');
%  Graficamos salida y
figure,
plot(1:largo, y,'-k', 'linewidth',3); grid on
title('Salida del Sistema:  y');
xlabel('Número de iteraciones, k'); ylabel('Amplitud');
%  Graficamos W
figure;
subplot (211), plot(real(W(1,:)), 'linewidth',3); axis tight; grid on
title('Evolución del primer coeficiente (parte real)');
xlabel('Número de iteraciones, k'); ylabel('Coeficientes');
subplot (212), plot(imag(W(1,:)), 'linewidth',3, 'color','r');axis tight;
title('Evolución del primer coeficiente (parte imaginaria)');
xlabel('Número de iteraciones, k'); ylabel('Coeficientes');
```

En las Fig. 155 y Fig. 156 se muestran los resultados.

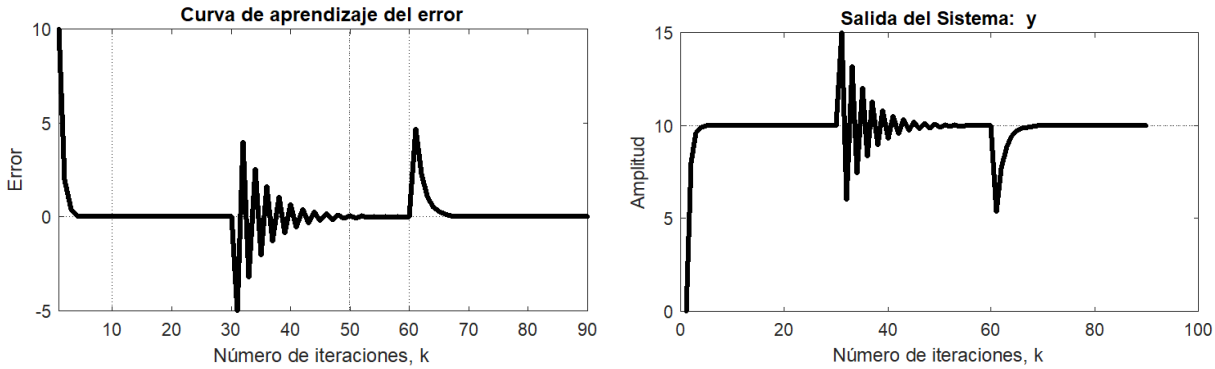
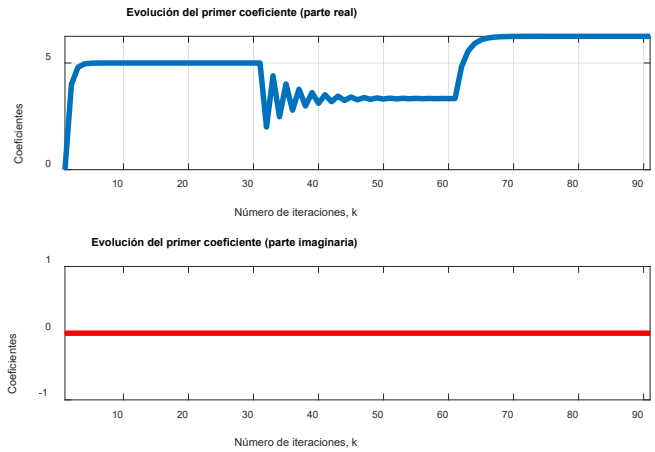


Fig. 155 – Curva de aprendizaje del error y salida del sistema



Solicite el código al autor

Fig. 156 – Evolución del primer coeficiente

Ejercicio 4.4

Cancelación de Ruido de Audio

En la Fig. 157 se muestra un sistema adaptativo para cancelar ruido de un micrófono. Siendo:

$d[n] = ss[n] + n[n]$.

El segundo micrófono se coloca donde solo se capta ruido sin la voz del orador, a la salida del segundo canal ADC tendremos Ruido $x[n]$.

Ruido $x[n]$ con ruido $n[n]$ del primer canal.

Ruido $n[n]$ no correlacionado con la señal $ss[n]$, se puede separar

Ruido $x[n]$ no correlacionado con la señal $ss[n]$, se puede separar

Suponemos que el ruido corruptor en el primer canal es una versión lineal desfasada del ruido del segundo canal, ya que tiene un recorrido físico diferente del ruido del segundo canal, y la fuente de ruido varía en el tiempo, por lo que podemos simular el ruido $n[n]$ usando un filtro lineal desfasador (esto es auxiliar solo para simular ruido).

Este filtro a su salida tendrá una estimación del ruido que llamamos $y[n]$. Esta señal $y[n]$ se restará de la señal corrupta $d[n] = ss[n] + n[n]$. Cuando la estimación de ruido $y[n]$ es igual o similar al ruido

$n[n]$ de la señal corrupta, es decir, $y[n] \approx n[n]$, la señal de error se expresa de la siguiente forma: $\boldsymbol{e[n] = ss[n] + n[n] - y[n] = \widehat{ss}[n]}$. Donde $e[n]$ se aproximará a la señal de voz limpia $ss[n]$. Por tanto, se cancela el ruido.

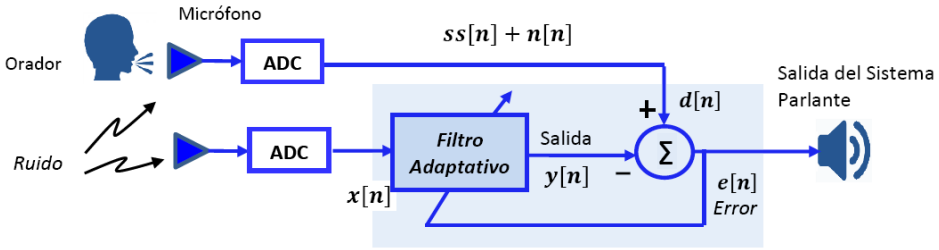


Fig. 157 – Sistema adaptativo para cancelación de ruido de audio

Se desea implementar dicho sistema en Matlab®.

A continuación, mostramos un ejemplo con un archivo de audio de voz y un sistema adaptativo con las siguientes características:

- Frecuencia de muestreo: **11025 Hz** del audio
- Señal de voz corrompida por **ruido gaussiano con una potencia de 1 retardada por 5 muestras** de la referencia de ruido
- Referencia de ruido que contiene ruido gaussiano con una potencia de 1
- Filtro FIR adaptable utilizado para eliminar el ruido, con cantidad de coeficiente (número de tomas de filtro) igual a 19
- Factor de convergencia para el algoritmo LMS elegido: $\boldsymbol{\mu = 0,01}$ **, cumple con ser menor a 1/19**.
- Se grafican las formas de onda de voz y los gráficos espectrales para el ruido original, corrupto y de referencia y para la voz limpia de salida del filtro, señal $e[n]$. A partir de las figuras, se observa que la forma de onda y el espectro mejorados del habla están muy cerca de los originales. El algoritmo LMS converge después de aproximadamente 150 iteraciones. El método es un enfoque muy eficaz para la cancelación de ruido.

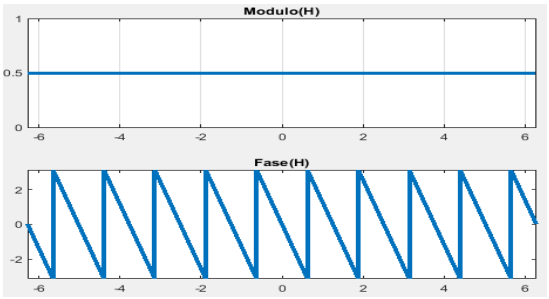


Fig. 158 – Respuesta en frecuencia del Filtro auxiliar, solo para corromper ruido. Arriba) módulo, abajo) fase

Resolución

Utilizar un archivo de audio de corta duración con el nombre: 'Hello.mp3'

```
close all; clear all; clc
%% Se lee un archivo de audio con frecuencia de muestreo: ¡¡fs=11025 !!
 [x1,fs] = audioread('Hello.mp3') ;%
 duracion = fs *3 ;  % 3 segundo
```

```
 % Recorto el audio a 3 segundo
 ss= transpose( x1(1: duracion)) ;
 t= (0:1:length(ss)-1) / fs ; % Vector de tiempo
%%% Generamos ruido distribuido uniformemente
 x=randn(1,length(ss)); % Generamos ruido aleatorio
% Aplicamos ""filtro auxiliar""  para corromper el ruido
% En n obtenemos ruido corrompido !! Cambiamos la fase
  n=filter([ 0 0 0 0 0 0.5 ],1,x); % filtro auxiliar
  freqz([ 0 0 0 0 0 0.5 ],1)
  w= -2*pi: pi/5000:  2*pi;  H= 0.5*exp(-5*i*w)   ;
  figure; subplot(2,1,1);
  plot(w,abs(H), 'linewidth', 3);ylim([0 1]); grid on
  title('Modulo(H)') ; xlim([-2*pi  2*pi])
  subplot(2,1,2);  plot(w,angle(H),'linewidth', 3); axis tight;
  title('Fase(H)') ;
%%% Filtro Adaptativo LMS
 % Inicializamos
 d=ss + n; % Señal d= ss (audio) + ruido
 mu=0.01; % Parmetro mu (step size)
 n_coeficientes = 19 ;
 w=zeros(1, n_coeficientes); % Inicializamos los coeficientes del filtro Adaptativo
 y=zeros(1,length(t)); % Inicializamos en cero vector de salida
 error=y; % Inicializamos vector error (ceros)
 % Filtro Adaptativo con Algoritmo LMS
 for m=  (n_coeficientes+1): length(t)-1
  suma=0;
  for i=1: n_coeficientes
   suma =suma +  w(i)* x(m - i);
  end
  y(m)=suma;
  error(m)=d(m)-y(m);
  for i=1: n_coeficientes
   w(i) =  2*mu*error(m)*x(m - i) + w(i) ;
  end
  % Guardamos histórico de los coeficientes, "solo" para analizar convergencia
  W_matriz_hist(m-n_coeficientes, :)=w ; % Opcional, funciona más lento
 end

% Calculamos y Graficamos los Espectros (single-sided amplitude)
  % Espectro de Señal Original
  SS_FFT = 2* abs(fft(ss))/length(ss);
  SS_FFT(1)=SS_FFT(1)/2;
  % Espectro de:  d= ss + ruido (señal corrupta)
  D = 2 *abs(fft(d))/length(d); D(1)=D(1)/2;
  f=[0:1:length(ss)/2]* fs/length(ss);
  % Espectro de señal con ruido cancelado
  E = 2 * abs(fft(error))/length(error);E(1)=E(1)/2;
%%% Graficamos señales y espectros
 figure;subplot(5,1,1), plot(ss);grid;title('Señal Audio Original: ss');
 subplot(5,1,2),plot(d);grid; title('Audio Corrupto: d')
 subplot(5,1,3),plot(x);grid; title('Ruido de Referencia x');
```

```
 subplot(5,1,4),plot(n);grid; title('Ruido n');
 subplot(5,1,5),plot(error);grid; title('Audio salida, con Ruido eliminado: e');
 xlabel('Numero de muestras');
 figure;  subplot(3,1,1),plot(f,SS_FFT(1:length(f)));grid
 title('Espectro de Audio Original: SS')
 subplot(3,1,2),plot(f,D(1:length(f)));grid; title('Espectro corrompido D')
 subplot(3,1,3),plot(f,E(1:length(f)));grid
 title('Salida - Espectro limpiado E');xlabel('Frequency (Hz)');
% Analizamos convergencia del filtro
 figure; plot(d,'y'); hold on;plot(error,'r'); plot(ss,'b'); axis tight;
 legend('ss + n', 'e', 'ss')
 figure;subplot(211); stem(W_matriz_hist(end,:));
 subplot(212);plot(W_matriz_hist(:,5))
 %% Correlación las señales
 dt =t(2) -t(1) ;
 [Rss, tau] = xcorr( ss, ss) ;
 figure; subplot(3,2,1)
 plot(tau*dt, Rss*dt); axis tight;grid on
 title('Rss: Autocorrelación Señal Original ss')
 [Rdd, tau] = xcorr( d, d) ;
 subplot(3,2,2)
 plot(tau*dt, Rdd*dt); axis tight;grid on
 title('Rdd: Autocorrelación Señal d')
 [Rxx, tau] = xcorr(x , x) ;
 subplot(3,2,3)
 plot(tau*dt, Rxx*dt); axis tight; grid on
 title('Rxx: Autocorrelación del ruido x')
 [Rnn, tau] = xcorr(n , n) ;
 subplot(3,2,4)
 plot(tau*dt, Rnn*dt); axis tight; grid on
 title('Rnn: Autocorrelación del ruido n')
 [Ree, tau] = xcorr( error, error) ;
 subplot(3,2,5)
 plot(tau*dt, Ree*dt); axis tight;grid on
 title('Ree: Autocorrelación Señal error')
 [Rse, tau] = xcorr( ss, error) ;
 subplot(3,2,6)
 plot(tau*dt, Rse*dt); axis tight;grid on
 title('Rse: Correlación entre ss (audio) y error')
 %[Rxn, tau] = xcorr( x, n) ;
 %figure; plot(tau*dt, Rxn*dt); axis tight;grid on
 %title('Rxn: Correlación entre x y n')
%% Reproducimos las señales
 sound(ss,fs) ; % señal original de audio
 pause(3)
 sound(d,fs) ; % Señal con ruido corrupto
 pause(3)
 sound(error,fs) ; % Señal con ruido cancelado
 pause(3)
```

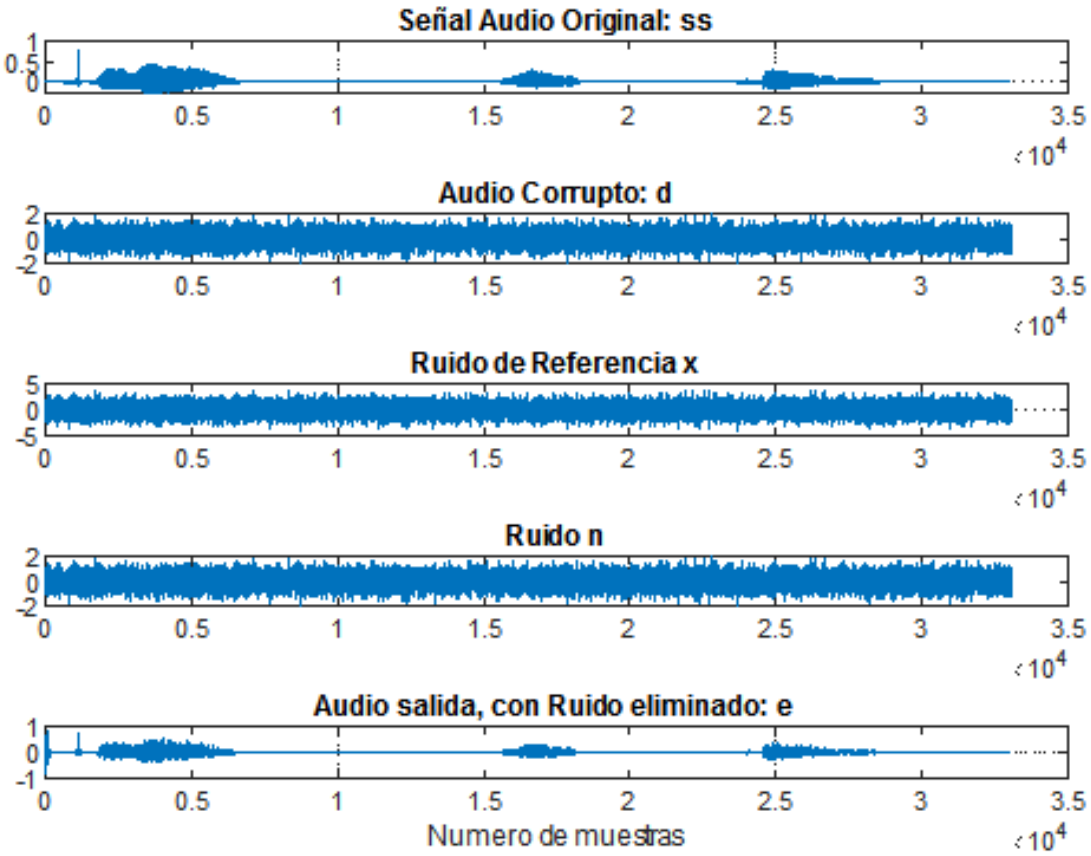


Fig. 159 – Señales temporales, original, con ruido y con ruido eliminado.

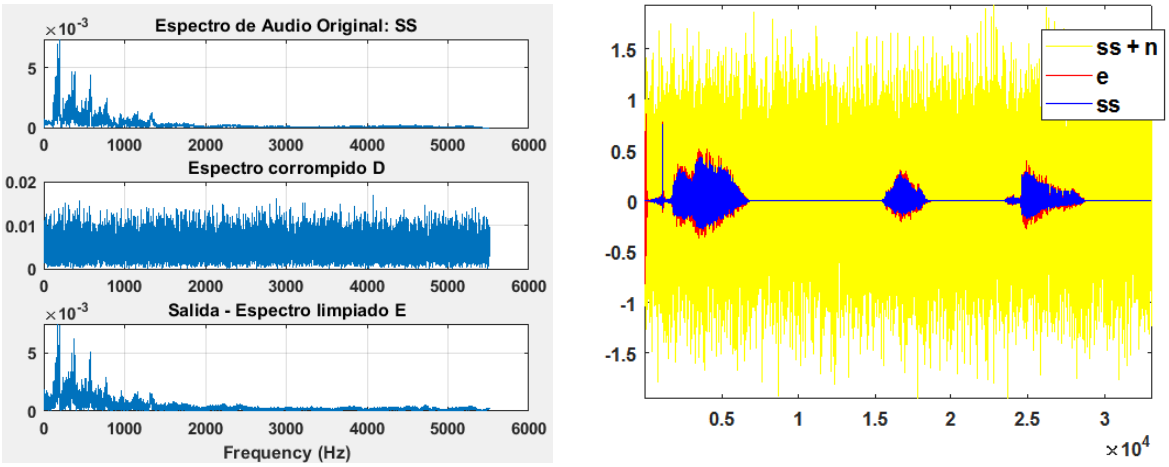


Fig. 160 – Izquierda) Espectros de las señales, derecha) señales temporales con vista ampliada

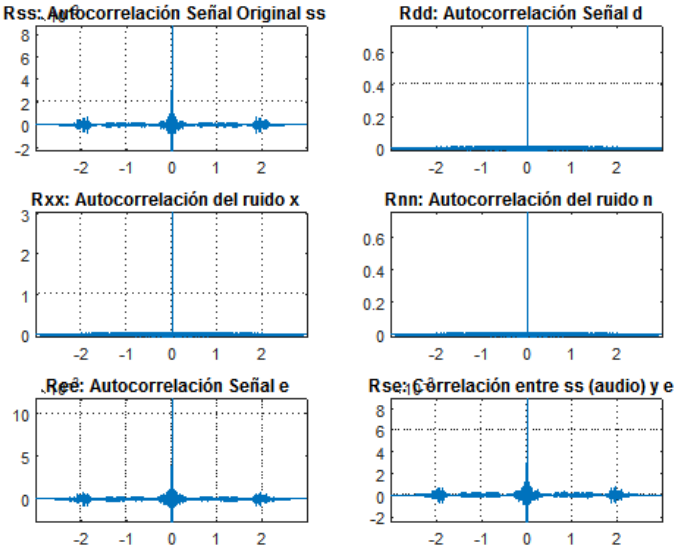


Fig. 161 – Autocorrelación de las señales

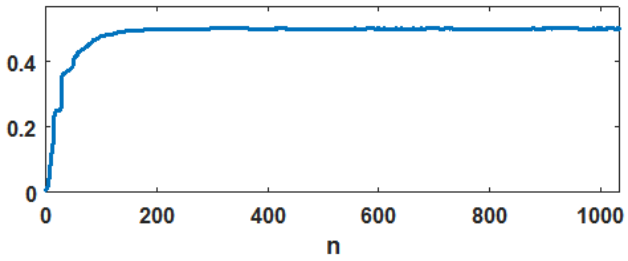


Fig. 162 – Evolución del coeficiente de adaptación

Ejercicio 4.5

Cancelación de Ruido en señal Triangular

Implementar un sistema adaptativo en Matlab® para eliminar el ruido en una señal triangular ruidosa

Resultados

Ejercicios correspondientes al seminario doctoral. Contactarse con el autor del libro para consultas y detalles de algoritmos.

En la Fig. 163 y Fig. 164 se muestran los resultados del algoritmo para una función triangular

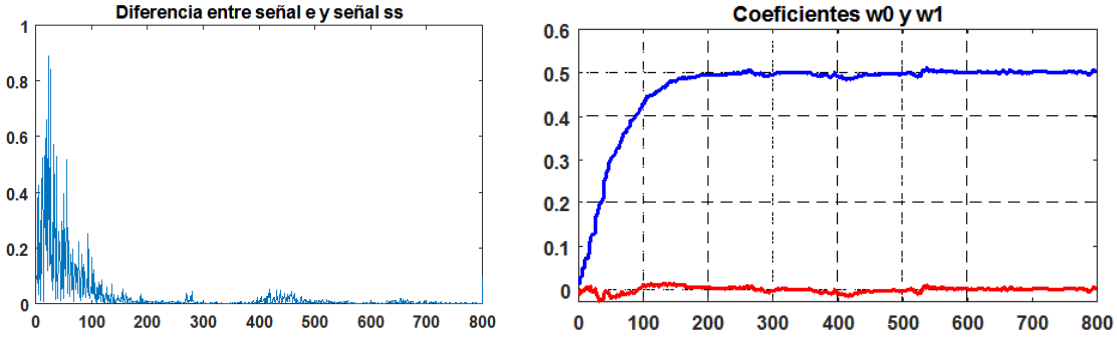
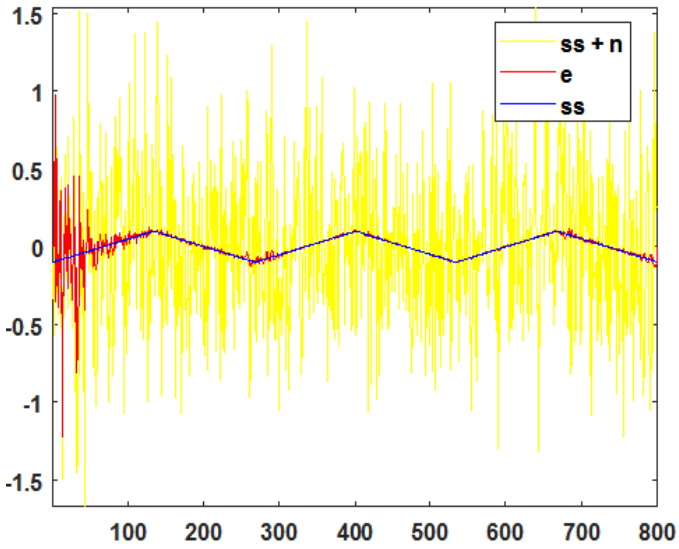


Fig. 163 – Error y evolución de los coeficientes



Solicite el código al autor

Fig. 164 – Respuestas del filtro adaptativo.
ss: señal triangular , e: señal de error, n: señal de ruido

Ejercicio 4.6

Ejemplo de «Identificación» de filtro IIR

Mediante Matlab® se desea implementar el sistema de la Fig. 165 correspondiente a un filtro adaptativo para la Identificación de sistemas desconocidos.

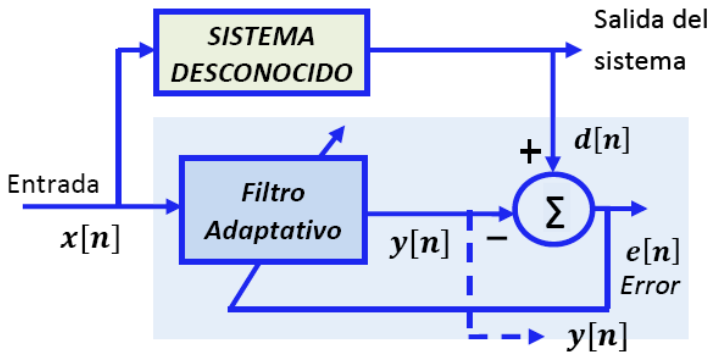


Fig. 165 – Sistema adaptativo para identificación

Se pide Implementar en Matlab® un sistema adaptativo para reconocer el siguiente sistema

Sistema desconocido: **filtro IIR** de paso de banda de cuarto orden con frecuencias de corte superior e inferior con atenuación de 3 dB son 1.400 Hz y 1.600 Hz operando a 8.000 Hz. Frecuencia central del filtro pasa banda: 1500 Hz.

Usamos una **entrada $x[n]$ que consta de tonos de 500, 1.500 y 2.500 Hz.**

La salida del sistema desconocido contendrá solo un tono de 1.500 Hz, ya que los otros dos tonos son rechazados por el sistema desconocido.

Filtro FIR adaptativo con N=19 coeficientes (número de entradas) y un factor de convergencia $\mu = 0.01$. En el dominio temporal, la señal de salida correspondientes al sistema desconocido $d[n]$ y la salida del filtro adaptativo $y[n]$ son muy similares luego de 70 iteraciones del algoritmo LMS. También se grafica la señal de error $e[n]$, en todo momento el sistema busca minimizar este error. Si el sistema desconocido se modifica, o si la entrada cambia, el sistema seguirá buscando el mínimo error $e[n]$.

Resolución

```
%% Diseño del sistema desconocido
 fs = 8000; T = 1/fs; % frecuencia de muestreo y tiempo de muestreo
% Diseño filtro pasa banda IIR. Utilizamos transformación bilineal (BLT)
% Mediante BLT pasamos de T. Laplace a TZ
 wd1 = 1400*2*pi; wd2 = 1600*2*pi;
 wa1 = (2/T)*tan(wd1*T/2); wa2 =(2/T)*tan(wd2*T/2);
 BW=wa2-wa1;
 w0 = sqrt(wa2*wa1);
 [B,A]=lp2bp([1],[1 1.4141 1],w0,BW); % transforma de pasa bajo a pasa banda (help lp2bp)
 [b,a]=bilinear(B,A,fs); % Transformación bilineal
 freqz(b,a,512,fs); axis([0 fs/2 -80 1]); % Graficamos su respuesta en frecuencia.
%% Diseño del Filtro Adaptativo.
% Armamos una señal temporal con 3 tonos puros
 t=0:T:0.1 ;
 x = cos (2*pi*500*t) + sin (2*pi*1500*t) + cos (2*pi*2500*t + pi/4);
% Generamos la salida del sistema desconocido (d).
 d=filter( b , a , x ) ;
 mu= 0.01 ; % Factor de convergencia
 n_coeficientes = 19 ; % Cantidad de coeficientes del filtro
% Inicializamos los coeficientes w, la salida y el error
 w=zeros(1, n_coeficientes); y=zeros(1,length(t));
 error =y;
```

```
% Algoritmo LMS
inicio = n_coeficientes+1 ;
 for m= inicio :length(t)-1
  suma=0;
 for i=1:n_coeficientes
  suma = suma + w(i)*x(m -i);
 end
  y(m)=suma;
  error(m) = d(m)-y(m);
  for i=1:n_coeficientes
    w(i) = 2*mu*error(m)*x(m-i) + w(i) ;
  end
 end
% Calculamos el espectro de la entrada
 X = 2*abs(fft(x))/length(x); X(1)=X(1)/2;
% Calculamos el espectro de la salida del sistema desconocido
 D = 2*abs(fft(d)) /length(d);  D(1)=D(1)/2;
% Calculamos el espectro de la salida del filtro adaptativo
 Y = 2*abs(fft(y))/length(y); Y(1)=Y(1)/2;
% Vector de frecuencias en Hz
 f =[0:1:length(x)/2]*fs/length(x);
% Graficamos señales temporales y espectros
 figure; subplot(4,1,1),
 plot(x);grid;axis([0 length(x) -3 3]);
 title('Entrada del sistema: x[n]');
 subplot(4,1,2), plot(d);grid;axis([0 length(x) -1.5 1.5]);
 title('Salida del sistema: d[n]');
 subplot(4,1,3),plot(y);grid;axis([0 length(y) -1.5 1.5]);
 title('Salida del Filtro Adaptativo: y[n]')
 subplot(4,1,4),plot(error);grid;axis([0 length(error) -1.5 1.5]);
 title('Error'); xlabel('Número de muestra')
 figure
 subplot(3,1,1),plot(f,X(1:length(f)));grid;title('Espectro de la entrada: X')
 subplot(3,1,2),plot(f,D(1:length(f)));grid;title('Espectro de la salida: D')
 subplot(3,1,3),plot(f,Y(1:length(f)));grid
```

```
  title('Espectro de la salida del filtro: Y');xlabel('Frequency (Hz)');
% w contiene coef.  del filtro Adaptativo
  figure;freqz(w,1,512,fs);
  title('Respuesta del filtro Adaptativo: PREDECIDO');
  xlabel('Frecuencia (Hz)');
```

En las Fig. 166, Fig. 167y Fig. 168 se muestran los gráficos obtenidos. En la Fig. 168 se observa que el sistema adaptativo predijo correctamente la frecuencia central del filtro pasa banda de 1500 Hz.

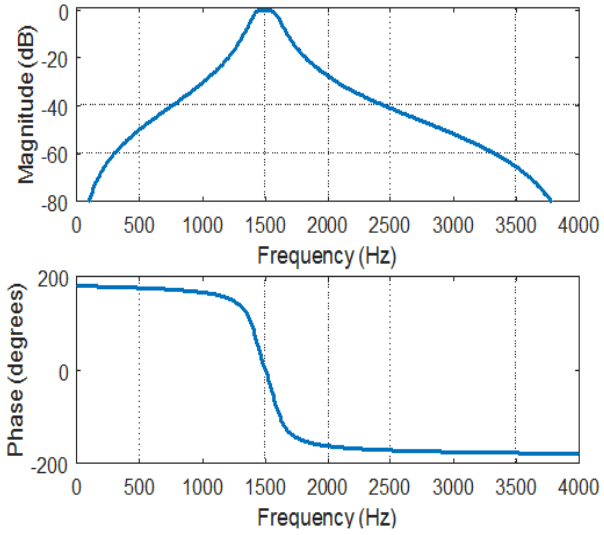


Fig. 166 –Respuesta en frecuencia del sistema desconocido (filtro IIR)

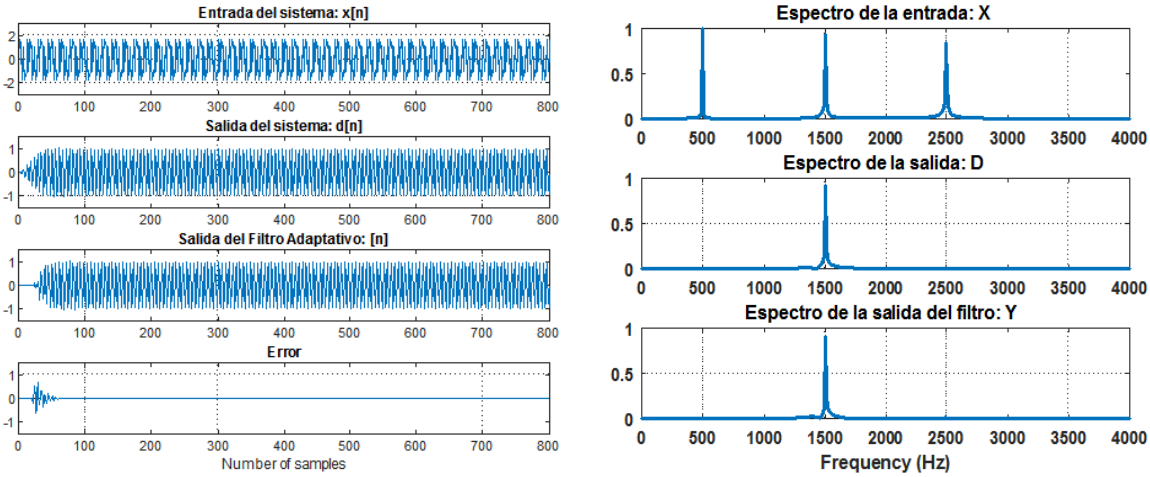


Fig. 167 – Señales temporales y Espectros original y filtrado de la salida del sistema

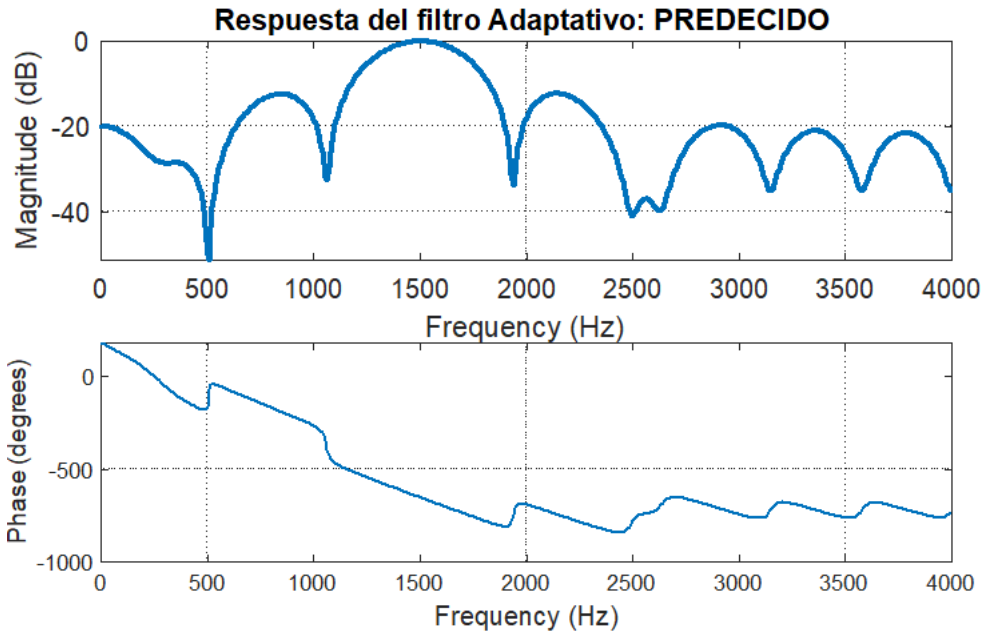


Fig. 168 – Respuesta en frecuencia del filtro adaptativo predicho

Ejercicio 4.7

Ejemplo de Mejora de Línea

Si el contenido de la frecuencia de una señal es muy estrecho en comparación con el ancho de banda total y se modifica con el tiempo, entonces la señal se puede mejorar de manera eficiente mediante un filtro adaptativo, esta técnica se llama mejora de línea.

La señal $d[n]$ es señal sinusoidal corrompida por el ruido blanco gaussiano $n[n]$.

La línea mejorada consiste en el elemento de retardo para retardar la señal corrupta por D muestras para generar la entrada al filtro adaptativo. **El filtro adaptativo es en realidad un predictor lineal de la señal de banda estrecha deseada.** Un filtro FIR adaptativo de dos tomas de muestra puede predecir una sinusoide. El valor de D generalmente se determina en forma práctica. Estas técnicas se utilizan muchas veces en sistemas de transmisión donde se tiene una portadora sinusoidal

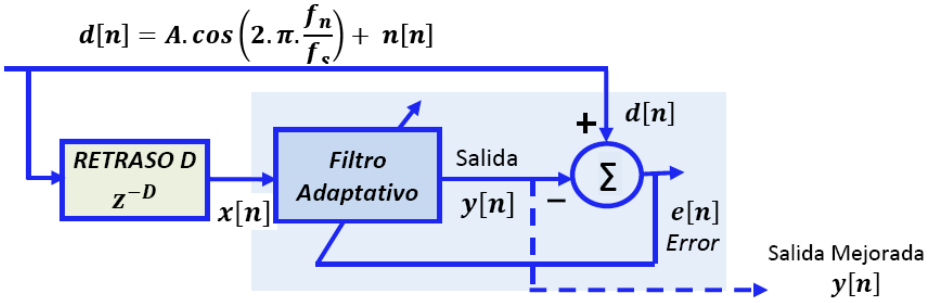


Fig. 169 – Ejemplo de mejora de línea

Implementar un ejemplo en Matlab® con las siguientes especificaciones:

- Frecuencia de muestreo: 8000 Hz
- Señal corrupta tono de 400 Hz con amplitud de unidad agregada con blanco

- Ruido gaussiano
- Filtro adaptativo tipo FIR, 19 tomas
- Factor de convergencia $\mu = 0{,}001$
- Valor de retardo D = 7
- Algoritmo LMS

Resolución

```
% Frecuencia de muestreo y período y vector temporal de 1 seg.
 fs = 8192; T = 1/fs;
 t = 0:T:0.1;
% Ruido Blanco Gaussiano
 n =1*randn(1,length(t));
% Tono de 400 Hz + ruido
 d = 1*cos(2*pi*400*t) + n;
% Filtramos la señal con filtro de retardo
 x = filter([0 0 0 0 0 0 0 1], 1, d);
% Inicializamos el Algoritmo LMS
 mu = 0.001;
 n_coeficientes = 19 ;
 w = zeros(1, n_coeficientes);
 y = zeros(1,length(t)); % y: Salidas del filtro
 e = y; % vector de error
% Algoritmo LMS
   inicio = n_coeficientes+1 ;
   for m = inicio :1:length(t)-1
    suma = 0;
    for i = 1:1:n_coeficientes
     suma = suma + w(i)*x(m-i);
    end
    y(m) = suma;
    error(m) = d(m)-y(m);
    for i = 1: n_coeficientes
     w(i) =  2*mu*error(m)*x(m-i) + w(i) ;
    end
   end
```

```
% Calculamos los espectros de la señal corrupta y de la señal mejorada Y
 D = 2*abs(fft(d))/length(d);D(1) = D(1)/2;
 Y = 2*abs(fft(y))/length(y);Y(1) = Y(1)/2;
% Graficamos
 f=[0:1:length(x)/2]*8000/length(x); % Eje de frecuencias en Hz
 figure;subplot(2,1,1),plot(d);grid;axis([0 length(x) -2.5 2.5]);title('Señal d ruidosa');
 subplot(2,1,2),plot(y);grid;axis([0 length(y) -2.5 2.5]);
 title('Salida y (señal mejorada)');xlabel('Numero de muestra')
 figure ; subplot(2,1,1),plot(f,D(1:length(f)),'linewidth',2);
 grid;axis([0 fs/2 0 1.5]);
 title('Espectro de señal ruidosa: D')
 subplot(2,1,2),plot(f,Y(1:length(f)), 'linewidth',2);
 grid;axis([0 fs/2 0 1.5]);
 title('Espectro de la salida: Y');xlabel('Frecuencia(Hz)');
```

En la Fig. 170 y Fig. 171 se muestran los resultados obtenidos en Matlab®

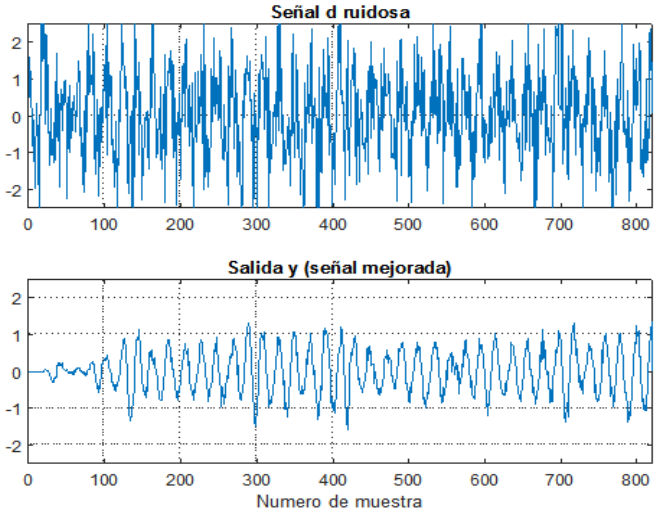


Fig. 170 –Señales temporales. Arriba) señal ruidosa, abajo) señal mejorada.

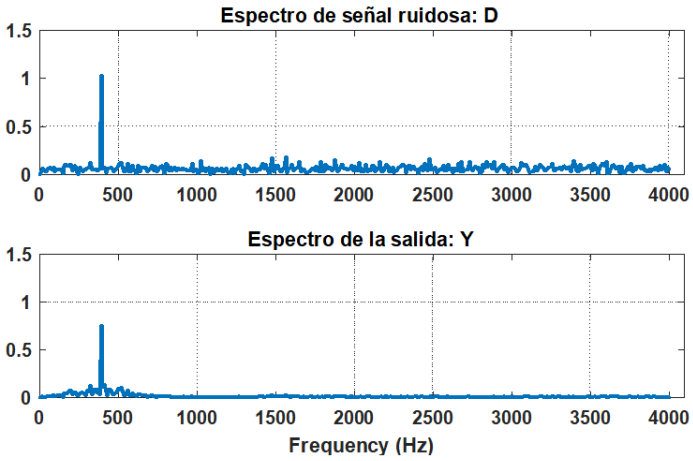


Fig. 171 –Gráfico de espectros de las señales

Ejercicio 4.8

Implementar una interfaz gráfica de usuario (GUI) para analizar señales mediante algoritmo LMS

Resultados

Ejercicios correspondientes al seminario doctoral. Contactarse con el autor del libro para consultas y detalles de algoritmos.

En la Fig. 172 se muestran los resultados del algoritmo para una función triangular modulada con un coseno. La línea roja corresponde a la señal de salida del sistema adaptativo, se observa una correcta adaptación.

El código de la interfaz gráfica es compatible con Matlab® y Octave.

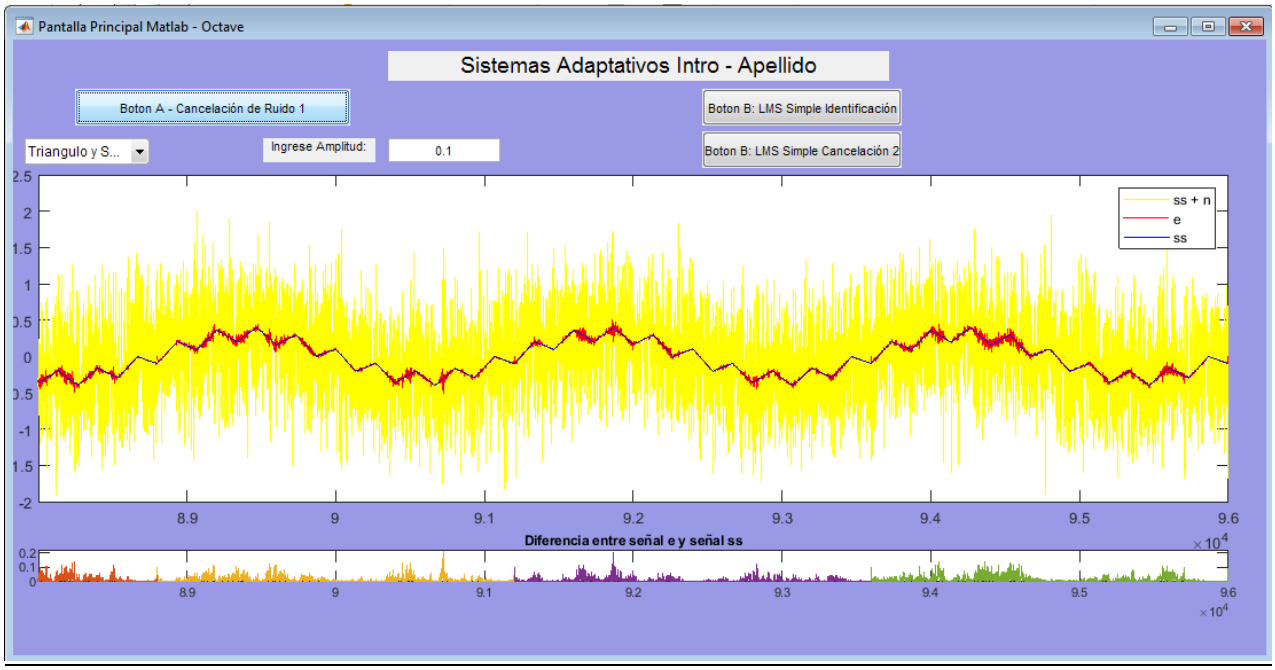


Fig. 172 – Interfaz gráfica implementada en Matlab® para analizar sistemas adaptativos

Solicite el código al autor

Ejercicio 4.9

Ejemplo en Simulink para comparar diferentes algoritmos LMS

Dado el siguiente programa en Matlab® con herramienta Simulink DSP, se pide analizar su funcionamiento.

Escribir en línea de comandos: lmsxyplot

En la Fig. 173 se observa el diagrama Simulink del programa y en la Fig. 175 se muestran los resultados de los coeficientes μ, donde se observa la convergencia de los 4 algoritmos.

Algoritmos utilizados:

NLMS – Algoritmo LMS Normalizado con $\mu = 0{,}005$

NLMS – Algoritmo LMS Normalizado con $\mu = 0{,}05$

SELMS - Algoritmo LMS error signo con $\mu = 0{,}005$

SSLMS - Algoritmo LMS signo con $\mu = 0{,}005$

Sin embargo, si se duplican los 4 valores de factor de convergencia μ, se obtienen los resultados graficados en la Fig. 176. Para los algoritmos SELMS y SSLMS no se tiene una buena convergencia. Mientras los algoritmos LMS y LMS convergen más rápido

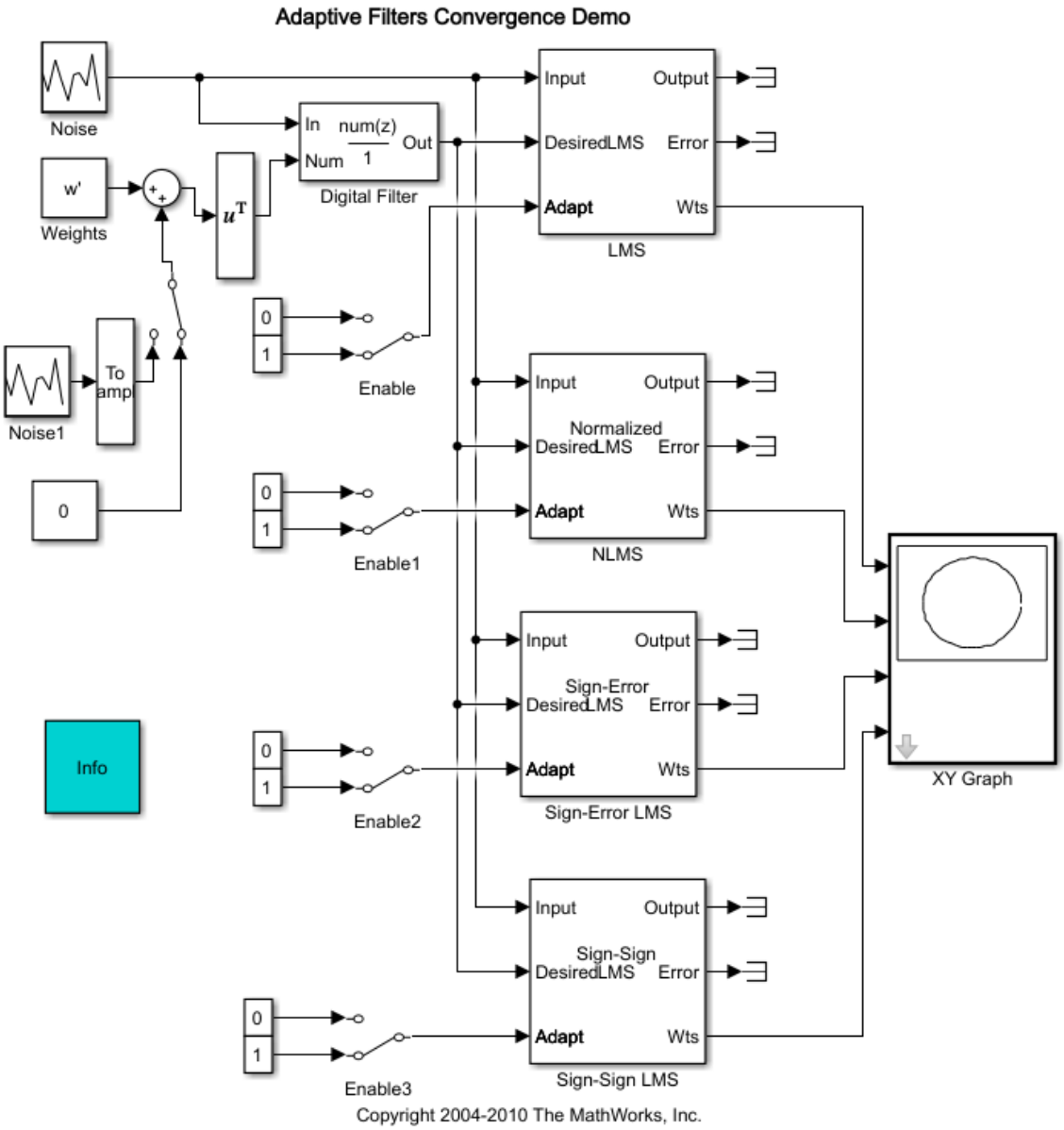


Fig. 173 – Diagrama simulink para análisis de filtros adaptativos LMS

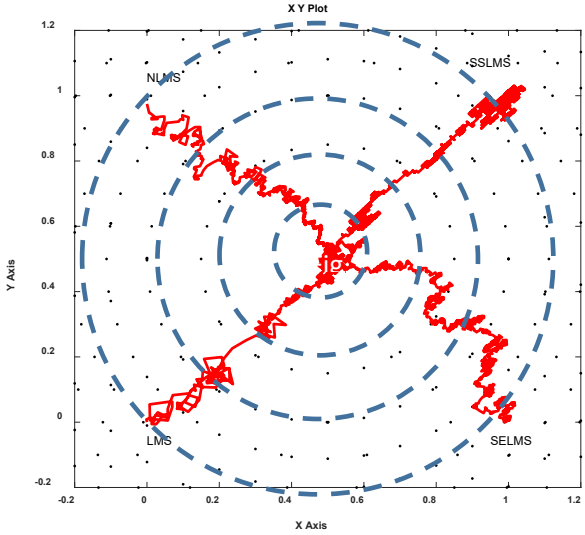


Fig. 175 – Resultados del programa Simulink LMS con coeficientes μ originales

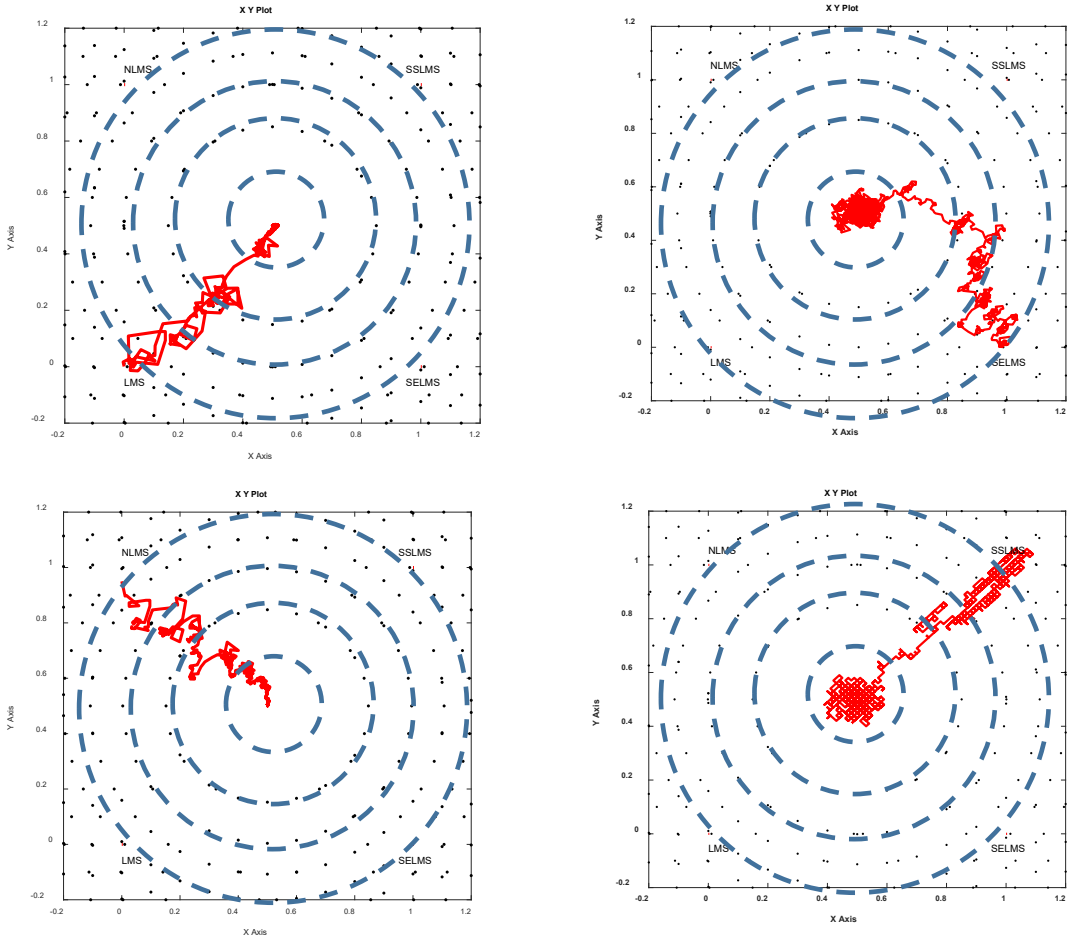


Fig. 176 – Resultados del programa Simulink LMS duplicando los valores de los coeficientes μ

Ejercicio 4.10

Ejemplo de cancelación de ruido con filtro RLS y herramienta DSP

Se desea cancelar ruido en señal compuesta por 2 señales senoidales según el esquema mostrado en la Fig. 177, se utiliza la herramienta DSP de Matlab®

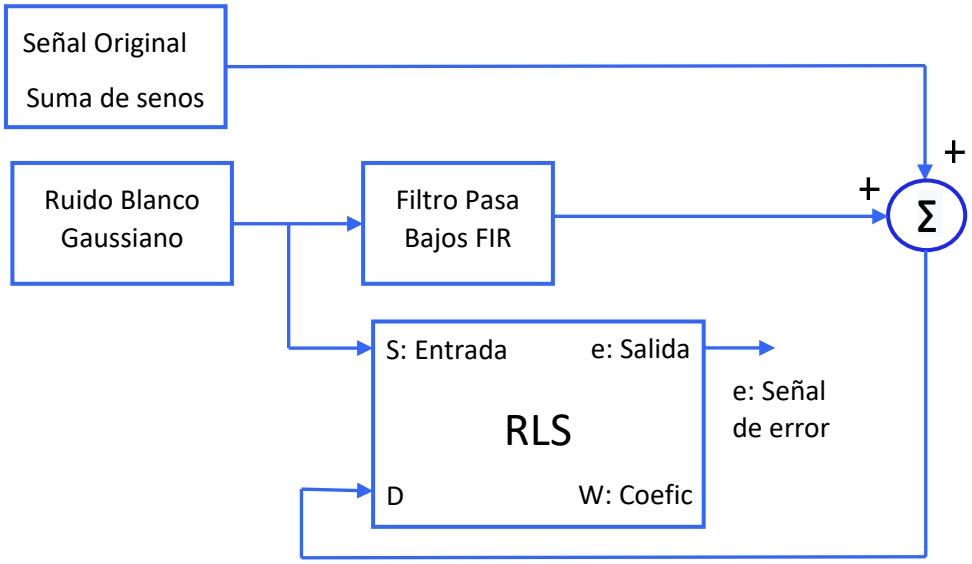


Fig. 177 – Cancelación de ruido mediante filtro RLS con herramienta DSP

Resolución

```
% Ejemplo de cancelación de ruido con filtro RLS y herramienta DSP
% Generamos señales y fuentes de señales
 % Señal S: suma de señales senoidales
 s1 =sin(2*pi*0.04*(0:1000-1)')+(0.7*cos(2*pi*0.0012*(0:1000-1)'));
 signalSource = dsp.SignalSource(s1,'SamplesPerFrame',120,'SignalEndAction','Cyclic repetition');
 noise = randn(1000,1)*1.1;  % Ruido Blanco con varianza 0.7
 noiseSource = dsp.SignalSource(noise,'SamplesPerFrame',120,'SignalEndAction','Cyclic repetition');
% Graficamos
 figure;  subplot(211)
 plot(0:999,s1(1:1000), 'linewidth',2); xlabel('Número de muestra: k');
 title('Señal de información s');  grid on
 subplot(212) ;
 plot(0:999,noise+s1, 'linewidth',1.2); xlabel('Número de muestra: k');
 title('Señal d = s+ n (n:Ruido del micrófono secundario)');
 grid; axis([0 999 -4 4]);
```

```
% Filtro FIR Pasa bajo y filtro RLS
 pb1 = dsp.FIRFilter('Numerator',fir1(29,[0.5]));
 delta = 0.11; % Estimación de la covarianza de entrada inicial
 M = 30;      % Orden del filtro
 P0 = (1/delta) *eye(M,M); % Configuración inicial para la matriz P
 rlsfilt = dsp.RLSFilter(M,'InitialInverseCovariance',P0);
% Gráfico dinámico
 scope = dsp.TimeScope('TimeSpanSource','property','TimeSpan',2400, 'YLimits',[-2,2]);
% Medimos en tiempo real
   ciclos =20;
   for k = 1:ciclos
      n = noiseSource(); % fuente de ruido
      s = signalSource(); % fuente de señal
      d = pb1(n) + s;
      [y,e]  = rlsfilt(n,d);
      scope([s,e,n]); % graficamos
   end
   release(scope)
% Graficamos último conjunto de señales
   figure; k= ((120*(ciclos-1)):(120*ciclos-1)) ;
   plot(k,s,'linewidth',2); xlabel('Número de Muestra');hold on,
   plot(k,d,'linewidth',1.2); plot(k,e,'linewidth',2); grid on
   legend('señal s','señal d','señal e'); xlabel('Número de muestra: k')
% Respuesta en frecuencia
   H  = abs(freqz(rlsfilt.Coefficients,1,64));
   H1 = abs(freqz(pb1.Numerator,1,64));
   wf = linspace(0,1,64);
   figure
   plot(wf,H,wf,H1,'linewidth',2);
   xlabel('Frecuencia Normalizada  (\times\pi rad/muestra)');
   ylabel('Magnitud');
   legend('Respuesta del filtro Adaptativo' , 'Respuesta de filtro Pasa Bajos');
   grid;  axis([0 1 0 1.5]);
```

En la Fig. 178 y Fig. 179 se muestran los resultados obtenidos. Se observa una correcta adaptación teniendo en cuenta que la señal original (s) y la señal del sistema (e) son similares, a pesar de tener una señal d muy contaminada con ruido. Se debe tener en cuenta que el ruido no se encuentra correlacionado con la señal de información.

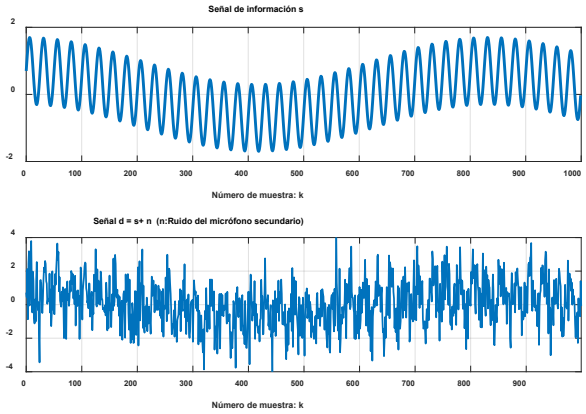


Fig. 178 – Señal original s y señal con ruido

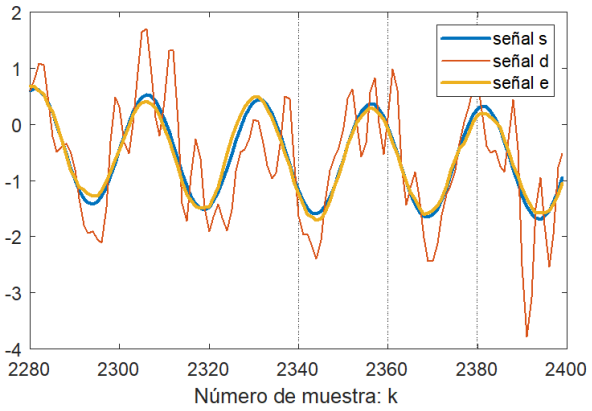


Fig. 179 – Señales temporales s, d y e

Ejercicios en Python

Ejercicio 4.11

Comparación de 9 Algoritmos adaptativos para identificación de sistemas con señal chirp

Se pide analizar y probar el siguiente programa

```
"""
Comparación de Algoritmos adaptativos.
Ejemplo de Identificación de señal chirp con ruido agregado. La señal chirp modifica su frecuencia con el tiempo
Utilizamos librería padasip
  pip install padasip
Juan Vorobioff
"""
import numpy as  np # importamos
import padasip as pa #
import matplotlib.pylab as plt
def Graficar(y, e, w , titulo):
   # Mostramos los resultados
   plt.rc('font', size=14) ; plt.rc('axes', titlesize=18)
   fig, (ax1, ax2) =  plt.subplots(2, 1, figsize=(8,6))
   ax1.set_title(titulo); ax1.set_xlabel("Número de muestra - k")
   ax1.plot(d,"b", label="d: salida deseada")
   ax1.plot(y,"y", label="y: salida del sistema"); ax1.legend(loc='upper right')
   ax1.legend(loc='upper right', bbox_to_anchor=(1, 1.4))
   ax1.grid(True) ;  ax2.set_title("Error del filtro");
   ax2.set_xlabel("Número de muestra - k")
   plt.plot(e,"r");  plt.grid()
   plt.tight_layout()
   plt.show()
   ecm = sum(e**2) / largo  ; print(ecm)
   return ecm;
######### Generamos datos y graficamos
largo = 302 ;
from scipy.signal import chirp, spectrogram
```

```
t = np.linspace(0, 10, largo) # vector de tiempo
y = chirp(t, f0 = 1, f1 = 5, t1 = 10 ,  method='linear')
plt.rc('font', size=14) ; plt.rc('axes', titlesize=16)
plt.plot(t,y) ; plt.title("Función Lineal Chirp, frec. inicial: 1, frec. final: 5")
plt.xlabel('t [seg.]') ; plt.show()
n=3  # tamaño del filtro
# Generamos matriz de datos x1 de tamaño: largo x 3
# Donde se agregan columnas desfasadas
x1 = pa.input_from_history(y, n)
# Ruido gaussiano para agregar
np.random.seed(seed=1)
v1 = np.random.normal(0, 0.2, largo-n+1)
# Señal original s y Señal deseada: d
s1 = 3*x1[:,0]
d =  s1 + v1   # agregamos ruido
#############################################################################
######### Algoritmo LMS
ecm = np.zeros(9) # Definimos array de error cuadrático medio
n=3  # número de columnas
# Algoritmo LMS convencional (LMS)
# Generamos el filtro
  # Clase: padasip.filters.lms.FilterLMS(n, mu, w)
  # n: largo del filtro (filas), mu: taza de aprendizaje, w:pesos iniciales del filtro
filtro1 = pa.filters.FilterLMS(n, mu=0.05, w="random")
# Identificación
  # run(d, x)   ; d: matriz de 1 dimensión, x: matriz de 2 dimensiones
y, e, w = filtro1.run(d, x1)
ecm[0]=Graficar(y, e, w ,"LMS")
######### Algoritmo LMS Normalizado (NLMS)
filtro2 = pa.filters.FilterNLMS(n, mu=0.15, w="random")
y, e, w = filtro2.run(d, x1)
ecm[1]=Graficar(y, e, w , "NLMS")
######### Algoritmo de Proyección afín (AF)
filtro3 = pa.filters.FilterAP(n, order=5, mu=0.5, eps=0.001, w="random")
y, e, w = filtro3.run(d, x1)
```

```
ecm[2]=Graficar(y, e, w ,"AF")
######### Algoritmo Generalized Normalized Gradient Descent (GNGD)
filtro4 = pa.filters.FilterGNGD(n, mu=0.1, w="random")
y, e, w = filtro4.run(d, x1)
ecm[3]=Graficar(y, e, w , "GNGD")
######### Algoritmo Least-mean-fourth (LMF)
# ver detalles del algoritmo en publicación [1]
#[1] Azzedine Zerguine.  Convergence behavior of the normalized least
#mean fourth algorithm. In Signals, Systems and Computers, 2000. Conference
#Record of the Thirty-Fourth Asilomar Conference on, volume 1, 275–278. IEEE, 2000.
filtro5 = pa.filters.FilterLMF(n, mu=0.06, w="random")
y, e, w = filtro5.run(d, x1)
ecm[4]=Graficar(y, e, w , "LMF")
######### Algoritmo Normalized Least-mean-fourth (NLMF)
# Ver publicación [1]
filtro6 = pa.filters.FilterNLMF(n, mu=0.06, w="random")
y, e, w = filtro6.run(d, x1)
ecm[5]=Graficar(y, e, w ,"NLMF")
######### Algoritmo Normalized Sign-sign Least-mean-squares (NSSLMS)
filtro7 = pa.filters.FilterNSSLMS(n, mu=0.1, w="random")
y, e, w = filtro7.run(d, x1)
ecm[6]=Graficar(y, e, w ,"NSSLMS")
######### Algoritmo Sign-sign Least-mean-squares (SSLMS)
filtro8 = pa.filters.FilterSSLMS(n, mu=0.09, w="random")
y, e, w = filtro8.run(d, x1)
ecm[7]=Graficar(y, e, w ,"SSLMS")
######### Algoritmo Recursive Least Squares (RLS)
filtro9 = pa.filters.FilterRLS(n, mu=0.25, w="random")
y, e, w = filtro9.run(d, x1)
ecm[8]=Graficar(y, e, w ,"RLS")
######### Graficamos Error cuadrático medio
# Definimos lista de algoritmos
Algoritmos = ['LMS','NLMS','AFÍN','GNGD','LMF','NLMF','NSSLMS','SSLMS','RLS']
plt.rc('font', size=18)
fig, ax = plt.subplots(figsize=(12,6))
```

```
ax.set_ylabel('Error cuadrático medio')
#Creamos gráfico de barras ecm
plt.bar(Algoritmos, ecm)
plt.grid(axis='y' ,color='g', linestyle=':', linewidth=1)
plt.savefig('Algoritmos de Identificación 02.png')
plt.show()

######### Graficamos evolución de algunos coeficientes
fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2, figsize=(10,6))
ax1.plot(filtro1.w_history,"b") ;ax1.set_title("LMS");  ax1.grid(True)
ax1.set_xlabel("Número de muestra: k")
ax2.plot(filtro3.w_history,"b") ;ax2.set_title("Proyección Afín"); ax2.grid(True)
ax2.set_xlabel("Número de muestra: k")
ax3.plot(filtro5.w_history,"b") ; ax3.set_title("LMF"); ax3.grid(True)
ax3.set_xlabel("Número de muestra: k")
ax4.plot(filtro9.w_history,"b") ; ax4.set_title("RLS"); ax4.grid(True)
ax4.set_xlabel("Número de muestra: k")
fig.suptitle("Evolución de los coeficientes")
fig.tight_layout() ; fig.subplots_adjust(top=0.86)
```

En la Fig. 180 y Fig. 181 se muestran las señales deseadas y las señales predecidas para los algoritmos LMS, NLMS, LMF y RLS. Se observa una rápida convergencia para los algoritmos LMS y RLS.

En la Fig. 182 se muestra el error cuadrático medio, se observa que con los parámetros utilizados los algoritmos de proyección afín y RLS presentan menos error. En la Fig. 183 se muestra la evolución de los coeficientes para 4 algoritmos.

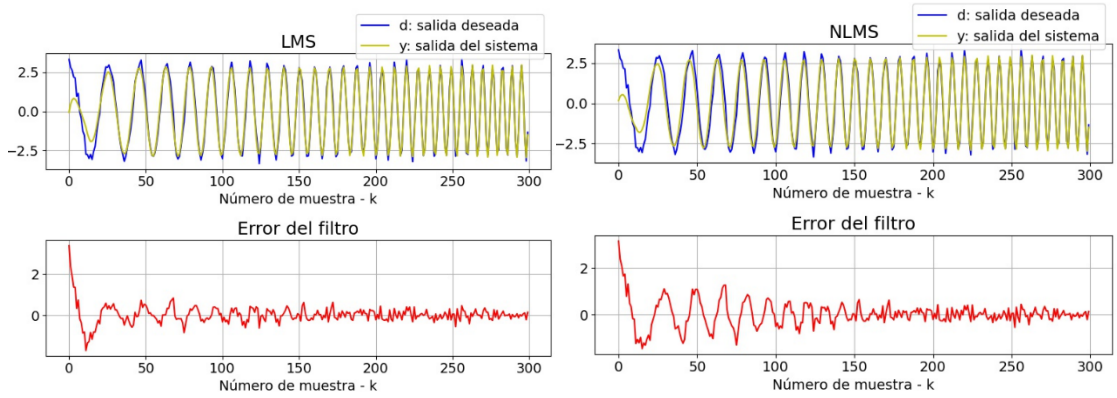


Fig. 180 – Adaptación del filtro: respuestas temporales y errores para función chirp. Izq.) LMS, derecha) NLMS

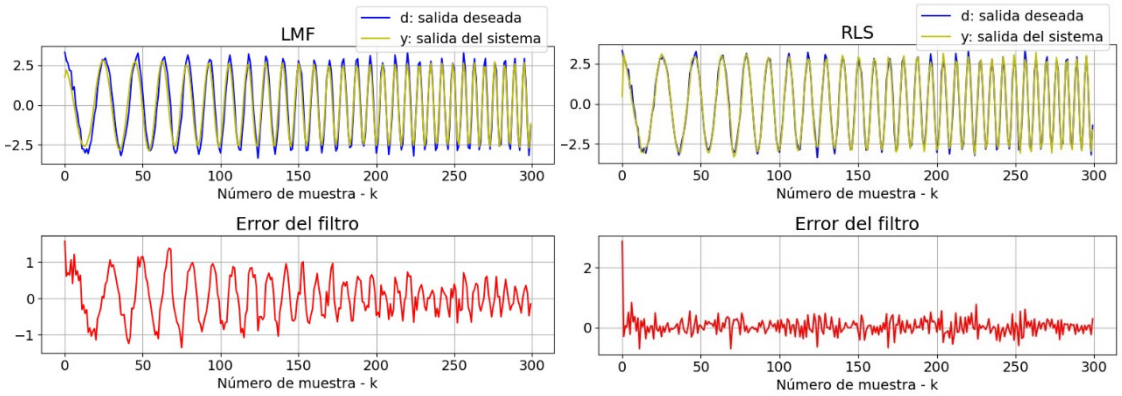


Fig. 181 – Adaptación del filtro: respuestas temporales y errores para función chirp. Izq.) LMF, derecha) RLS

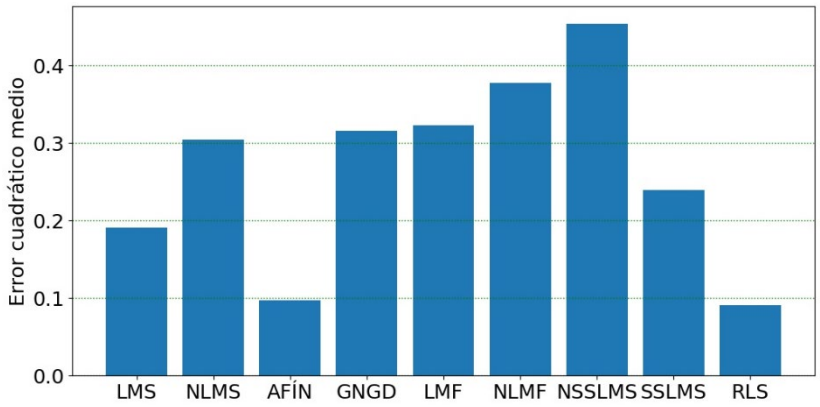


Fig. 182 – Error cuadrático medio para diferentes algoritmos con señal chirp

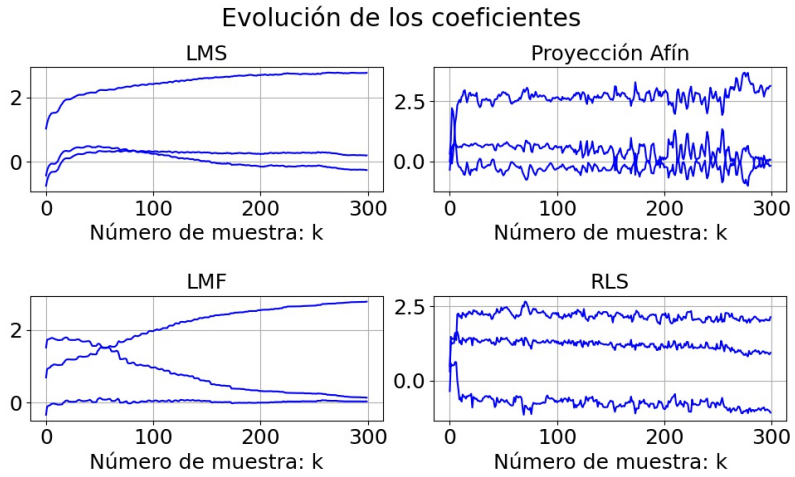


Fig. 183 – Evolución de los coeficientes para 4 algoritmos con señal chirp.

Ejercicio 4.12

Comparación 9 de Algoritmos adaptativos para identificación de sistemas con señal senoidal

Repetir el programa anterior para una señal senoidal de frecuencia angular $w_0 = 3\frac{rad}{seg}$. Modificar los parámetros de los algoritmos para mejorar la respuesta.

Resultados

En la Fig. 184 se muestran las señales deseadas y las señales predecidas para el algoritmo LMS. Se observa una rápida convergencia. En la Fig. 185 se muestra el error cuadrático medio, se observa que con los parámetros utilizados los algoritmos de proyección afín y RLS presentan menos error. En la Fig. 186 se muestra la evolución de los coeficientes para 4 algoritmos

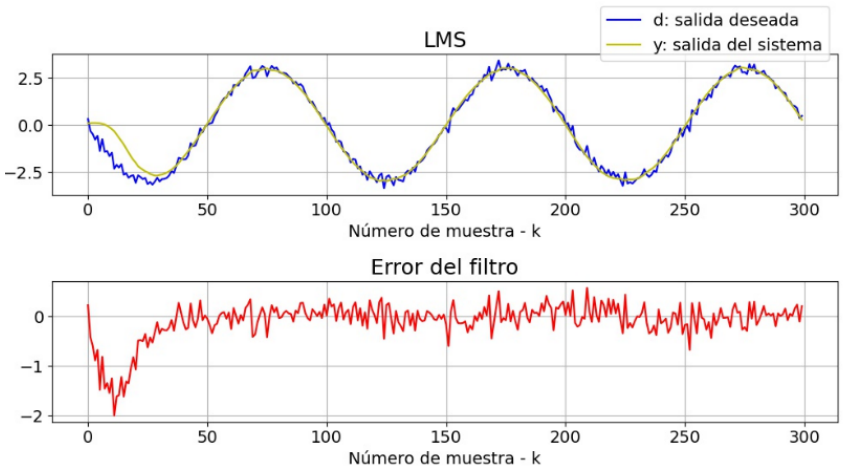


Fig. 184 – Adaptación del filtro: respuestas temporales y error para función seno y algoritmo LMS

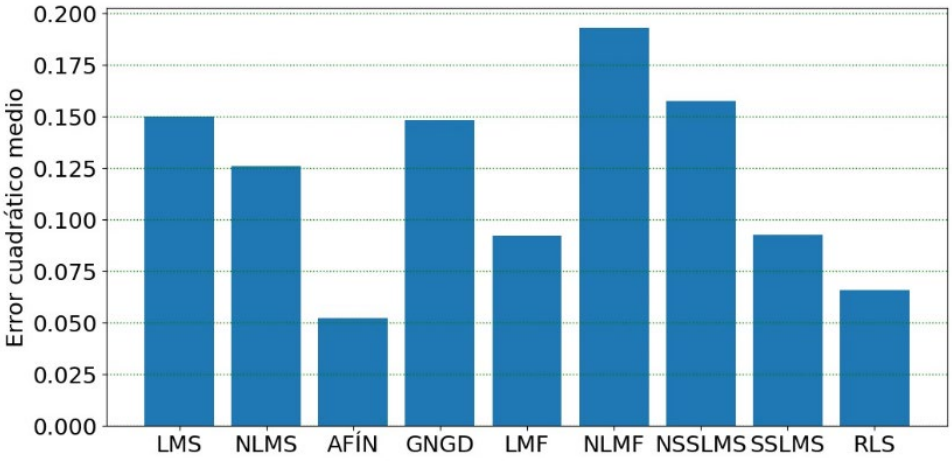


Fig. 185 – Error cuadrático medio para diferentes algoritmos con señal senoidal

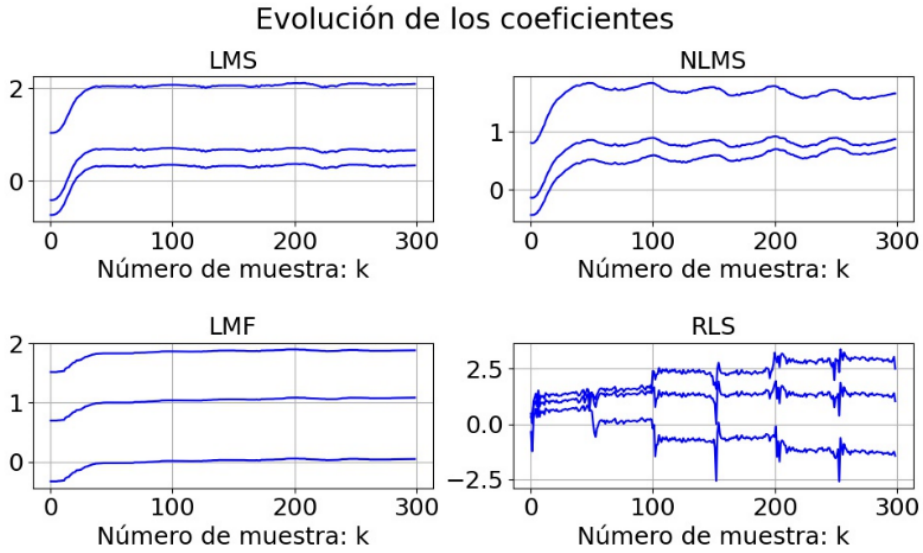


Fig. 186 – Evolución de los coeficientes para señal senoidal

Ejercicio 4.13

Comparación de 9 Algoritmos adaptativos para identificación de sistemas con Ruido Blanco Gaussiano

Repetir el programa anterior para el siguiente conjunto de señales.

x_1: ruido con distribución normal (gaussiana)

$s_1 = 3.x_1[n] - 3.x_1[n-1] + 0{,}9.x_1[n-2]$

$d = s_1 + v_1$

Resultados

En la Fig. 187 se muestran las señales deseadas y las señales predecidas para el algoritmo LMS. Se observa una rápida convergencia. En la Fig. 188 se observa el error cuadrático medio, con los parámetros utilizados los algoritmos de proyección NLMS, LMF y RLS presentan menos error.

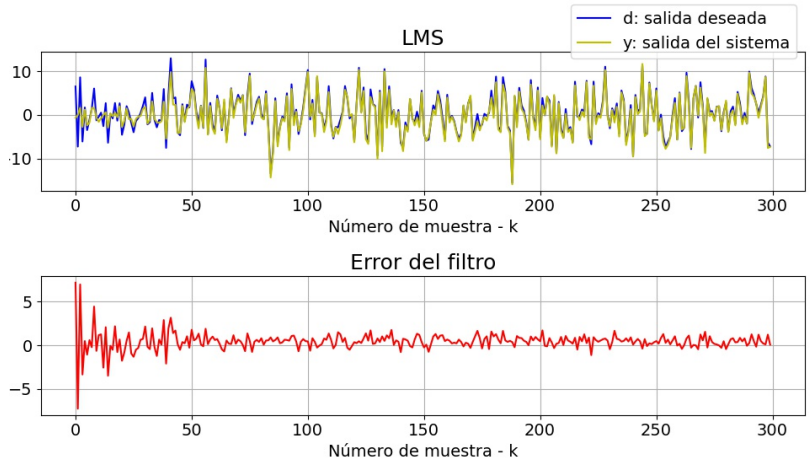


Fig. 187 – Adaptación del filtro: respuestas temporales y error para ruido blanco gaussiano y algoritmo LMS

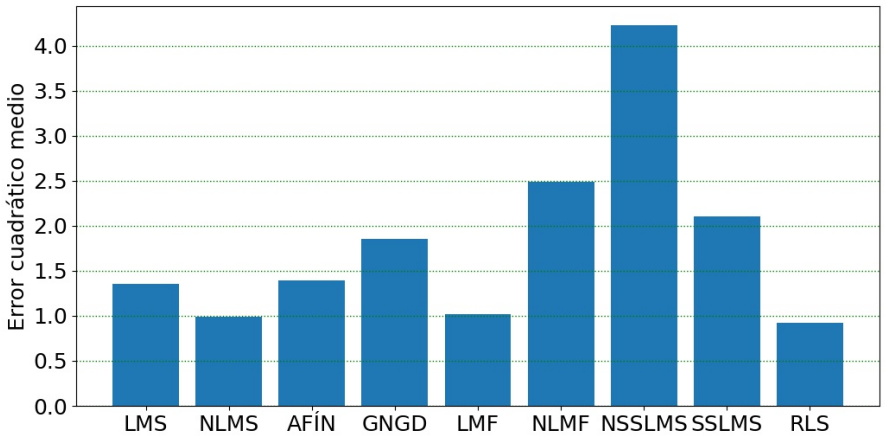


Fig. 188 – Error cuadrático medio para diferentes algoritmos con ruido blanco gaussiano

Ejercicio 4.14

Gráfico de la trayectoria de los coeficientes W de un sistema adaptativo LMS

Se pide modificar el Ejercicio 4.11 de manera de tener un filtro LMS con solo 2 coeficientes. Graficar la trayectoria para un valor muy bajo de μ y para otro valor muy alto, por ejemplo $\mu = 0{,}005$ y $\mu = 0{,}5$.

Se pide calcular el error cuadrático medio (ECM) para cada caso y escribir conclusiones

Los resultados se pueden observar en la Fig. 189 y Fig. 190

Para $\mu = 0{,}005$ se obtiene $ECM = 4{,}36$ en 300 iteraciones

Para $\mu = 0{,}5$ se obtiene $ECM = 0{,}157$ en 300 iteraciones

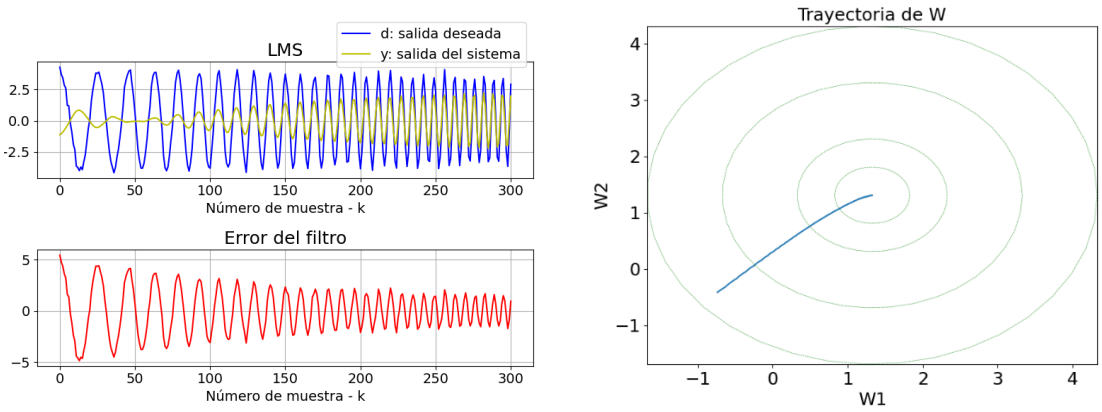


Fig. 189 – Izq.) Señales temporales y error, derecha) trayectoria de W para $\mu = 0{,}005$

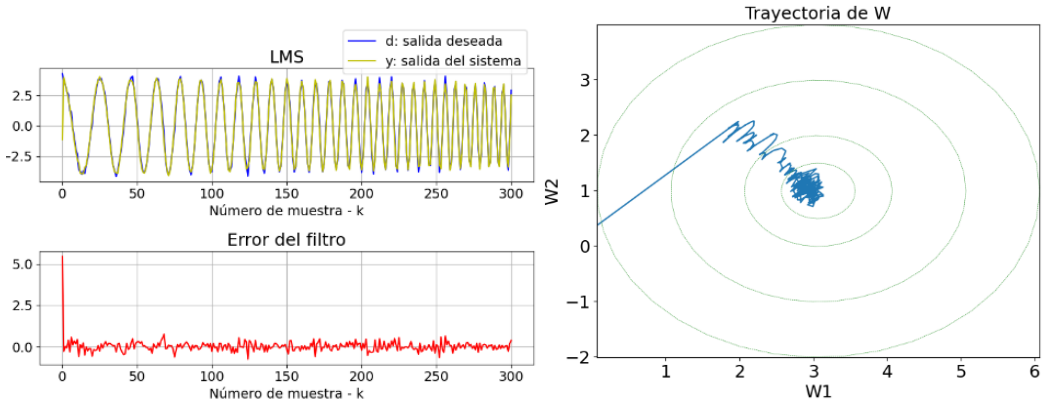


Fig. 190 – Izq.) Señales temporales y error, derecha) trayectoria de W para $\mu = 0{,}5$

Ejercicio 4.15

Ejemplo de Predicción de un filtro FIR con mediciones en tiempo real

Se desea predecir los coeficientes un sistema compuesto por un filtro tipo FIR mediante los algoritmos LMS y RLS

x_1: Entrada ruido Blanco Gaussiano

Salida del Filtro FIR:

$$d = 2{,}5.x_1[0] + 0{,}8.x_1[1] - 1.x_1[n-2] - 1.x_1[n-3] - 1{,}2.x_1[n-4]$$

Ayudas:

Utilizar la función padasip.filters.FilterLMS

filtro1 = pa.filters.FilterLMS(5, mu=0.6)

y = filtro1.predict(x1)

filtro2 = pa.filters.FilterRLS(5, mu=0.1, w="random")

Resultados obtenidos

Error cuadrático medio para LMS: 0.213

Error cuadrático medio para RLS: 0.323

En la Tabla VI se muestran los coeficientes reales y los predecidos, se observa una buena estimación para los algoritmos LMS y RLS. En la Fig. 191 y Fig. 192 se muestran las respuestas temporales y errores para los algoritmos LMS y RLS en la Identificación de un filtro FIR desconocido.

Tabla VI – Resultados de la estimación de coeficientes de un filtro FIR con LMS y RLS

Coeficientes reales del filtro FIR	2.5	0.8	-1	-1	1,2
Coeficientes predichos LMS	2.49999815	0.79999334	-1.00001044	-0.99998613	1.20000566
Coeficientes predichos RLS	2.50000154	0.79995975	-0.99991937	-0.99989452	1.19989362

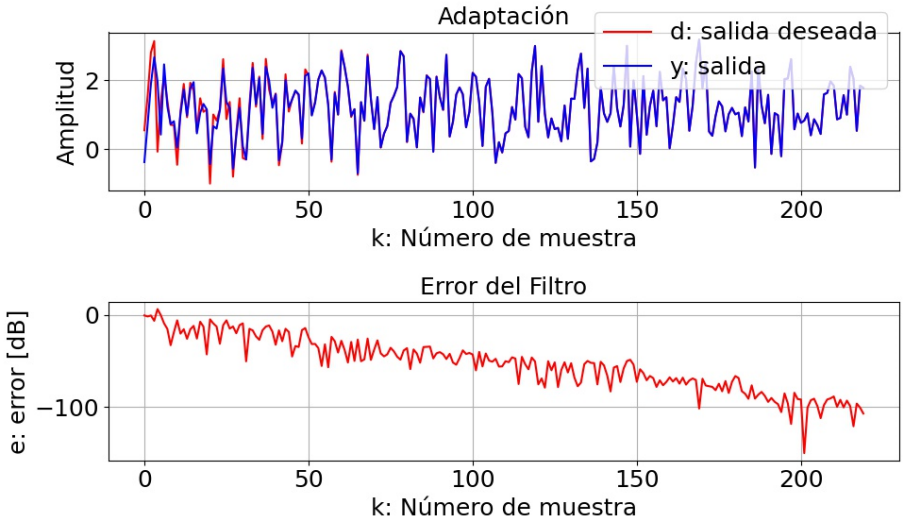


Fig. 191 – Identificación de un filtro FIR desconocido. Respuestas temporales y error para algoritmo LMS

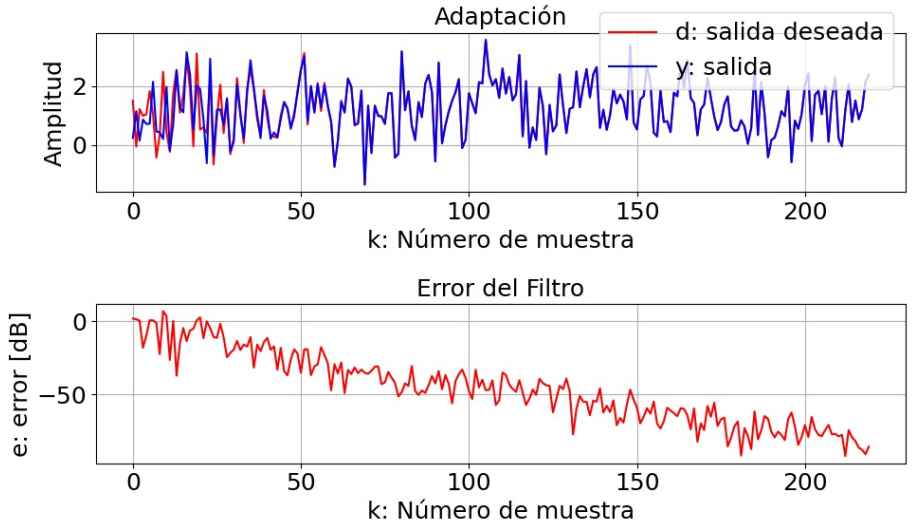


Fig. 192 – Identificación de un filtro FIR desconocido. Respuestas temporales y error para algoritmo RLS

Descarga de los códigos de los ejercicios

V – Breve Introducción a Redes Neuronales

Introducción a redes neuronales estáticas

Una red neuronal artificial, o red neuronal consiste en la interconexión de muchas unidades de procesamiento no lineales llamadas neuronas, obteniendo bloques no lineales distribuidos a través de toda la red. El desarrollo de redes neuronales desde el inicio está motivado por la forma en que trabaja el cerebro humano, y la interconexión de sus neuronas.

El bloque de construcción fundamental para las redes neuronales es la neurona de entrada única, como se muestra en la Fig. 193.

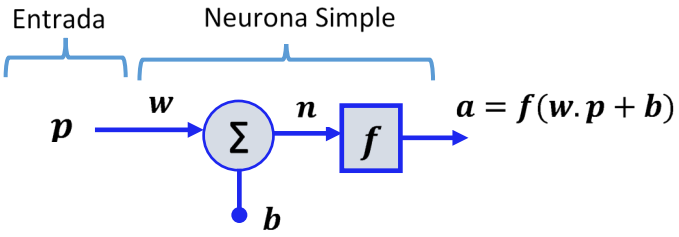


Fig. 193 – Esquema de una neurona simple

$\boldsymbol{p}$**:** entrada escalar

$\boldsymbol{w}$**:** peso escalar

$\boldsymbol{n}$**:** entrada neta, $\boldsymbol{n} = \boldsymbol{w}.\boldsymbol{p} + \boldsymbol{b}$

$\boldsymbol{f}$: Función de transferencia

Entradas y capas

Para describir redes que tienen múltiples capas, hay que utilizar una notación ampliada. Específicamente, se necesita hacer una distinción entre matrices de peso que están conectadas a entradas y matrices de peso que están conectadas entre capas. También se debe identificar el origen y el destino de las matrices de peso.

Llamaremos a las matrices de peso conectadas a las entradas ponderaciones o *pesos de entrada*; llamaremos *matrices de pesos* a las que tengan de entrada la salida de otra capa. Además, los superíndices se utilizan para identificar la fuente (segundo índice) y el destino (primer índice) para los diversos pesos y otros elementos de la red. En la Fig. 194 se muestra la red de entrada múltiple de una capa en forma abreviada. Siendo

R: cantidad de elementos en el vector de entrada

R: cantidad de neuronas en la capa 1

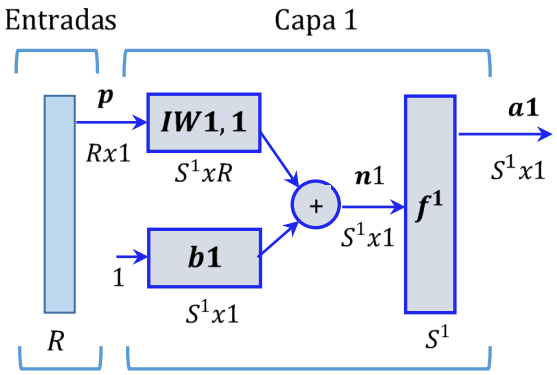


Fig. 194 – Red Neuronal de 1 capa de neuronas

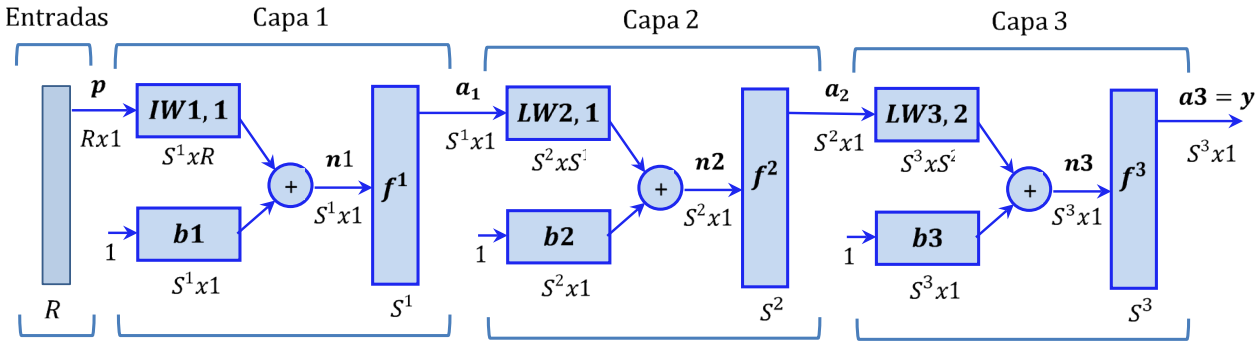
Como se puede ver, la matriz de ponderación o pesos conectada al vector de entrada p se etiqueta como una matriz de ponderación de entrada ($IW^{1,1}$) que tiene una fuente 1 (segundo índice) y un destino 1 (primer índice). Los elementos de la capa 1, como su sesgo, entrada neta y salida, tienen un superíndice 1 para indicar que están asociados con la primera capa.

Para redes con múltiples capas de neuronas se utiliza matrices de pesos de capa (LW), y para la capa de entrada se utiliza matrices de peso de entrada (IW). Es decir que cambia el nombre según sea capa de entrada o capa intermedia.

Red Neuronal Multicapa

Una red puede tener una o varias capas. En cada capa se tiene una matriz de pesos W, un vector de bias (sesgo escalar) b y un vector de salida a. Para distinguir entre las matrices de peso, los vectores de salida, etc., para cada una de estas capas se agrega el número de la capa mediante un superíndice en la variable de interés.

En la Fig. 195 se muestra una red de tres capas también se puede dibujar usando notación abreviada



$$a^1 = f^1.(I.W^{1,1}.p + b^1) \;\; ; \;\; a^2 = f^2(L.W^{2,1}.a^1 + b^2) \; ; \;\; a^3 = f^3(L.W^{3,2}.a^2 + b^3)$$

$$a^3 = f^3.(L.W^{3,2}.f^2(L.W^{2,1}.f^1(I.W^{1,1}.p + b^1) + b^2) + b^3)$$

$$a^3 = y$$

Fig. 195 – Red Neuronal Multicapa, se utiliza notación abreviada

Las redes de múltiples capas son bastante potentes. Por ejemplo, mediante una red de dos capas, con función sigmoidea en la primera capa y lineal en la segunda capa, puede entrenarse para aproximarse cualquier función (con un número finito de discontinuidades). Este red de dos capas se usa ampliamente en Redes neuronales multicapa y entrenamiento de propagación hacia atrás.

Aquí se supone que la salida de la tercera capa, a3, es la salida de red de interés, y esta salida se etiqueta como y. Esta notación se utiliza para especificar la salida de redes multicapa.

Introducción a Redes neuronales dinámicas

Las redes neuronales se pueden clasificar en estáticas y dinámicas. En las redes neuronales estáticas la salida se puede calcular directamente a partir de la entrada a través de conexiones feedforward. En redes dinámicas, la salida depende no solo de la entrada actual a la red, también de las entradas, salidas o estados actuales o anteriores de la red. Por ejemplo, las redes con filtros adaptativos son redes dinámicas, ya que la salida se calcula a partir de una línea de retardo con derivaciones de entradas anteriores. Las redes neuronales Hopfield también son redes dinámicas. Tiene conexiones recurrentes (de retroalimentación), lo que significa que la salida actual es función de las salidas correspondientes a tiempos anteriores.

El entrenamiento se basa en algoritmos de optimización que utilizan gradientes (como en el descenso más pronunciado y algoritmos de gradiente conjugado) o Jacobianos (como en los algoritmos de Gauss-Newton y Levenberg-Marquardt). La diferencia en el entrenamiento de redes estáticas y dinámicas está en la forma en que se calcula el gradiente o Jacobiano.

Las redes dinámicas contienen retardos y operan con una secuencia de entradas. Esto quiere decir, que el orden de las entradas es relevante para el correcto funcionamiento de la red. Estas redes dinámicas pueden tener conexiones puramente feedforward, como los filtros adaptativos, o también pueden tener algunas conexiones de retroalimentación (recurrentes), como la red Hopfield.

Las redes dinámicas tienen memoria. Su respuesta en un momento dado dependerá no solo de la entrada actual, sino del historial de la secuencia de entrada.

En la Fig. 196 se muestra se muestra el esquema de una red neuronal recurrente, a la derecha se muestra la red detallada con su evolución temporal.

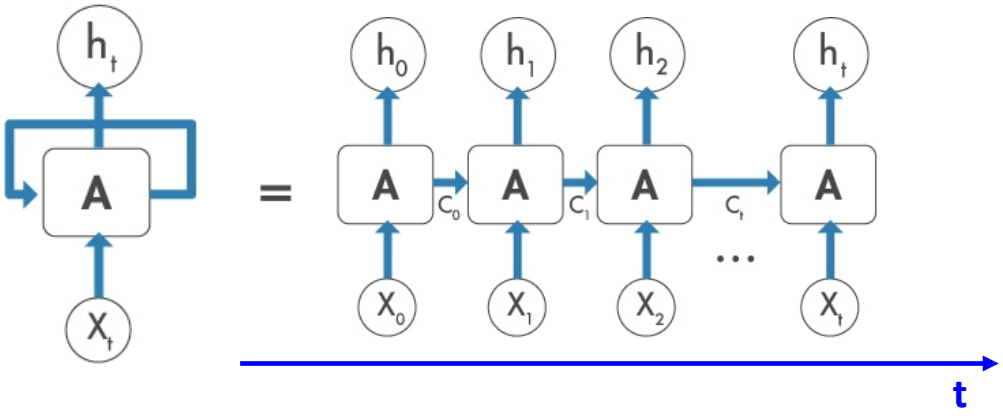


Fig. 196 – Red Neuronal Recurrente (RNN)

Breve Introducción a Redes Neuronales Convolucionales (CNN)

En 1988 Yann LeCun introduce las CNN en la literatura al mostrar la red convolucional LeNet, esto generó nuevas formas de trabajo y nuevas investigaciones en la visión por computadora.

Las redes neuronales convolucionales son redes artificiales con aprendizaje supervisado. Trabajan imitando a la corteza visual del cerebro humano. Tiene varias capas ocultas jerarquizadas y especializadas. Las primeras capas se encargan de detectar curvas y líneas, posteriormente las capas más profundas reconocen formas más avanzadas como rostros, especies de animales, objetos, etc. Como trabajan con matrices bidimensionales resultan efectivas para detección y clasificación de imágenes.

Se utilizan las siguientes capas:

- Convolución: utiliza filtros convolucionales con núcleos que detectan ciertas propiedades de la imagen
- ReLU o Unidad lineal rectificada o función de rectificación: generan salidas siempre positivas, mediante esta capa el entrenamiento resulta más rápido y eficiente
- Submuestreo o Pooling: reduce el tamaño de la imagen bajando la tasa de muestreo, de esta forma baja la carga computacional

Luego de estas capas se utiliza una red de clasificación.

En la Fig. 197 mostramos un ejemplo de convolución sucesiva de 3x3. La Fig. 198 muestra un ejemplo de CNN, las primeras capas se encargan del aprendizaje de características (en inglés feature learning).

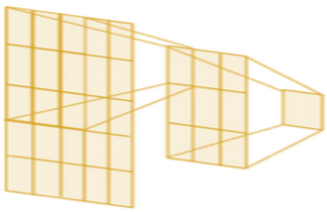


Fig. 197 – Convolución sucesiva de 3x3

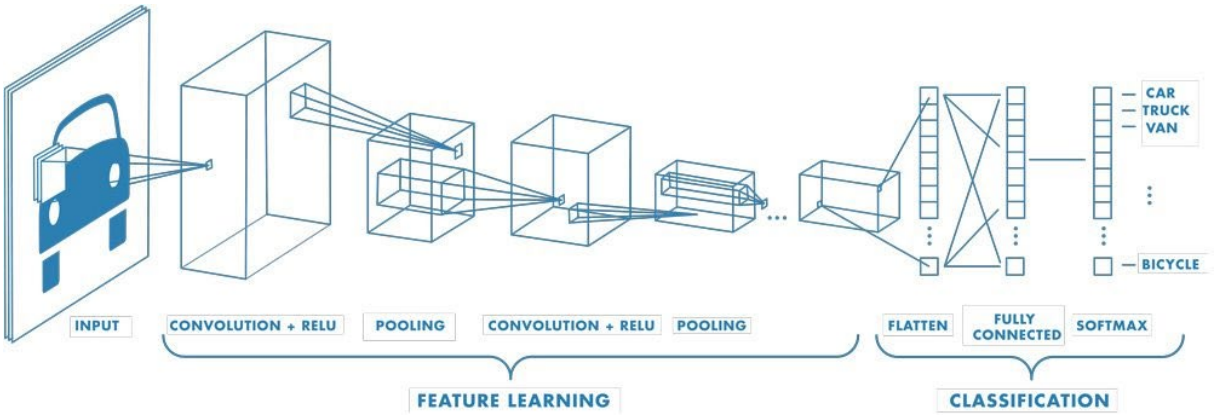


Fig. 198 – Red Neuronal Convolucional (CNN)

Fuente de la imagen: https://la.mathworks.com/discovery/convolutional-neural-network-matlab.html

En el Aprendizaje Automático (en inglés Machine Learning) las redes neuronales necesitan una etapa previa de extracción de características. En cambio, las CNN de Aprendizaje Profundo (en inglés Deep Learning) la extracción de características la realiza la red, ver Fig. 199.

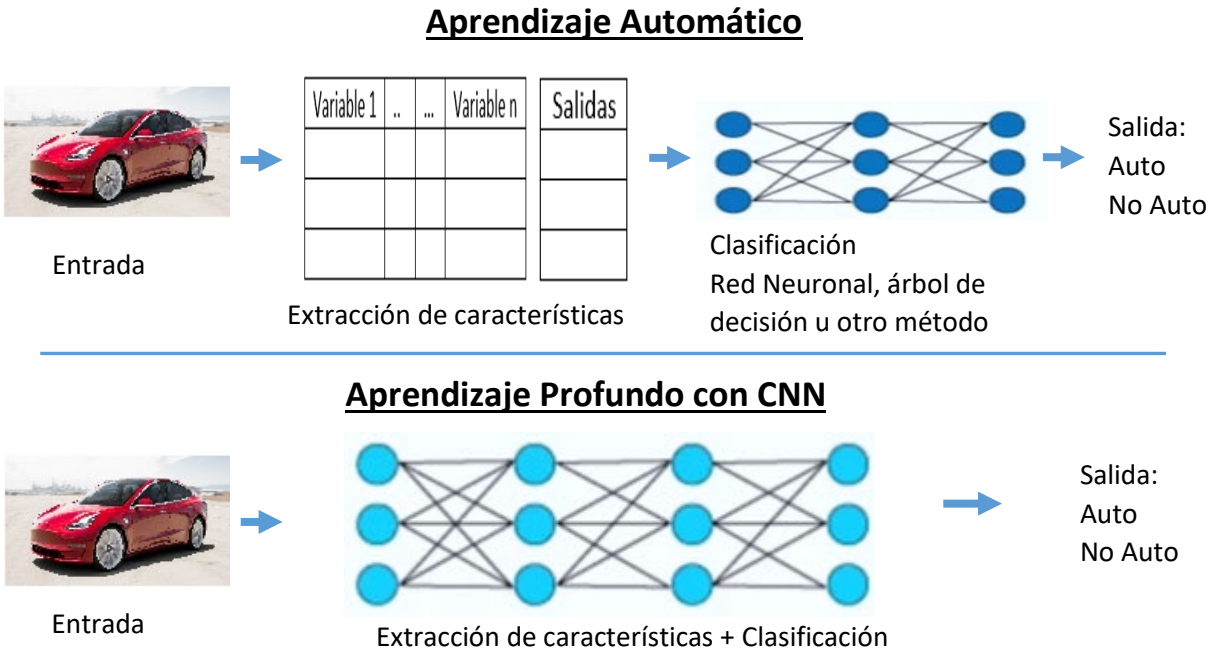


Fig. 199 – Diferencias entre aprendizaje Automático y aprendizaje profundo

Software, herramientas y librerías para redes neuronales

En el entorno Python utilizamos las siguientes librerías de software:

- Scikit-learn, Machine Learning in Python
  https://scikit-learn.org/stable/
- Pytorch
  https://pytorch.org/
- Tensorflow
  https://www.tensorflow.org/

En matlab® utilizamos la herramienta “nnstart” y la herramienta de redes neuronales

https://la.mathworks.com/

Las CNN necesitan mucha cantidad de imágenes para ser entrenadas satisfactoriamente. Por este motivo se utilizan bases de datos públicas, por ejemplo, ImageNet que en 2009 llego a 3.2 millones de imágenes. Para procesar grandes volúmenes de datos se necesita mucha capacidad computacional, en 2012 AlexNet es la primer red CNN que se entrena con NVIDIA GeForce 256 GPU (en inglés Graphics Processing Unit). Aparecen nuevas redes preconfiguradas y empiezan a competir entre ellas, a continuación, mencionamos algunas:

- Alexnet
- Googlenet/Inception
- VGG-19
- ResNet

Se dispone en forma pública distintos conjuntos de entrenamiento, se enumeran los más utilizados:

- ImageNet ahora contiene más de 14 millones clasificadas con 20000 etiquetas
- CIFAR10 contiene 60000 imágenes, cada imagen tiene un tamaño 32x32x3 en RGB. Posee 10 clases:
- avión, automóvil, ave, gato, venado, perro, rana, caballo, barco y camión.
- MNIST contiene imágenes de dígitos del 0 al 9, con 60000 imágenes de entrenamiento y 10000 de prueba

Ejercicios en Matlab

Ejercicio 5.1

Ejemplo de red neuronal Adaline simple

Se pide analizar el funcionamiento del siguiente código

```
%% Mi primer Red Neuronal
% ADALINE (linearlayer) simple
% Armamos una Red con Dos entradas y una salida.
% También se puede utilizar train
% Valor predeterminando de pesos y bias: 0
% Mostramos los valores actuales:
net = linearlayer
net1 = configure (net, [0; 0], [0]);
% Asignamos valores  a los pesos y al bias:
net1.IW { 1, 1 } = [2  3] ;
net1.b {1} = -4;
% Mostramos los valores de pesos y bias:
W = net1.IW {1,1}
b = net1.b {1}
% Mostramos la red
view(net1)
% Simulamos datos de entrada p
p = [5; 6];
a = sim (net1, p)
% Graficamos "2 Regiones":
x=-10:0.01:10 ;
y=4/3 -2/3*x ;
plot(x,y, 'linewidth',3);  grid on
```

Resultados

```
W =     2     3
b =    -4
a =    24
```

En la Fig. 200 se muestra la topología de la red. En la Fig. 201 se muestran las 2 regiones correspondientes a cada clase. Se observa la recta que corresponde al límite de decisión

Analizamos los resultados de la red ADALINE simple

$\boldsymbol{a = w_{11}.p_1 + w_{12}.p_2 + b}$

$\rightarrow \; w_{11}.p_1 + w_{12}.p_2 + b = 0$

Reemplazamos: $p_1 = x \;\; ; p_2 = y \;\;;$

$w_{11} = 2\,;\; w_{12} = 3 \quad ; \;\; ; \; b = -4$

$\rightarrow \; 2.x + 3y - 4 = 0 \quad \rightarrow \; \boldsymbol{y = 4/3 - 2/3.x}$

Graficamos:

x=-10:0.01:10 ; y=4/3 -2/3*x ; plot(x,y, 'linewidth',3); grid on

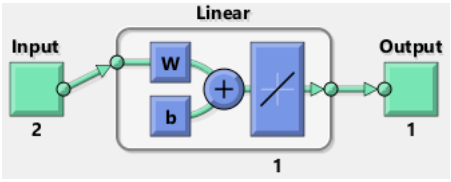


Fig. 200 – Topología de la red Adaline

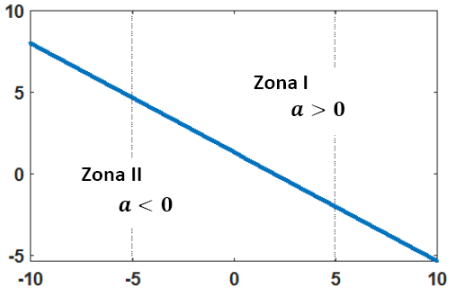


Fig. 201 – Límite de decisión de una neurona simple ADALINE

Ejercicio 5.2

Ejemplo de redes neuronales para ajuste de datos, reconocimiento de patrones, agrupamiento y series temporales

Mediante la herramienta "Neural Network Start" de Matlab®, con el comando nnstart, se muestra la interfaz gráfica de usuario (GUI) para redes neuronales, ver Fig. 202. Dispone de botones hacia las otras herramientas

Se pide analizar el funcionamiento para:

- Ajuste de datos
- Reconocimiento de Patrones
- Agrupamiento
- Series dinámicas temporales

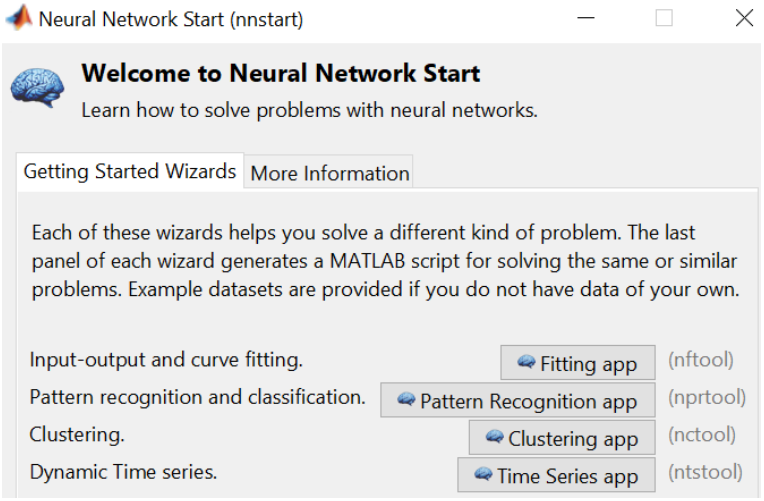


Fig. 202 – Herramienta "Neural Network Start" de Matlab®

Solicite el código al autor

Ejercicio 5.3

Ejemplo de Ajuste de datos con Redes Neuronales

Dado el programa del ejercicio, se pueden cargar datos propios o utilizar datos de ejemplo de Matlab®. Se utiliza 70 % de entrenamiento, 15 % de validación y 15 % de testeo. Se configura con 10 neuronas correspondiente a la capa oculta y algoritmo Levenberg-Marquardt

Se pide

Analizar el siguiente programa de ajuste de datos

Modificar la cantidad de neuronas de la red y analizar el cambio de Performance de la red.

```
%% Ajuste de datos entrada-salida mediante red neuronal
% x entradas, t salidas deseadas
 clear all ; clc; close all
 [x,t] = simplefit_dataset;
 % x = simplefitInputs; t = simplefitTargets;
% Elegimos algoritmo de entrenamiento, para ayudas tipear help nntrain
% trainlm: algoritmo de propagación hacia atrás, Levenberg Marquardt, generalmente rápido.
% trainbr: algoritmo de regulación Bayesiana. Más lento, pero puede dar mejores resultados
% en problemas desafiantes.
% trainscg: Algoritmo de propagación hacia atrás con gradiente conjugado. Utiliza menos memoria.
 trainFcn = 'trainlm';  % Levenberg-Marquardt backpropagation.
% Creamos la red neuronal para el ajuste de datos
 neuronas_capa_oculta = 10;
 net1 = fitnet(neuronas_capa_oculta,  trainFcn);
```

```
% Elegimos funciones de pre y post procesamiento de las entradas y salidas
% Para mostrar la lista de todas las funciones tipear: help nnprocess
 net1.input.processFcns = {'removeconstantrows','mapminmax'};
 net1.output.processFcns = {'removeconstantrows','mapminmax'};
% Configuración de datos de entrenamiento, validación y testeo
% Para ayudas de funciones tipear: help nndivision
 net1.divideFcn = 'dividerand';  % Divide los datos en forma aleatoria
 net1.divideMode = 'sample';  % Divide cada muestra
 net1.divideParam.trainRatio  = 72/100 ; % Porcentaje para el entrenamiento
 net1.divideParam.valRatio   = 14/100 ; % Porcentaje para validación
 net1.divideParam.testRatio  = 14/100 ; % Porcentaje para testeo
% Seleccionamos función de Performance
% Para ayudas tipear:  help nnperformance
 net1.performFcn =  'mse'  ;  % Error cuadrático medio
% Elegimos las funciones a graficar
% Para ayudas tipear:  help nnplot
 net1.plotFcns = { 'plotperform' , 'plottrainstate' , 'ploterrhist' , 'plotregression' , 'plotfit' } ;
% Entrenamos la red
 [net1,tr] = train(net1,x,t);
% Testeamos la red
 y1 = net1(x);
 e = gsubtract(t, y1);
 performance = perform(net1,t,y1)
% Recalculamos Performance de Entrenamiento, Validación y testeo
 train_Target = t .* tr.trainMask{1};
 val_Target = t .* tr.valMask{1};
 test_Target = t .* tr.testMask{1};
 train_Performance = perform(net1,train_Target,y1)
 val_Performance = perform(net1,val_Target,y1)
 test_Performance = perform(net1,test_Target,y1)
% Mostramos la red neuronal
 view(net1)
% Graficamos
% Agregar comentarios según los resultados que se quieran graficar
   figure; plotperform(tr) ;
```

```
figure; plottrainstate(tr) ;
figure; ploterrhist(e) ;
figure; plotregression(t,y1) ;
figure; plotfit(net1,x,t) ;
% Desarrollo, cambiar (false) o (true) para habilitar los distintos códigos
if (true)   % Generamos función de red neuronal
   genFunction(net1,'NeuralNetworkFuncion1');
   y = NeuralNetworkFuncion1(x);
end
if (true)
   % Generamos función de red neuronal solo soporta matrices de entrada (no array de celdas)
   genFunction(net1,'NeuralNetworkFuncion1','MatrixOnly','yes');
   y = NeuralNetworkFuncion1(x);
end
if (true)   % Generamos diagrama Simulink
   gensim(net1) ;
end
```

En la Fig. 203 se muestra la topología de la red, donde la entrada y la salida son analógicas. En la Fig. 204 se observa el correcto ajuste de datos con errores bajos y los coeficientes de regresión R cercanos a 1.

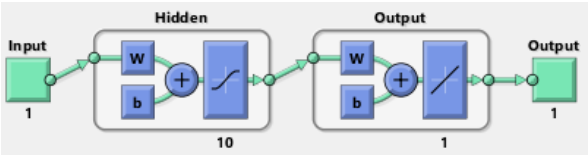


Fig. 203 –Topología de la red para el Ajuste de datos

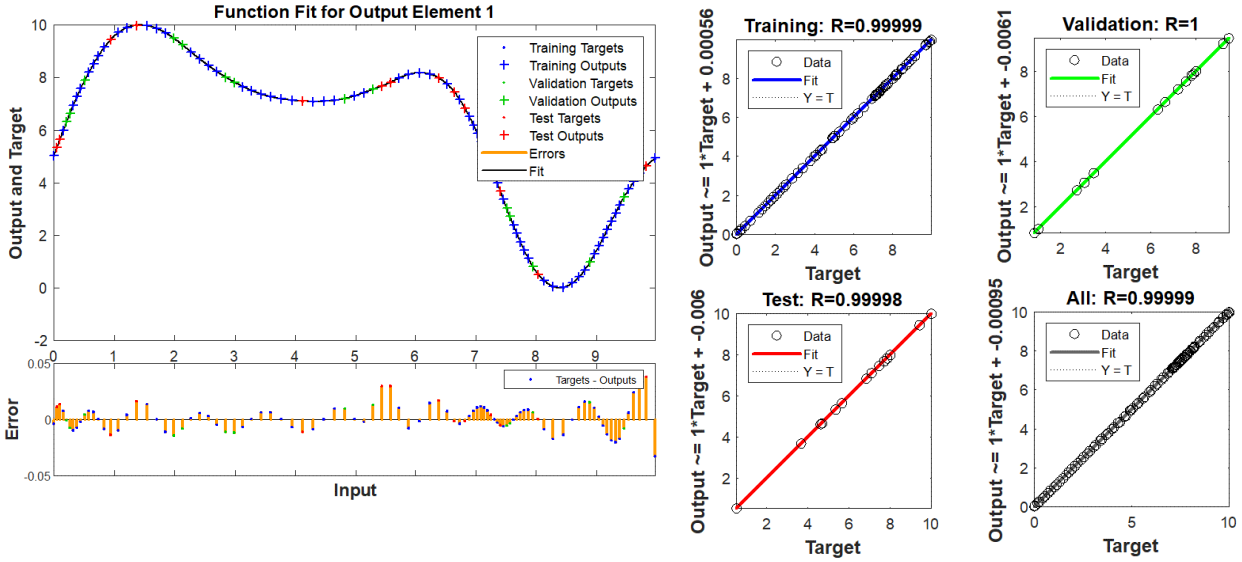


Fig. 204 – Resultados del Ajuste de datos: curvas de datos, coeficiente de correlación R

Ejercicio 5.4

Reconocimiento de Patrones: Clasificación con redes neuronales

Se desea analizar la respuesta de una red neuronal de clasificación de datos modificando su topología y sus algoritmos.

Se pide:

a) Abrir la herramienta de Clasificación de Matlab® (nprtool), al cargar datos, seleccionar “simpleclass_dataset”. También se pueden cargar datos propios o utilizar datos de ejemplo de Matlab®. Posteriormente seleccionar 70 % de entrenamiento, 15 % de validación y 15 % de testeo. Elegir 10 neuronas para la capa oculta. Pulsar el botón correspondiente para “entrenar” la red. Presionar botones “Plot confusion” y “Performance” (en nntraintool). En la Fig. 205 se muestran los resultados, se observa que en todos los casos las 4 categorías se clasificaron correctamente (100% correcto).

b) En la pantalla final de la herramienta generar el script (programa) correspondiente, usar el botón “Advanced Script”.

c) Modificar la cantidad de neuronas de la red y los algoritmos de propagación de errores y analizar el cambio de Performance de la red.

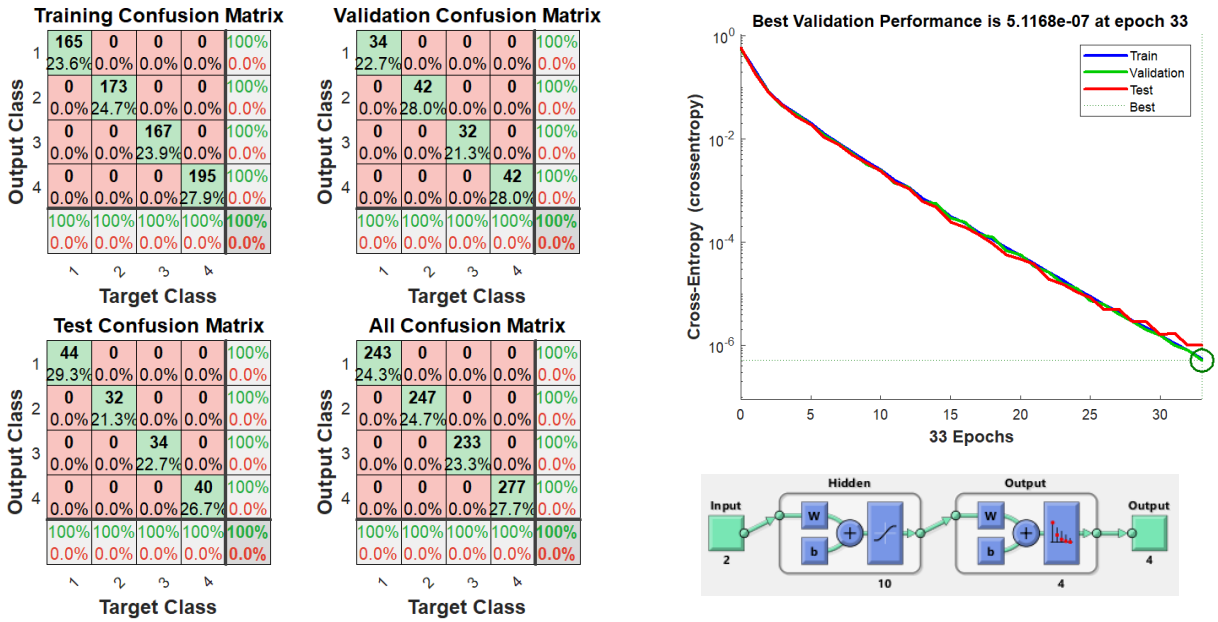


Fig. 205 – Resultados de la Clasificación de datos en 4 categorías. Izquierda) Matriz de Confusión, Derecha) Performance y topología de la red

Ejercicio 5.5

Ejemplo de red neuronal no lineal auto regresiva (NAR) para predecir serie temporal.

La predicción de una secuencia de valores en una serie temporal también se conoce como predicción de varios pasos. Las redes de circuito cerrado pueden realizar predicciones de varios pasos. Cuando falta la retroalimentación externa, las redes de circuito cerrado pueden seguir prediciendo mediante la retroalimentación interna. En la predicción NAR, los valores futuros de una serie de tiempo se predicen solo a partir de valores anteriores (pasados) de esa serie.

Se pide analizar el siguiente código y verificar su funcionamiento

```
%% Entrenamos una red NAR y predecimos datos nuevos
% Importamos datos simples y creamos la red
T = simplenar_dataset;
net1 = narnet(1:2, 10);
% Preparamos los datos con preparets, entrenamos la red y luego la mostramos
[ Xs , Xi , Ai , Ts ] = preparets( net1 , {}, {} , T) ;
net1 = train(net1,Xs,Ts,Xi,Ai);
view(net1)
% Calculamos la performance
[Y,Xf,Af] = net1(Xs,Xi,Ai);
perf = perform(net1,Ts,Y)
```

```
% Para predecir la salida los siguientes 20 pasos temporales, primero simulamos la red en
% un lazo cerrado. En este modo, la red solo tiene una entrada conectada con la salida
[netc1, Xic, Aic] =  closeloop(net1,Xf,Af); % lazo cerrado
view(netc1)
% Para simular la red 20 pasos de tiempo, la entrada es un array de largo 20
% La red necesita las condiciones iniciales |Xic| y |Aic|.
y2 = netc1( cell(0, 20), Xic, Aic)
plot(cell2mat(T), 'linewidth',3); hold on
y = cell2mat(y2) ;  n= (length(T)-length(y))+1 :length(T) ;
plot(n,y, 'linewidth',3); grid on
legend('Datos reales', 'Datos predecidos')
```

En la Fig. 206 se muestran los resultados del programa.

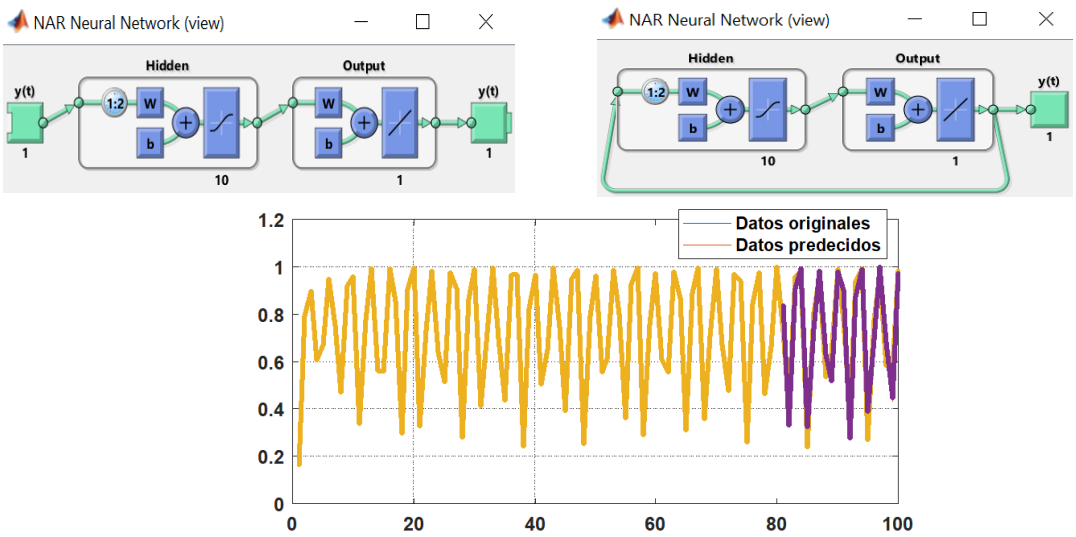


Fig. 206 – Resultados de Red Neuronal dinámica NAR para predecir serie temporal. Izquierda arriba) red a lazo abierto. Arriba derecha) red a lazo cerrado. Abajo) Serie temporal original y datos predecidos

Ejercicio 5.6

Ejemplo simple de Red Neuronal estática

Se desea aproximar la siguiente función mediante la red neuronal de la Fig. 207.

$$g(p) = 0{,}5 + cos\left(\frac{\pi}{5}.p\right) + ruido \;\; para \;\; 0 \leq p \leq 10$$

La red debe ser entrenada con muchos valores de p, utilizando el diagrama de la Fig. 208.

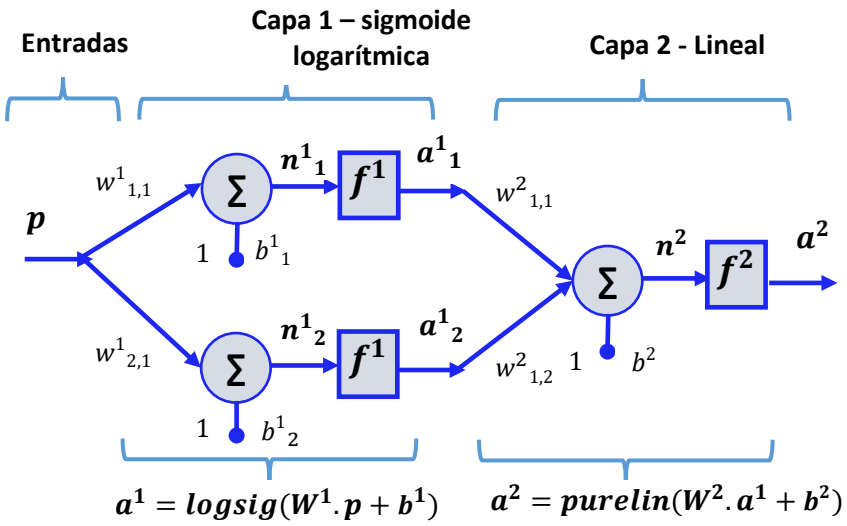


Fig. 207 – Topología de la red neuronal 1-2-1 para aproximación de función

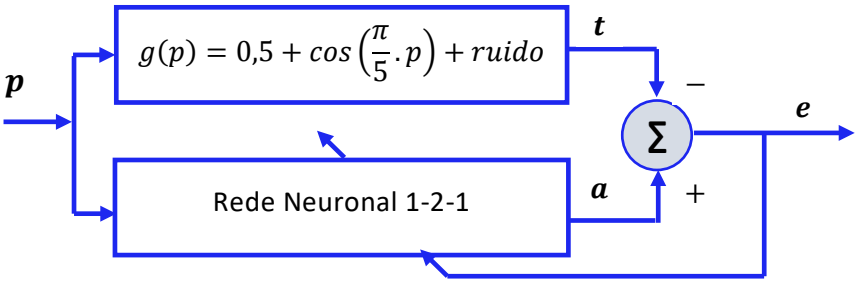


Fig. 208 – Diagrama de adaptación de la red neuronal 1-2-1

Resolución

```
%% Conjunto de datos
 x=0: 0.02:10 ;
 t_set = 0.5 +cos((pi/5)*x);
 t= t_set;
 t= t +0.2*randn(1,length(t)) ;  % Agregamos ruido
%% Creamos la red neuronal feedforward
 net1 = feedforwardnet(2);
%% Elegimos las funciones a graficar
% Para ayudas tipear:  help nnplot
 net1.plotFcns =  { 'plotperform' , 'plottrainstate', 'ploterrhist', 'plotregression', 'plotfit' } ;
%% Entrenamos la red
[net1 , tr]  =  train(net1, x, t);
```

```
view(net1) % Mostramos la red
%% Testeamos la red
 y1 = net1(x);
 e1 = gsubtract(t,y1);
 performance = perform(net1, t, y1)
%% Graficamos, Agregar comentarios para no mostrar todos los resultados
   figure; plotperform(tr) ; figure; plottrainstate(tr);
   figure; ploterrhist(e1) ;  figure; plotregression(t,y1);
   figure; plotfit(net1, x, t);
 %% nnet.guis.closeAllViews()
```

En la Fig. 209 se muestran los resultados obtenidos. Para el conjunto de entrenamiento, se obtiene un alto coeficiente de correlación, $R = 0{,}964$

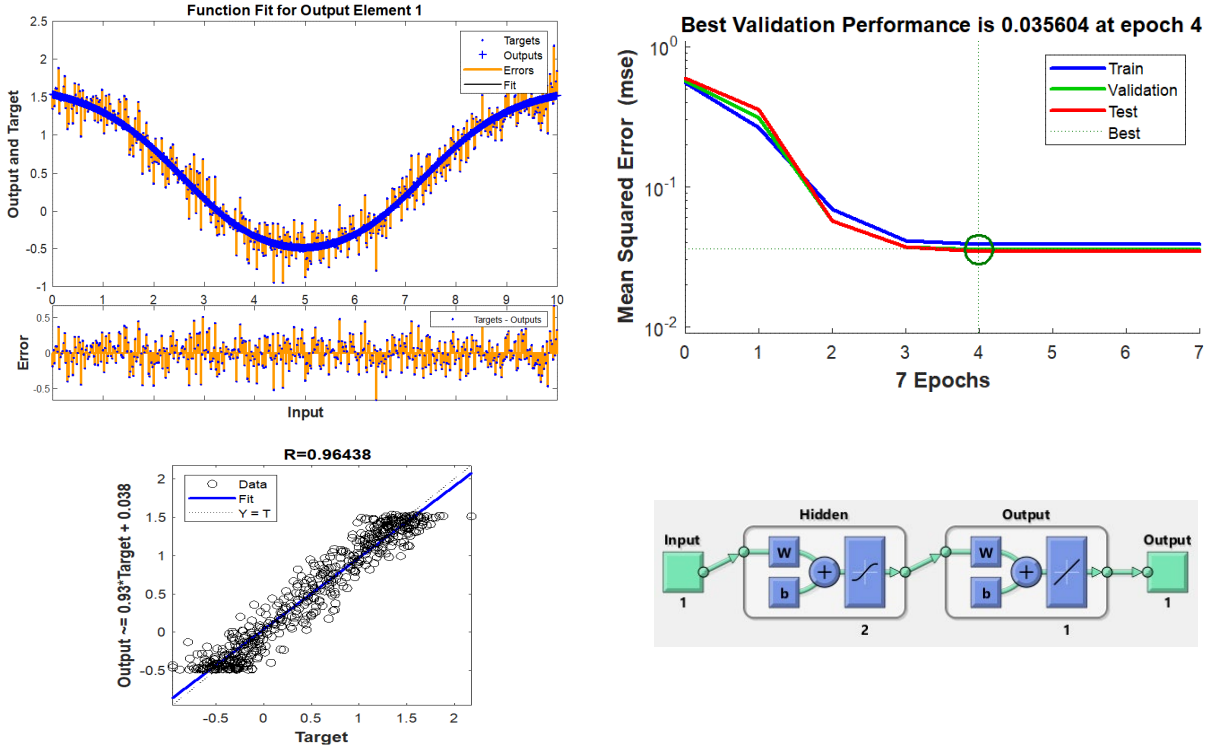


Fig. 209 – Resultados del ajuste de datos de la función senoidal con ruido

Ejercicio 5.7

Ejemplo de Red Neuronal estática para ajuste de datos

A continuación, se muestra un programa de ajuste datos mediante redes neuronales en entorno Matlab®, del tipo alimentación hacia adelante (feedforward).

```
%% Ajuste de datos entrada-salida mediante red neuronal
%% Importamos o armamos el conjunto de datos
 %load mis_datos1 ; x= inputs ;  t=targets ;
 % x entradas, t salidas deseadas
 x=0;
 for i=1:99
  dx = abs(0.01*randn()) ;
  x=[x x(end)+dx] ;
 end
 % t corresponde a las salidas deseadas
 t= sin(20*x) +cos(8*x) +1; % t= sin(20*x) +10*x;
 t= t +0.2*randn(1,length(t)) ;  % Agregamos ruido
 figure; plot(x,t, 'linewidth',2); axis tight; grid on
%% Creamos la red neuronal de ajuste de datos: fitnet
 % para ayudas tipear help nntrain
 % 'trainlm' algoritmo de propagación hacia atrás Levenberg-Marquardt, generalmente rápido.
 % 'trainbr' algoritmo de regulación Bayesiana. Lento, mejores resultados en problemas desafiantes.
 % 'trainscg' Algoritmo de propagación hacia atrás con gradiente conjugado. Utiliza menos memoria.
 trainFcn = 'trainlm';  % Levenberg-Marquardt backpropagation.
 hiddenLayerSize = 8;
 net1 = fitnet(hiddenLayerSize,trainFcn);
%% Elegimos funciones de pre y post procesamiento de las entradas y salidas
% Para mostrar la lista de todas las funciones tipear: help nnprocess
 net1.input.processFcns = {'removeconstantrows','mapminmax'};
 net1.output.processFcns = {'removeconstantrows','mapminmax'};
%% Configuración de datos de entrenamiento, validación y testeo
% Para ayudas de funciones tipear: help nndivision
 net1.divideFcn = 'dividerand';  % Divide los datos en forma aleatoria
 net1.divideMode = 'sample';  % Divide cada muestra
 net1.divideParam.trainRatio =   80/100;
```

```
 net1.divideParam.valRatio   =   10/100;
 net1.divideParam.testRatio  =   10/100;
%% Seleccionamos función de Performance. Para ayudas tipear: help nnperformance
 net1.performFcn = 'mse';  % Error cuadrático medio
%% Elegimos las funciones a graficar
% Para ayudas tipear: help nnplot
 net1.plotFcns = { 'plotperform' , 'plottrainstate' , 'ploterrhist' , 'plotregression' , 'plotfit'} ;
%% Entrenamos la red
 [net1,tr] = train(net1,x,t);
%% Testeamos la red
 y1 = net1(x);
 e1 = gsubtract(t,y1);
 performance = perform(net1,t,y1)
%% Recalculamos Performance de Entrenamiento, Validación y testeo
 train_Targets = t .* tr.trainMask{1};
 val_Targets = t .* tr.valMask{1};
 test_Targets = t .* tr.testMask{1};
 train_Performance = perform(net1,train_Targets,y1)
 val_Performance = perform(net1,val_Targets,y1)
 test_Performance = perform(net1,test_Targets,y1)
%% Mostramos la red neuronal
 view(net1)
%% Graficamos
% Agregar comentarios según los resultados que se quieran graficar
   figure; plotperform(tr) ; figure; plottrainstate(tr) ;
   figure; ploterrhist(e1) ;  figure; plotregression(t,y1) ;
   figure; plotfit(net1,x,t);
%% Desarrollo, generamos script (opcional)
 % Generamos función de red neuronal
      genFunction(net1,'NeuralNetworkFunction1') ;
      y = NeuralNetworkFunction1(x);
 % Generamos función de red neuronal solo soporta matrices de entrada (no array de celdas)
      genFunction(net1,'NeuralNetworkFunction1', 'MatrixOnly', 'yes');
      y = NeuralNetworkFunction1(x);
 % Generamos diagrama Simulink
```

```
% gensim(net1);
```

En la Fig. 210 se muestra el resultado del ajuste de datos con redes neuronales, también se observa la topología de la red utilizada. Para el conjunto de entrenamiento, se obtiene un alto coeficiente de correlación, $R = 0{,}991$

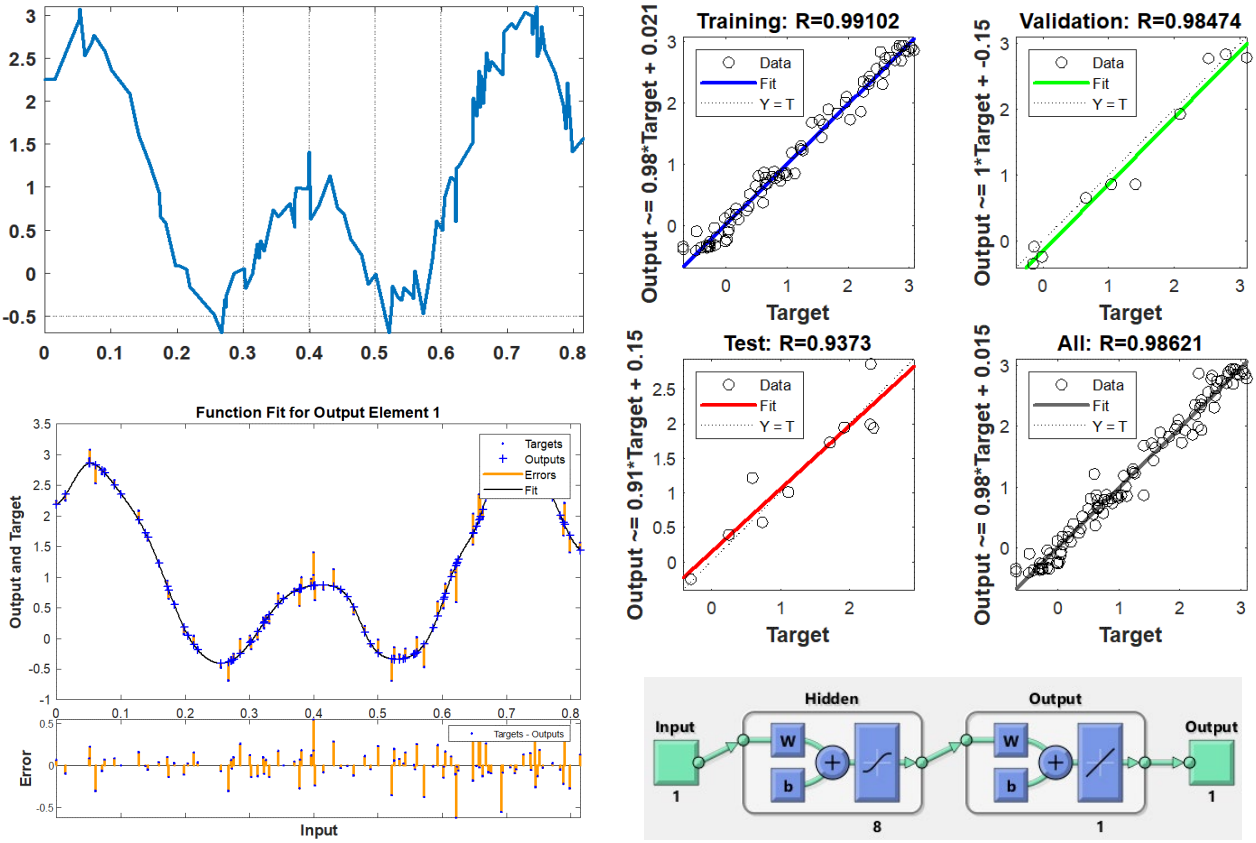


Fig. 210 – Resultados del Ajuste de datos con redes neuronales estáticas

Ejercicio 5.8

Ejemplo de Red Neuronal Recurrente (RNN)

Las redes neuronales recurrentes son similares a las redes de del tipo alimentación hacia adelante (feedforward), excepto que cada capa tiene una conexión recurrente con un retraso de derivación asociado. Esto permite que la red tenga una respuesta dinámica infinita para datos de entrada correspondientes a series temporales. Esta red es similar a las redes neuronales de retardo de tiempo y retardo distribuido. En las redes neuronales recurrentes cada capa tiene una conexión recurrente con un retraso de derivación asociado. Las RNN son similares a las redes neuronales de retardo de tiempo y de retardo distribuido. En la Fig. 211 se muestra la topología de la red utilizada.

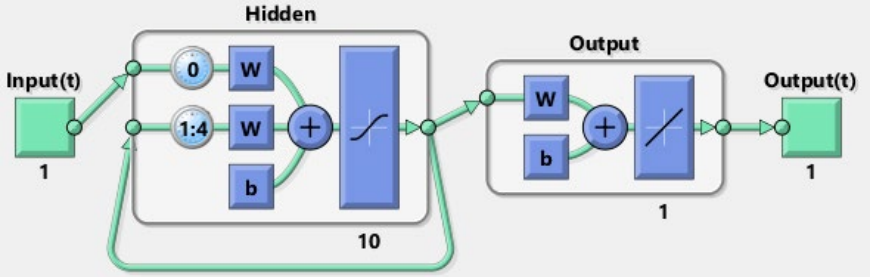


Fig. 211 – Topología de red neuronal recurrente con 10 neuronas ocultas y 4

A continuación, se muestra un programa de ajuste datos de una serie temporal mediante redes neuronales recurrentes (RNN) en entorno Matlab®. Se pide analizar el funcionamiento del programa.

```
%% Importamos o armamos el conjunto de datos
 %load mis_datos2 ;
 % x entradas, t salidas deseadas
 x=0;
 for i=1:99
   dx = abs(0.01*randn()) ;
   x=[x x(end)+dx] ;
 end
 % t corresponde a las salidas deseadas
 t= sin(20*x) +cos(8*x) +1;
 t= t +0.2*randn(1,length(t)) ; % Agregamos ruido
 figure; plot(x,t, 'linewidth',2); axis tight; grid on
 X=num2cell(x) ;   T=num2cell(t) ;
%% Creamos una red recurrente con 4 retardos y 10 neuronas en la capa oculta
net1 = layrecnet(1:4, 10);
% función preparets, acomoda los datos de series temporales de entrada y destino
% para simular o entrenar la red
[Xs,Xi,Ai,Ts] = preparets(net1, X, T); % acomodamos los datos
[net1, tr1] = train(net1, Xs, Ts, Xi, Ai) ; % Entrenamos la red
view(net1) % Mostramos la red
Y = net1(Xs,Xi,Ai);
perf1 = perform(net1 ,Y, Ts)
%% Graficamos
 figure; plotperform(tr1) ;
```

En la Fig. 212 se muestran los resultados del programa. Se obtienen los gráficos con ayuda de la herramienta nntraintool de Matlab®. Para el conjunto de entrenamiento, se obtiene un alto coeficiente de correlación, $R = 0{,}9984$

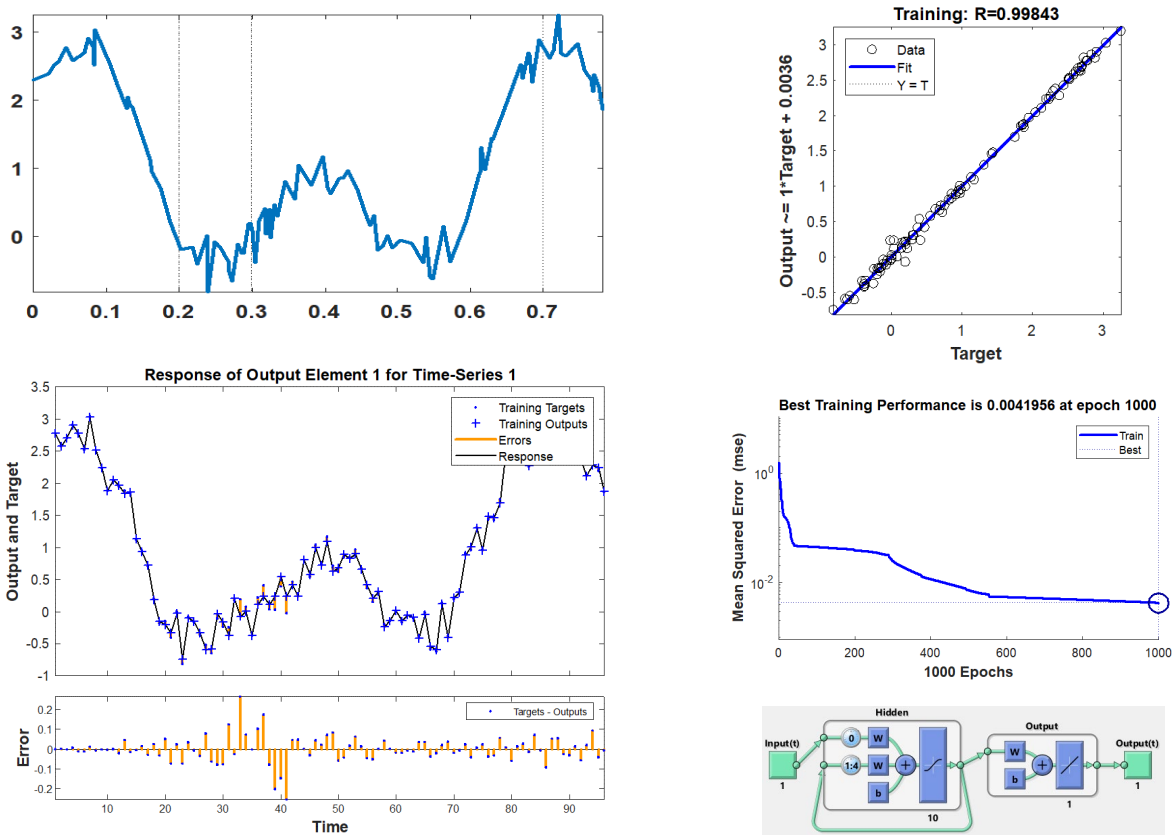


Fig. 212 – Resultados del Ajuste de datos con redes neuronales recurrentes

Ejercicio 5.9

Ejemplo de red neuronal GoogleNet para identificación de imágenes.

Se pide analizar y probar el funcionamiento del siguiente programa

```
%% Clasificación de Imágenes con Red Neuronal GoogLeNet y Matlab®
% Clasificamos una imagen utilizando la red neuronal convolucional profunda previamente
% entrenada con GoogLeNet. Esta red ha sido entrenada en más de un millón de distintas
% imágenes. Puede clasificar imágenes en 1000 categorías de objetos (como teclado, taza
% de café, lápiz y muchos animales).

% Cargamos la Red entrenada previamente con comando googlenet
  net1 = googlenet ;
% Mostramos el tamaño de la imagen.
```

```
 inputSize1 = net1.Layers(1).InputSize
% ClassNames son los nombres de las clases. Se muestran 10 clases en forma aleatoria
% Total de clases: 1000
 classNames = net1.Layers(end).ClassNames;
 numClasses = numel(classNames);
 disp(classNames(randperm(numClasses,12)))
% Leemos la imagen para Clasificar y cambiamos su tamaño.
 I = imread('peppers.png');
 figure ;  imshow(I)
 size(I)
% Tamaño de  imagen 384-by-512 pixeles y 3 colores (RGB). Cambiamos su tamaño con imresize
 I = imresize(I, inputSize1(1:2));
 figure ; imshow(I)
% Clasificamos la imagen
 [label1,scores1] = classify(net1,I);
 label1
% Mostramos la imagen con la etiqueta y la probabilidad de predicción.
 figure  ; imshow(I)
 display(string(label1) + ", " + num2str(100*scores1(classNames == label1),3) + "%");
% Mostramos los 5 resultados con mayor probabilidad
%  La red clasifico la imagen como bell pepper con alta probabilidad.
 [~,idx] = sort(scores1, 'descend');
 idx = idx(5:-1:1);
 class_Names_Top = net1.Layers(end).ClassNames(idx);
 scores_Top = scores1(idx);
% Graficamos
 figure
 barh(scores_Top)
 xlim([0 1])
 title('5 Predicciones con mayor probabilidad')
 xlabel('Probabilidad')
 yticklabels(class_Names_Top)
```

Resultados

Se muestran algunas clases:

Ponche de huevo, vaca, castillo, bolsa de dormir, apósito adhesivo, cinturón de seguridad, naranja, pimientos morrones (bell pepper)

En la Fig. 213 y Fig. 214 se muestran los resultados obtenidos

Fig. 213 – Imagen de pimientos morrones

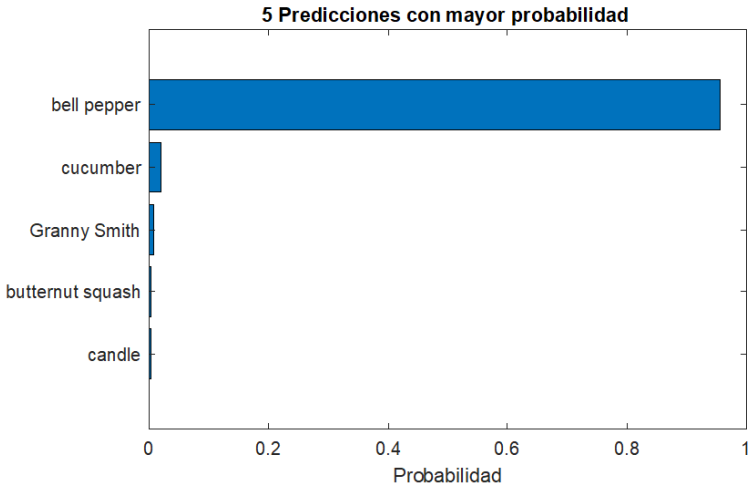


Fig. 214 – Resultados de la predicción

Ejercicio 5.10

Herramienta para diseño de CNN

A partir de la versión 2018b de Matlab®, se incorporó una herramienta para diseño de redes de aprendizaje profundo. En Matlab® se debe escribir el comando *deepNetworkDesigner* para abrir la herramienta, ver Fig. 215

Se pide diseñar una red con esta herramienta y probar su funcionamiento

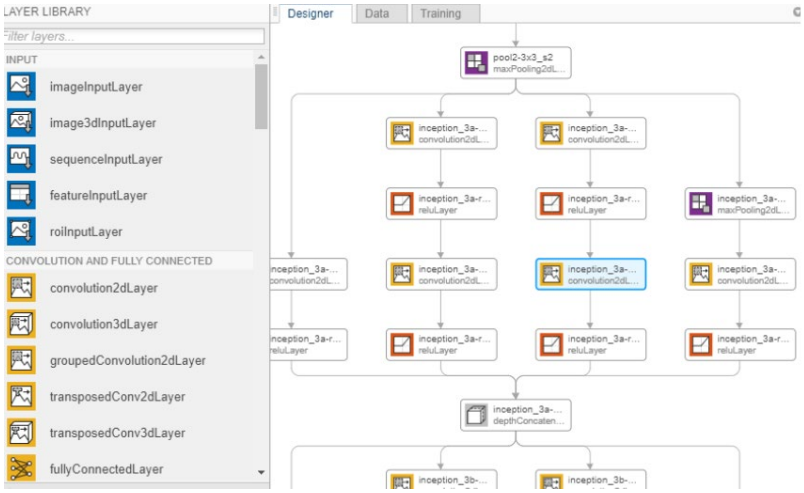


Fig. 215 – Herramienta de Matlab® para diseño de redes de aprendizaje profundo

Ejercicio 5.11

Diseño de CNN (con script)

Mediante el programa de abajo se detectan imágenes con números (ver Fig. 216) mediante una red convolucional con la siguiente arquitectura

```
layers = [
   imageInputLayer([28 28 1]) % primer capa: entradas
   convolution2dLayer(3,8,'Padding',1) ; % capa convolucional
   batchNormalizationLayer;  reluLayer
   maxPooling2dLayer(2,'Stride',2) ;
   convolution2dLayer(3,16,'Padding',1); % capa convolucional
   batchNormalizationLayer;  reluLayer ;
   maxPooling2dLayer(2,'Stride',2) ;
   convolution2dLayer(3,32,'Padding',1) ; % capa convolucional
    ……..  Ver programa];
```

Se obtiene una precisión mayor al 99%

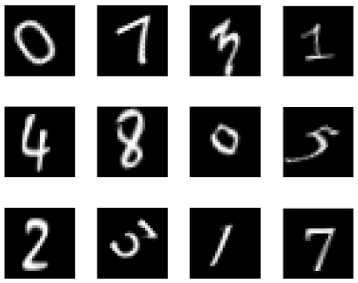


Fig. 216 – Subconjunto de datos

Se pide analizar el funcionamiento del programa, luego reducir la estructura de la red y sacar conclusiones

```
% Diseño de Red CNN para clasificación de números
% Importamos los datos
Path_imagenes1 = fullfile(matlabroot,'toolbox','nnet','nndemos','nndatasets','DigitDataset');
conjunto_imagenes = imageDatastore(Path_imagenes1,  IncludeSubfolders', true, ...
                          'LabelSource', 'foldernames');
% Graficamos
figure;
data1 = randperm(10000,12);
for i = 1:12
   subplot(3,4,i);
   imshow(conjunto_imagenes.Files{data1(i)});
end
cuentaEtiquetas = countEachLabel(conjunto_imagenes)
img1 = readimage(conjunto_imagenes,1);
size(img1)
numTrainFiles = 700;
[conjunto_imagenesEntrenamiento,conjunto_imagenesValidacion] =
splitEachLabel(conjunto_imagenes,...
  numTrainFiles,'randomize');
% Definimos arquitectura de red convolucional
layers = [
   imageInputLayer([28 28 1]) % entradas
   convolution2dLayer(3,8,'Padding',1) ; % capa convolucional
   batchNormalizationLayer;  reluLayer ; % batch y relu
   maxPooling2dLayer(2,'Stride',2) ; % Pooling
   convolution2dLayer(3,16,'Padding',1); % capa convolucional
   batchNormalizationLayer;  reluLayer ; % batch y relu
   maxPooling2dLayer(2,'Stride',2) ; % Pooling
   convolution2dLayer(3,32,'Padding',1) ; % capa convolucional
   batchNormalizationLayer; reluLayer ; % batch y relu
   fullyConnectedLayer(10) ;
   softmaxLayer ;
   classificationLayer ; ];
% Opciones
options = trainingOptions('sgdm', ...
   'MaxEpochs',5, 'ValidationData',conjunto_imagenesValidacion, ...
```

```
    'ValidationFrequency',40,  'Verbose',false, 'Plots','training-progress');
% Entrenamos la red
net = trainNetwork(conjunto_imagenesEntrenamiento,layers,options);
% Clasificamos y probamos la red
SalidaPrededida = classify(net,conjunto_imagenesValidacion);
SalidaValidacion = conjunto_imagenesValidacion.Labels;
precision = sum(SalidaPrededida == SalidaValidacion)/numel(SalidaValidacion)
```

Ejercicio 5.12

Clasificación imágenes webcam con red googlenet

Se pide analizar el siguiente código

```
% CNN Red Alexnet
% Se pueden modificar las etiquetas
mi_camara = webcam; % Nos conectamos a la cámara
net = alexnet; % Cargamos la red neuronal alexnet
for i=1:15
    im1 =  snapshot(mi_camara); % Capturamos una imagen
    image(im1); % Mostramos
    im1 = imresize(im1,[227 227]);% Cambiamos el tamaño para alexnet
    etiqueta1 = classify(net,im1); % Clasificamos
    title(char(etiqueta1)); % Mostramos etiqueta
    etiqueta1
    drawnow
    pause(0.5) % pausa de 0,5 seg.
end
clear mi_camara
```

Ejercicios en Python

Ejercicio 5.13

Clasificación de datos con red neuronal Perceptron Multi Capa (MLP)

Se utilizan librerías scikit-learn de Python. Se pide probar y analizar el siguiente código. Modificar parámetros para mejorar el rendimiento de la red

```
from sklearn.datasets import make_blobs # librería sklearn
import matplotlib.pyplot as  plt # importamos
import numpy as  np #
# Definimos colores y marcadores para graficar
colores= ['r', 'g', 'b', 'y', 'y']
marcadores = ['o', (5,1), ',', 'v', '^', '<', '>', 's', 'd', '.', 'x', '+']
# Generamos conjunto de datos y graficamos
cant_muestras = 300
centros1 = ([-1.6, 1], [1, 4.2], [-1.4, 3.7],[1.5, 1.5],[0, 7.5])
datos1, labels = make_blobs(n_samples=cant_muestras,
                    centers=centros1,   cluster_std=0.7, #0.6
                    random_state=0)
colores = ('blue', 'red', 'green', 'orange', 'magenta')
plt.rc('font', size=16) ; plt.rc('axes', titlesize=18)
fig, ax1 = plt.subplots(figsize=(8,4))
for n_clases in range(len(centros1)):
   ax1.scatter(datos1[labels==n_clases][:, 0],
           datos1[labels==n_clases][:, 1],
           c=colores[n_clases],s=60, label=str(n_clases) ,
           marker=  marcadores[n_clases] )
ax1.legend()  ; ax1.grid()
# Separamos los datos en entrenamiento y prueba
from sklearn.model_selection import train_test_split
dataset1 = train_test_split(datos1, labels, test_size=0.15)
train_data, test_data, train_labels, test_labels = dataset1
############################################################
### Creamos la red neuronal MLP para clasificar
```

```
from sklearn.neural_network import MLPClassifier
alpha= 1e-5
clf1 = MLPClassifier(solver='lbfgs', # optimizador basado en método de Newton
                     alpha=1e-5, # Parámetro de penalización L2 (término de regularización).
                     hidden_layer_sizes =(5,), # 1 capa oculta con 5 neuronas, se puede probar otro valor
                     random_state=0) # Determina generación de números aleatorios para w y b
# Se puede crear la red con pocos parámetros, pero se generan redes grandes
 #clf1 = MLPClassifier(random_state=1, max_iter=500).fit(train_data, train_labels)
 #clf1.get_params(1)
# Entrenamos la red neuronal MLP
clf1.fit(train_data, train_labels)

################################################################
###  Probamos la red neuronal
# Valor medio de precisión
prec =clf1.score(train_data, train_labels)
print("Valor medio de Precisión para datos de prueba: ", prec)
# Clasificamos los datos y calculamos precisión
from sklearn.metrics import accuracy_score
predict_test = clf1.predict(test_data)
predict_train =  clf1.predict(train_data)
train_precision =  accuracy_score(predict_train, train_labels)
print("Precisión para datos de entrenamiento: ", train_precision)
test_precision = accuracy_score(predict_test, test_labels)
print("Precisión para datos de prueba: ", test_precision)
# Mostramos primeros 10 resultados predecidos
a=predict_train[:10]
print(a)
# Matriz de confusión
from sklearn.metrics import confusion_matrix
conf_matr_train = confusion_matrix(predict_train, train_labels)
conf_matr_test = confusion_matrix(predict_test, test_labels)
print(conf_matr_train)
print(conf_matr_test)
from sklearn.metrics import classification_report
```

```
print(classification_report(predict_test, test_labels))
print(classification_report(predict_train, train_labels))

#################################################################
###   Generamos gráfico 2D con diferentes regiones con distintos parámetros
vector_alpha =[1e-5, 1e-2 ,0.1 ,1]
# Generamos grilla
x_min, x_max = datos1[:, 0].min() - 0.45 , datos1[:, 0].max() + .45
y_min, y_max = datos1[:, 1].min() - 0.45 , datos1[:, 1].max() + .45
h = .02 # paso de la grilla
xx, yy  = np.meshgrid(np.arange(x_min, x_max, h),
                np.arange(y_min, y_max, h)) # grilla
for alpha in vector_alpha:
   clf1 = MLPClassifier( solver = 'lbfgs', ## este optimizador se basa en el método de Newton
         alpha=alpha, # # Parámetro correspondiente a la penalización L2 (o término de regularización).
         hidden_layer_sizes=(5,), # capas ocultas, probar 6 u otro valor
         max_iter=2000,
         random_state=0,) # Determina generación de números aleatorios para w y b
   # Entrenamos la red neuronal MLP
   clf1.fit(train_data, train_labels)
   # Clasificamos la grilla y graficamos zonas
   Z = clf1.predict(np.c_[xx.ravel(), yy.ravel()])  # predecimos
   Z = Z.reshape(xx.shape)
   fig, ax2  =  plt.subplots(figsize=(8,4))
   levels = [0,  1,  2,  3,  4]
   cs = ax2.contourf(xx, yy, Z, levels,
                colors=('r', 'g', 'b', 'y', 'c'),
                origin='lower', extend='both', alpha=.5 )
   plt.colorbar(cs)
   # Graficamos los datos
   colores= ['r', 'g', 'b', 'y', 'y']
   marcadores = ['o', (5,1),  ',',  'v',  '^',  '<',  '>',  's',  'd',  '.',  'x',  '+']
   for n_clases in range(len(centros1)):
      ax2.scatter(datos1[labels==n_clases][:, 0],
            datos1[labels==n_clases][:, 1],
```

```
            c=colores[n_clases], s=60, label=str(n_clases) ,
            marker=  marcadores[n_clases] ) #
    ax2.set_xlim(xx.min(), xx.max())
    ax2.set_ylim(yy.min(), yy.max())
    ax2.set_xticks(()) , ax2.set_yticks(())
    ax2.set_title('Clasificación') ;ax2.grid()
```

Resultados

Valor medio de Precisión para datos de prueba: 0.9686

Precisión para datos de entrenamiento: 0.9686

Precisión para datos de prueba: 0.9777

Mostramos los 10 primeros resultados de la clasificación

[0 0 2 0 3 1 1 2 3 2]

En la Tabla VII mostramos matriz de confusión para los datos de entrenamiento. En la Tabla VIII mostramos matriz de confusión para los datos de prueba

Tabla VII – Matriz de Confusión del entrenamiento

Clase	0	1	2	3	4
0	51	0	1	0	0
1	0	48	2	1	0
2	2	0	47	0	0
3	0	2	0	48	0
4	0	0	0	0	53

Tabla VIII – Matriz de Confusión para los datos de prueba

Clase	0	1	2	3	4
0	51	0	1	0	0
1	7	0	0	0	0
2	0	10	0	1	0
3	0	0	0	10	0
4	0	0	0	0	7

En la Tabla IX se muestra el reporte de clasificación para los datos de prueba

Tabla IX – Reporte de Clasificación

Clase	Precisión	Sensibilidad	Medida F1	soporte
0	1	1	1	7
1	1	0,91	0,95	11
2	1	1	1	10
3	0,91	1	0,95	10
4	1	1	1	7

En la Fig. 217 se muestran los datos. En las Fig. 218 y Fig. 219 se muestran los resultados para distintos valores de parámetros alpha de la red neuronal

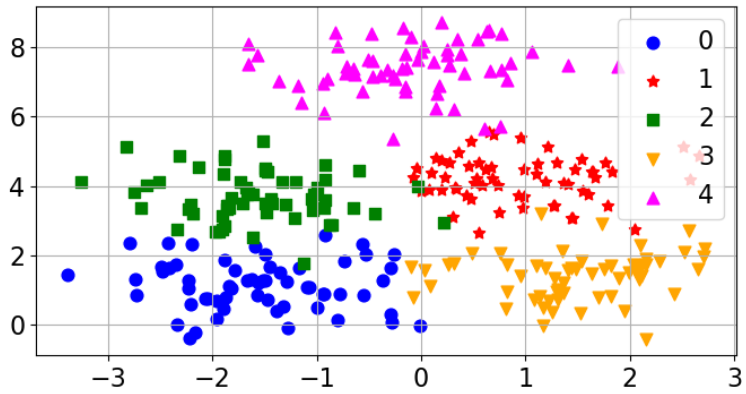


Fig. 217 –Gráfico de datos con 5 categorías

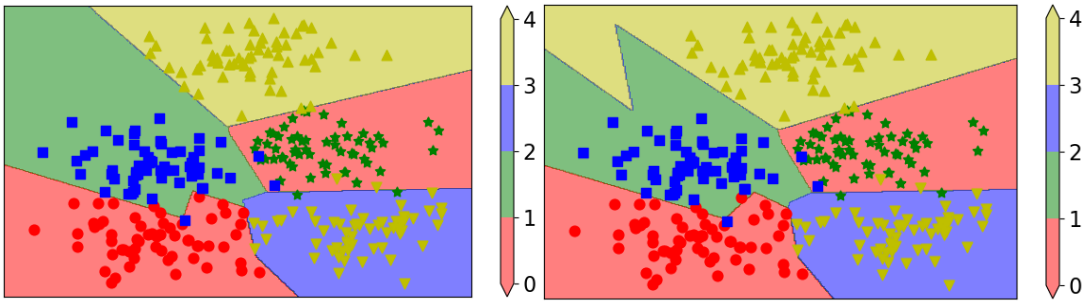


Fig. 218 – Clasificación de datos. Izq.) coeficiente alpha 1e-5, derecha) 1e-2

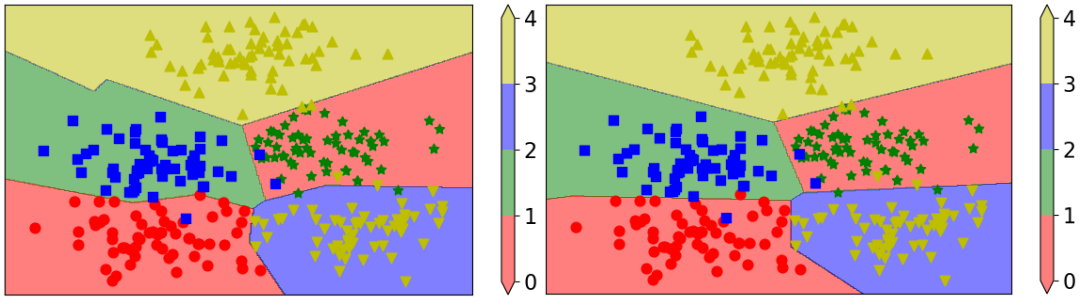


Fig. 219 – Clasificación de datos. Izq.) coeficiente alpha 0,1, derecha) 1

Ejercicio 5.14

Ajuste de datos con las librerías scikit-learn de Python

Se pide analizar y probar el siguiente ejemplo. Luego modificar parámetros para mejorar la respuesta de la red neuronal

```
from sklearn.model_selection import train_test_split # importamos
from sklearn.neural_network import MLPRegressor # importamos librerías red neuronal MLP
import numpy as np # importamos
from sklearn.datasets import make_regression # importamos
import matplotlib.pyplot as plt
X, y  =  make_regression(n_samples=200, random_state=1) # datos

# Generamos los datos
largo = 300
np.random.seed(seed=1)
mu, sigma = 0, 1 # valor medio y desviación estándar
# x1: Matriz de datos de tamaño: largo x 3.
# Contiene ruido con distribución normal (gaussiana)
X = np.random.normal(mu, sigma, (largo, 2))
# Ruido para agregar
v1 = np.random.normal(0.5, 0.5, largo)
# Señal original s y Señal deseada: d
y = 3*X[:,0] -2*X[:,1] #+ 0.9*X[:,2]
y =  y + 3*v1   # agregamos ruido

X_train, X_test, y_train, y_test  =  train_test_split(X, y, random_state=1)
regr = MLPRegressor( random_state = 1, max_iter=800).fit(X_train, y_train)
a=regr.predict(X_test[:2])
```

```
b=regr.score(X_test, y_test)
print(a)  ; print(b)

y_test_predecido = regr.predict(X_test)

plt.rc('font', size=16) ; plt.rc('axes', titlesize=18)
fig, ax1 = plt.subplots(figsize=(8,4))
n = np.arange((len(y_test)))
marcadores = ['o', (5,1), ',', 'v', '^', '<', '>', 's', 'd', '.', 'x', '+']
ax1.plot(n, y_test ,label='Valor Real')
ax1.plot(n, y_test_predecido ,label='Valor Predecido')
ax1.grid() ;ax1.legend()
ax1.set_xlabel('Número de muestra')
plt.tight_layout()
```

Resultados

Valores predecidos para las primeras 2 muestras

[-5.26701147 4.60825193]

Puntuación de confianza para los datos de prueba

0,88949

En la Fig. 220 se muestran los valores reales y los valores predecidos, se observa un correcto ajuste de datos

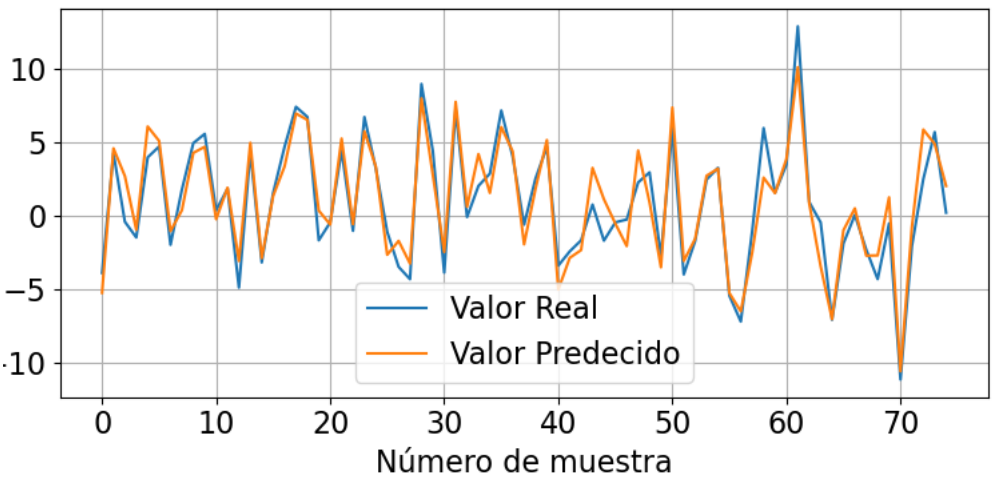


Fig. 220 – Resultados del ajuste de datos: Valores reales y valores predecidos

Ejercicio 5.15

Clasificación de imágenes con librerías Pytorch y conjunto de datos CIFAR10

CIFAR10 es una colección de datos que contiene 60000 imágenes separadas en 10 clases. De las cuales 50000 imágenes son para entrenamiento y el resto para prueba

Las librerías Torch se importan con los siguientes comandos

```
import torch
import torchvision
```

Se puede definir la red convolucional (CNN) de la siguiente manera

```
import torch.nn as nn1
import torch.nn.functional as F
class Net1(nn1.Module):
   def __init__(self):
      super().__init__()  #
      self.conv1  = nn1.Conv2d(3, 6, 5)  # capa convolucional
      self.pool   = nn1.MaxPool2d(2, 2) # capa pooling
      self.conv2  = nn1.Conv2d(6, 16, 5) # capa convolucional
      self.fc1    = nn1.Linear(16 * 5 * 5, 120) # capa lineal
      self.fc2    = nn1.Linear(120, 84) # capa lineal
      self.fc3    = nn1.Linear(84, 10) # capa lineal
   def forward(self, x):  #
      x1 =  self.pool(F.relu(self.conv1(x))) #
      x1 =  self.pool(F.relu(self.conv2(x))) #
      x1 =  torch.flatten(x, 1) # flatten todos excepto batch
      x1 =  F.relu(self.fc1(x)) #
      x1 =  F.relu(self.fc2(x)) #
      x1 =  self.fc3(x) #
      return x1
net = Net1()
```

Solicite el código al autor

Se muestran algunas imágenes del conjunto de datos CIFAR10, ver Fig. 221.

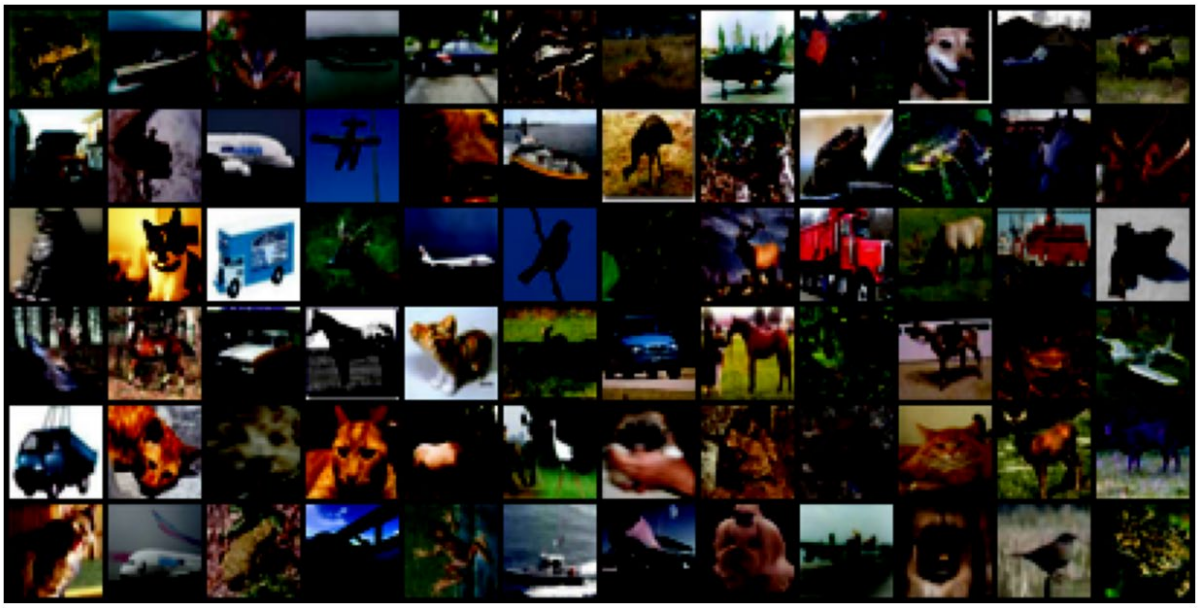


Fig. 221 – Ejemplo de imágenes del conjunto de datos CIFAR10

Resultados

A modo de ejemplo se espera obtener la siguiente respuesta al clasificar 4 imágenes:

Caballo gato gato automóvil

Consultar con el autor del libro el código del ejemplo.

Descarga de los códigos de los ejercicios

Anexo I – Tabla de Transformadas

Tabla X –Transformada Continua de Fourier

$f(t) \longleftrightarrow F(w)$; $f(t) = \frac{1}{2.\pi}.\int_{-\infty}^{\infty} F(w).e^{j.w.t}.dw$; $\boxed{F(w) = \int_{-\infty}^{\infty} f(t).e^{-j.w.t}.dt}$

$k \longleftrightarrow 2.\pi.k.\delta(w)$; $k.\delta(t) \longleftrightarrow k$

$a_1.f_1(t) + a_2.f_2(t) \longleftrightarrow a_1.F_1(w) + a_2.F_2(w)$ Linealidad

$f(t \pm t_o) \longleftrightarrow e^{\pm j.w.t_0}F(w)$ Desplazamiento temporal

$e^{\pm j.w_0}.f(t) \longleftrightarrow F(w \mp w_0)$ Desplazamiento Frecuencial

$f_1(t) * f_2(t) \longleftrightarrow F_1(w).F_2(w)$ Convolucion temporal

$f_1(t).f_2(t) \longleftrightarrow \frac{1}{2.\pi}F_1(w) * F_2(w)$ Producto en el tiempo

$\cos(w_0.t) \longleftrightarrow \pi.[\delta(w - w_0) + \delta(w + w_0)]$

$\text{sen}(w_0.t) \longleftrightarrow j.\pi.[\delta(w + w_0) - \delta(w - w_0)]$

Señal Periódica con Cn: coeficientes de la Serie de Fourier:

$$f(t) = \sum_{n=-\infty}^{\infty} C_n.e^{j.n,w_0.t} \longleftrightarrow F(w) = 2.\pi \sum_{n=-\infty}^{\infty} C_n.\delta(w - n.w_0) \; ; \; w_0 = \frac{2.\pi}{T} \; ; \; T: período\ de\ f(t)$$

$A.p_{2.T}(t) \longleftrightarrow A.2.T.sinc(w.T)$ Transformada del pulso.

$f(t) \longleftrightarrow F(w)$

$F(t) \longleftrightarrow 2.\pi.f(-w)$ Dualidad.

$\frac{d^n f(t)}{dt^n} \longleftrightarrow (j.w)^n.F(w)$; $n \epsilon N$ Derivada en dominio temporal.

$$Triangulo: \quad \Delta(t) = \begin{cases} A.\left(1 - \frac{|t|}{T}\right), & |t| < T \\ 0, & |t| > T \end{cases} \longleftrightarrow A.T.sinc^2\left(\frac{w.T}{2}\right)$$

$e^{-a..t}.u(t) \longleftrightarrow \frac{1}{a+j.w}$; $e^{-a.|t|} \longleftrightarrow \frac{2.a}{a^2+w^2}$;

$f_1(t) = f(t).\cos(w0.t) \longleftrightarrow F_1(w) = \frac{1}{2}.F(w - w_0) + \frac{1}{2}.F(w + w_0)$ Modulación

$f_1(t) = f(t).\text{sen}(w0.t) \longleftrightarrow F_1(w) = \frac{1}{2.j}.F(w - w_0) - \frac{1}{2.j}.F(w + w_0)$ Modulación

$e^{-a..t}sen(w_0.t).u(t) \longleftrightarrow \frac{w_0}{(a+j.w)^2+{w_0}^2}$; $e^{-a..t}cos(w_0.t).u(t) \longleftrightarrow \frac{a+j.w}{(a+j.w)^2+{w_0}^2}$

$u(t) \longleftrightarrow \frac{1}{j.w} + \pi.\delta(w)$; $f(a.t), con\ a \neq 0 \longleftrightarrow \frac{1}{|a|}F(w/a)$

$\int_{-\infty}^{t} f(t').dt' \longleftrightarrow \frac{1}{j.w}F(w) + \pi.F(0).\delta(w)$; $(-j.t)^n.f(t)\ ; n \in \aleph \longleftrightarrow \frac{d^n F(w)}{dw^n}$;

$\frac{f(t)}{-j.t} \longleftrightarrow \int_{-\infty}^{w} F(w').dw'$; $e^{\pm j.w_0.t} \longleftrightarrow 2.\pi.\delta(w \mp w_0)$

$sgn(t) \longleftrightarrow \frac{2}{j.w}$; $\frac{t^{n-1}}{(n-1)!} e^{-a..t}.u(t) \longleftrightarrow \frac{1}{(a+j.w)^n}$; $t^n;\ n \in N \longleftrightarrow 2.\pi.j^n.\frac{d^n\delta(w)}{dw^n}$

$\int_{-\infty}^{\infty} f_1(t).\overline{f_2}(t).dt \longleftrightarrow \frac{1}{2.\pi}\int_{-\infty}^{\infty} F_1(w).\overline{F_2}(w).dw$;

$\sum_{n=-\infty}^{\infty} \delta(t - n.T_s) \longleftrightarrow F(w) = w_s \sum_{n=-\infty}^{\infty} \delta(w - n.w_s);\ w_s = \frac{2.\pi}{T_s}$ Tren de Impulsos en t y en w

Tabla XI – Transformadas de Laplace (T.L.)

$f(t) = \frac{1}{2.\pi.j}\int_{\sigma-j.\infty}^{\sigma+j.\infty} F(s).e^{s.t}.ds$; $\boxed{s = \sigma + j.w}$ $F(s) = \int_0^{+\infty} f(t).e^{-s.t}.dt$; $f(t) \leftrightarrow F(s)$	
$a.f_1(t) + b.f_2(t) \longleftrightarrow a.F_1(s) + b.F_2(s)$ Linealidad	$k.\delta(t) \longleftrightarrow k$; $\forall s$
$f(a.t) \longleftrightarrow \frac{1}{a}.F\left(\frac{s}{a}\right)$; $a \neq 0$	$u(t) \longleftrightarrow \frac{1}{s}$; $\Re e(s) > 0$
$f(t - t_0) \longleftrightarrow e^{-s.t_0}.F(s)$; $R' = R$	$\cos(w_0.t).u(t) \longleftrightarrow \frac{s}{s^2+{w_0}^2}$; $\Re e(s) > 0$ $e^{-a.t}.\cos(w_0.t).u(t) \longleftrightarrow \frac{s+a}{(s+a)^2+{w_0}^2}$
$e^{\pm a.t}.f(t) \longleftrightarrow F(s \mp a)$; $R' = R + Re(a)$	$\text{sen}(w_0.t).u(t) \longleftrightarrow \frac{w_0}{s^2+{w_0}^2}$; $\Re e(s) > 0$ $e^{-a.t}.\text{sen}(w_0.t).u(t) \longleftrightarrow \frac{w_0}{(s+a)^2+{w_0}^2}$
$f_1(t) * f_2(t) \longleftrightarrow F_1(s).F_2(s)$	$e^{\pm a.t}.u(t) \longleftrightarrow \frac{1}{s\mp a}$; $\Re e(s) > -\Re e(a)$ $e^{-a.t}.u(t) \longleftrightarrow \frac{1}{s+a}$
$f_1(t).f_2(t) \longleftrightarrow \int_{c-j.\infty}^{c+j.\infty} F_1(\tau).F_2(s-\tau).d\tau$	$-e^{-a.t}.u(-t) \longleftrightarrow \frac{1}{s+a}$; $\Re e(s) < -\Re e(a)$
$\frac{d^n f(t)}{dt^n} \longleftrightarrow s^n.F(s) - s^{n-1}.f(0) - s^{n-2}.f'(0)\ \ldots\ldots\ldots - f^{n-1}(0)$ $n \in \aleph$; $t \geq 0$	$t^n.e^{\pm a.t}.u(t) \longleftrightarrow \frac{n!}{(s\mp a)^{n+1}}$; $n \in \aleph$
$f(t) \leftrightarrow F(s)$ $f(0) = \lim_{s\to\infty} s.F(s)$ Teorema del Valor Inicial: TVI	$(-t)^n.f(t) \longleftrightarrow \frac{d^n F(s)}{ds^n}$; $n \in \aleph$
$\lim_{t\to\infty} f(t) = \lim_{s\to 0} s.F(s)$ Teorema del Valor Final: TVF	$\int_{-\infty}^{t} f(t').dt' \longleftrightarrow \frac{F(s)}{s} + \frac{\int_{-\infty}^{0} f(t).dt}{s}$

Menos usadas:	
$\lim_{t\to\infty} f^{(n-1)}(t) = \lim_{s\to 0} s^n.F(s)$	$\sum_{n=-\infty}^{\infty} \delta(t-n.T) \longleftrightarrow \frac{1}{1-e^{-s.T}}$
$f(t-t_0).u(t-t_0) \longleftrightarrow e^{-t_0.s}.F(s)$	$\frac{d^n\delta(t)}{dt^n} \longleftrightarrow s^n$
$f(t)=f(t+T) \longleftrightarrow \frac{1}{1-e^{-s.T}} \int_0^T f(t).e^{-s.T}.dt$	$\frac{f(t)}{t} \longleftrightarrow \int_s^{\infty} F(u).du$
$e^{-a.\lvert t\rvert} \longleftrightarrow \frac{2.a}{a^2-s^2}$; con $F_b(s) = \int_{-\infty}^{+\infty} f(t).e^{-s.t}.dt$	$(1-e^{-a.t}).u(t) \longleftrightarrow \frac{a}{s.(s+a)}$
$\cosh(w_0.t).u(t) \longleftrightarrow \frac{s}{s^2-{w_0}^2}$; coseno hiperbólico	$\text{senoh}(w_0.t).u(t) \longleftrightarrow \frac{w_0}{s^2-{w_0}^2}$ seno hiperbólico

Tabla XII –Transformadas Z (T.Z)

TZB: $F_b(z)=\sum_{n=-\infty}^{\infty} f[n].z^{-n}$; TZU: $F(z)=\sum_{n=0}^{\infty} f[n].z^{-n}$ $f[n]=\frac{1}{2.\pi.j}.\int_c F(z).z^{-n}.dz$; $n\in Z$	
Propiedades TZU y TZB	
Linealidad: $a.f_1[n]+b.f_2[n] \longleftrightarrow a.F_1(z)+b.F_2(z)$	Convolución $f[n]*g[n] \longleftrightarrow F(z).G(z)$
$n^k.f[n] \longleftrightarrow (-1)^k.\left(z\frac{d}{dz}\right)^k F(z)$; $k\in \aleph\cup\{0\}$ $n.f[n] \longleftrightarrow -z.\frac{dF(z)}{dz}$	$a^{\pm n}.f[n] \longleftrightarrow F\left(a^{\mp 1}.z\right)$; $a\neq 0$
$\overline{f}[n] \longleftrightarrow \overline{F}(\overline{z})$; $\Re\{f[n]\} \longleftrightarrow \frac{1}{2}\left(F(z)+\overline{F}(\overline{z})\right)$	$Img\{f[n]\} \leftrightarrow \frac{1}{2.j}\left(F(z)-\overline{F}(\overline{z})\right)$
Propiedad de Desplazamiento Temporal	
$\boxed{f[n\pm a] \longleftrightarrow z^{\pm a}.F(z)}$; para CIN TZB y TZU Condiciones No Nulas -TZU $f[n+a] \longleftrightarrow z^a.\left[F(z)-f_{(0)}-z^{-1}.f_{(1)} \ldots .-z^{-a+1}.f_{(a-1)}\right]$; $a\in\aleph$ TZU	Condiciones No Nulas -TZU $f[n-a] \longleftrightarrow z^{-a}.F(z)+f_{(-a)}+z^{-1}.f_{(-a+1)}+z^{-2}.f_{(-a+2)} \ldots .+z^{-a+1}.f_{(-1)}$; $a\in\aleph$ TZU
Transformadas	
$\boxed{\delta[n] \longleftrightarrow 1;\ \forall z\in\mathbb{Z}}$ TZU y TZB	$\boxed{\delta[n-m] \longleftrightarrow z^{-m};\ \forall z\in\mathbb{Z}-\{0\}}$ TZU y TZB
$a^n.u[n] \longleftrightarrow \frac{z}{z-a}$; $\lvert z\rvert>\lvert a\rvert$;	Anti causal TZB

Si $a = 1$; $u[n] \longleftrightarrow \frac{z}{z-1}$; $\|z\| > 1$ TZU Y TZB $n.a^n.u[n] \longleftrightarrow \frac{a.z}{(z-a)^2}$; $\|z\| > \|a\|$ $n.a^{n-1}.u[n] \longleftrightarrow \frac{z}{(z-a)^2}$; $\|z\| > \|a\|$; TZU $n.u[n] \longleftrightarrow \frac{z}{(z-1)^2}$; $\|z\| > 1$ TZU $(n+1).a^n.u[n] \longleftrightarrow \left(\frac{z}{z-a}\right)^2$; $\|z\| > \|a\|$	$-n.a^n.u[-n-1] \longleftrightarrow \frac{a.z}{(z-a)^2}$; $\|z\| < \|a\|$ $-a^n.u[-n-1] \longleftrightarrow \frac{1}{1-a.z^{-1}} = \frac{z}{z-a}$; $\|z\| < \|a\|$ Si $a = 1$: $-u[-n-1] \longleftrightarrow \frac{1}{1-z^{-1}} = \frac{z}{z-1}$; $\|z\| < \|1\|$
Propiedades y Transformadas Exclusivas de TZB	
$e^{j.\Omega_0.n}.f[n] \longleftrightarrow F\left(e^{-j.\Omega_0}.z\right)$; TZB	$f[-n] \longleftrightarrow F\left(\frac{1}{z}\right)$ TZB
$\sum_{k=-\infty}^{n} f[n] \longleftrightarrow \frac{1}{1-z^{-1}}.F(z)$ Acumulación TZB	$\begin{cases} a^n ; & 0 \le n \le N-1 \\ 0 ; & otro\ caso \end{cases} \longleftrightarrow \frac{1-a^N.z^{-N}}{1-a.z^{-1}}$; $\|z\| > 0$ TZB
Transformadas y Propiedades Exclusivas de TZU	
$f[0] = \lim_{z\to\infty} F(z)$ Teorema del Valor Inicial (TVI) para TZU	$\lim_{n\to\infty} f[n] = \lim_{z\to 1}(1-z^{-1}).F(z)$ Teorema del Valor Final (TVF) para TZU
$\cos(\Omega_0.n).u[n] \longleftrightarrow \frac{z.(z-\cos(\Omega_0))}{z^2-2.z.\cos(\Omega_0)+1}$; $\|z\| > 1$ TZU	$(r^n \cos(\Omega_0.n)).u[n] \longleftrightarrow \frac{z.(z-r.\cos(\Omega_0))}{z^2-(2.r.\cos(\Omega_0)).z+r^2}$; $\|z\| > \|r\|$
$\text{seno}(\Omega_0.n).u[n] \longleftrightarrow \frac{z.seno(\Omega_0)}{z^2-2.z.\cos(\Omega_0)+1}$; $\|z\| > 1$ TZU	$(r^n \text{seno}(\Omega_0.n)).u[n] \longleftrightarrow \frac{r.z.\text{seno}(\Omega_0)}{z^2-(2.r.\cos(\Omega_0)).z+r^2}$; $\|z\| > \|r\|$
$\frac{f[n]}{n} \longleftrightarrow \int_z^{\infty} \frac{F(z')}{z'} dz' + \lim_{n\to\infty} \frac{f[n]}{n}$; TZU $n \in \aleph$	

Tabla XIII – Transformada Fourier de Tiempo Discreto (TFTD)

$f[n] \longleftrightarrow F(\Omega)$

$\boxed{F(\Omega) = \sum_{n=-\infty}^{\infty} f[n].e^{-j.\Omega.n}}$; $\boxed{f[n] = \frac{1}{2.\pi}\int_{-\pi}^{\pi} F(\Omega).e^{j.\Omega.n}.d\Omega}$

$x[n] \longleftrightarrow X(\Omega)$

$\delta[n] \longleftrightarrow 1$ (tablas) ; $x[n-n_0] \longleftrightarrow e^{-j.\Omega.\ n_0}.\ X(\Omega)$ (tablas)

$a^n u[n] \leftrightarrow \frac{1}{1-a.e^{-j\Omega}}$; $|a| < 1$

Anexo II – Autovectores y Autovalores

Dada una matriz $A = \begin{pmatrix} 3 & 2 \\ 1 & 4 \end{pmatrix}$

Multiplicamos esa matriz por un vector:

$$A.\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 3 & 2 \\ 1 & 4 \end{pmatrix}.\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 3.x + 2.y \\ x + 4.y \end{pmatrix}$$

Dado el siguiente recinto R, multiplicamos todos sus puntos por la matriz A

¿En que se transforma el recinto ?

Resolución:

El vector nulo se transforma en vector nulo

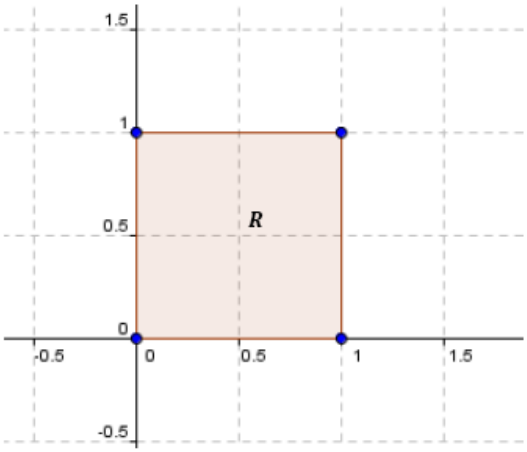


Fig. 222 – Recinto R original

Resolución:

El vector nulo se transforma en vector nulo. Analizamos los otros 3 puntos

$$A.\begin{pmatrix} x \\ y \end{pmatrix} = A.\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 3 & 2 \\ 1 & 4 \end{pmatrix}.\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 3 \\ 1 \end{pmatrix}$$

$$A.\begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 3 & 2 \\ 1 & 4 \end{pmatrix}.\begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 2 \\ 4 \end{pmatrix}$$

$$A.\begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 5 \\ 5 \end{pmatrix}$$

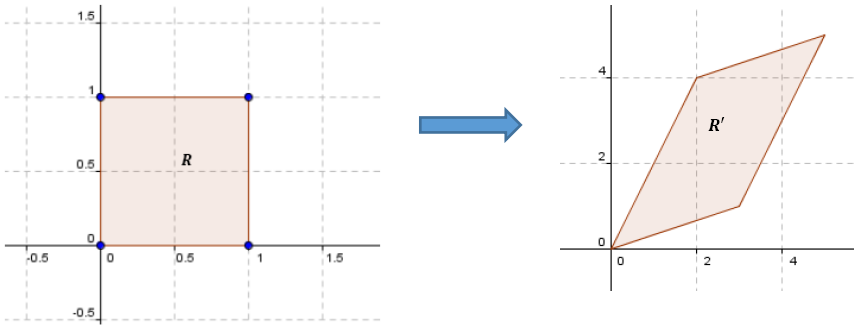


Fig. 223 – Recinto R original y transformado

Después de la Transformación, ¿Hay vectores que conservan su dirección?

- Analizamos vector (1,0): se convierte en el vector (3,1). Se modifica la dirección del vector.
- El (0,1) se convierte en (2,4). Se modifica la dirección.
- El (1,1) se convierte en (5,5). Se observa una dilatación por 5, y se mantiene la dirección.

Se llama autovector al vector que no modifico su dirección. Y el factor 5 por el cual se dilató corresponde al autovalor.

$$T\{(1,1)\} = A.\begin{pmatrix}1\\1\end{pmatrix} = 5.\begin{pmatrix}1\\1\end{pmatrix}$$

En este caso tenemos Autovalor: 5 y Autovector $\begin{pmatrix}1\\1\end{pmatrix}$

Definición de Autovalores y Autovectores

Dado $A \in \mathbb{R}^{nxn}$,

$\lambda \in \mathbb{R}$ es un Autovalor de A si y sólo si se puede obtener un vector $v \in \mathbb{R}^{nx1}$ distinto del vector nulo que cumpla:

$$\boxed{\boldsymbol{A.v = \lambda.v}} \quad ; \quad \text{con} \quad v \neq 0_V$$

v es el Autovector asociado a λ.

En este ejemplo el vector transformado de (1,1) es (5,5), de esta manera, para hallar los Autovalores y Autovectores utilizamos:

Por definición: $\boldsymbol{A.v = \lambda.v}$

Restamos en ambos miembros: $\boldsymbol{\lambda.v}$

$$\boldsymbol{A.v - \lambda.v = 0_V}$$

En la ecuación anterior, premultiplicamos a $\boldsymbol{v}$ por matriz Identidad $\boldsymbol{I}$

$$\boldsymbol{A.v - \lambda.I.v = 0_V}$$

$$\boxed{\boldsymbol{(A - \lambda.I).v = 0_V}}$$

$(\boldsymbol{A - \lambda.I})$ tamaño $\boldsymbol{nxn}$.

Es un sistema Homogéneo que tiene n ecuaciones y n incógnitas.

Sistema Homogéneo respecto a su compatibilidad

Resulta siempre compatible, debido a que siempre tiene como solución la solución trivial.

En estos casos los sistemas resultan: SCD o SCI. Sistemas compatibles determinados (SCD) si tiene solo la solución trivial y v vector nulo. O pueden ser Sistemas compatibles determinados (SCI). La idea es que sean SCI con vector no nulo.

Hay que encontrar un SCI.

Para un sistema cuadrado y homogéneo, se analiza su determinante: en caso de ser distinto de cero, solo tendrá solución única. Por este motivos, necesitamos que sea igual a cero:

$$(\boldsymbol{A} - \boldsymbol{\lambda}.\boldsymbol{I}) = \mathbf{0}$$

Esta ***ecuación se conoce como ecuación característica correspondiente a la matriz $\boldsymbol{A}$.***

Polinomio característico de $\boldsymbol{A}$:

$\boldsymbol{p(\lambda) = det(A - \lambda.I)}$ ***es el polinomio con grado n dependiente de λ***

Las raíces de este polinomio característico corresponden a los autovalores de $\boldsymbol{A}$.

Ahora debemos encontrar los autovectores. Analizaremos la expresión anterior.

Para cada $\boldsymbol{\lambda}$ hay que resolver el sistema: $(\boldsymbol{A} - \boldsymbol{\lambda}.\boldsymbol{I}).\boldsymbol{v} = 0_V$, para poder hallar sus autovectores.

Para una matriz de 2 por 2, se tiene un polinomio de grado 2.

Para una matriz de 3 por 3, se tiene un polinomio de grado 3.

También se utilizan estos nombres para los autovectores y autovalores:

- *Valores propios* y *vectores propios.*
- Utilizando la raíz del idioma alemán eigen: *Eigenvalores* y *Eigenvectores*.

Ejemplo 1: Analizamos el ejemplo anterior.

$$\boldsymbol{A - \lambda.I} = \begin{pmatrix} 3 & 2 \\ 1 & 4 \end{pmatrix} - \boldsymbol{\lambda}.\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} \boldsymbol{3-\lambda} & 2 \\ 1 & \boldsymbol{4-\lambda} \end{pmatrix}$$

$\mathbf{det}(\boldsymbol{A} - \boldsymbol{\lambda}.\boldsymbol{I}) = \mathbf{0}$; $(\mathbf{3} - \boldsymbol{\lambda}).(\mathbf{4} - \boldsymbol{\lambda}) - \mathbf{2} = \mathbf{0}$; $\boldsymbol{\lambda^2} - \mathbf{7}.\boldsymbol{\lambda} + \mathbf{10} = \mathbf{0}$

$\boldsymbol{\lambda_1} = \mathbf{2}$; $\boldsymbol{\lambda_2} = \mathbf{5}$; Usamos: $(\boldsymbol{A} - \boldsymbol{\lambda}.\boldsymbol{I}).\boldsymbol{v} = 0_V$

Para $\boldsymbol{\lambda} = \mathbf{2}$ resolvemos el sistema de ecuaciones:

$$(\boldsymbol{A} - \mathbf{2}.\boldsymbol{I}).\begin{pmatrix} \boldsymbol{x} \\ y \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \end{pmatrix} \quad ; \quad \begin{pmatrix} 1 & 2 \\ 1 & 2 \end{pmatrix}.\begin{pmatrix} \boldsymbol{x} \\ y \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \end{pmatrix} \; ; \; \begin{cases} \boldsymbol{x} + \mathbf{2}.\boldsymbol{y} = \mathbf{0} \\ \boldsymbol{x} + \mathbf{2}.\boldsymbol{y} = \mathbf{0} \end{cases}$$

$$\boxed{\boldsymbol{x} = -\mathbf{2}.\boldsymbol{y}}$$

La solución para sistema homogéneo corresponde a un subespacio. Los subespacios que corresponden a los autovectores se denominan *autoespacios*.

Hallamos una base del subespacio: $S_2 = gen\left\{\begin{Bmatrix} \mathbf{2} \\ \mathbf{-1} \end{Bmatrix}\right\}$

Es el subespacio para los autovectores correspondientes al autovalor 2.

Para $\boldsymbol{\lambda}$ =5 resolvemos el sistema de ecuaciones:

$$(\boldsymbol{A}-\mathbf{5}.\boldsymbol{I}).\begin{pmatrix}\boldsymbol{x}\\ y\end{pmatrix}=\begin{pmatrix}\mathbf{0}\\ \mathbf{0}\end{pmatrix} \quad ; \quad \begin{pmatrix}-2 & 2\\ 1 & -1\end{pmatrix}.\begin{pmatrix}\boldsymbol{x}\\ y\end{pmatrix}=\begin{pmatrix}\mathbf{0}\\ \mathbf{0}\end{pmatrix} \; ; \quad \begin{cases}-\mathbf{2}.\boldsymbol{x}+\mathbf{2}.\boldsymbol{y}=\mathbf{0}\\ \boldsymbol{x}-\boldsymbol{y}=\mathbf{0}\end{cases}$$

$$\boxed{\boldsymbol{x}=\boldsymbol{y}}$$

$$S_5 = gen\left\{\begin{pmatrix}\mathbf{1}\\ \mathbf{1}\end{pmatrix}\right\}$$

Subespacio donde están los autovectores asociados al autovalor 5

$\mathbf{5}.\begin{pmatrix}\mathbf{1}\\ \mathbf{1}\end{pmatrix}=\begin{pmatrix}\mathbf{5}\\ \mathbf{5}\end{pmatrix}$; Se verifica que el punto $(\mathbf{1},\mathbf{1})$ se transforma en $(\mathbf{5},\mathbf{5})$

Resolviendo con Matlab®

Ejercicio 1

Autovalores y autovectores

```
close all; clear all; clc
A=[3 2; 1 4] ;
autovalores = eig(A) % Devuelve 2 y 5 !!
% Autovalores y autovectores (izquierda y derecha)
[ V , D , W ] = eig( A ) ;
V1 = V(:1)
% Devuelve autovector:   V1   -0.8944   0.4472
V2 = V(:,2)
% Devuelve otro autovector:    V2  -0.7071  -0.7071
p1=[1; 1] %punto p1 que quiero transformar
A*p1      % devuelve 5 5
autovalores(2)*p1    % devuelve 5 5   ok !

p2=[3; 3] %punto p2 que quiero transformar
A*p2      % devuelve 15 15
autovalores(2)*p2    % devuelve 15 15   ok !
p3=[1; 3]
A*p3      % devuelve 9 13
autovalores(2)*p3    % devuelve 5 15
% No se encuentra en la misma dirección ! No alcanza con un autovector
```

V1 y V2 Son los autovectores

Cumplen con los subespacios: $S_2 = gen\left\{\begin{pmatrix}\mathbf{2}\\ \mathbf{-1}\end{pmatrix}\right\}$; $S_5 = gen\left\{\begin{pmatrix}\mathbf{1}\\ \mathbf{1}\end{pmatrix}\right\}$

¡¡PROPORCIONALES a V1 y V2!!

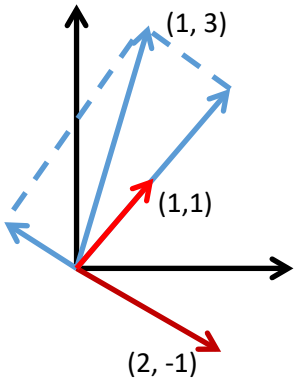


Fig. 224 – Autovectores y autovalores

Ejercicio 2: Propuesto

Autovalores y autovectores

Dada una matriz $A = \begin{pmatrix}2 & 4\\ 4 & 2\end{pmatrix}$

Y dado el punto p1=[3; 3], es decir x=3 e y=3

Mediante Matlab® se pide:

a) Obtener multiplicación de la matriz A por el punto p1

b) Ídem utilizando Autovalores

c) Escribir comentarios:

Resultados del punto a) y del punto b).

¿Coinciden los resultados?

Ejercicio 3: Propuesto

Autovalores y autovectores

Analizar el siguiente código Matlab®

```
A=[1 3; -1 2] ;
% Hallamos Autovalores
autovalores= eig(A)
% Hallamos Autovalores y Autovectores
```

```
[vectores valores] = eig(A) ;
A*vectores(:,1)
valores(1,1)*vectores(:,1)
```

Descarga de los códigos de los ejercicios

Referencias

Libros

Beale, M. H., Hagan, M., & Demuth, H. (2020). Deep Learning Toolbox - Getting Started Guide. In MathWorks.

Beale, M. H., Hagan, M., & Demuth, H. (2020). Deep Learning Toolbox™ User's Guide. In MathWorks. https://la.mathworks.com/help/deeplearning/index.html

Demuth H, Beale M, Hagan M. (2018). Neural Network Toolbox™ User's Guide Neural network toolbox. MathWorks.

Diniz, Paulo S. (2020). Adaptive filtering: Algorithms and Practical Implementation, 5th ed. Cham: Springer. ISBN: 978-3-030-29057-3 https://doi.org/10.1007/978-3-030-29057-3

Farhang-Boroujeny, B. (2013). Adaptive Filters: Theory and Applications, Second Edition. In Adaptive Filters: Theory and Applications, Second Edition. https://doi.org/10.1002/9781118591352

Girod, B. Rabenstein, R. Stenger, A. (2001). Signals and Systems. Wiley ISBN 13: 9780471988007. ISBN 10: 0471988006

Gray, R. M., & Davisson, L. D. (2004). An Introduction to Statistical Signal Processing. In An Introduction to Statistical Signal Processing. https://doi.org/10.1017/cbo9780511801372

Haykin, Simon S. (2014) Adaptive filter theory, 5th Edition. Upper Saddle River, New Jersey: Pearson. Print. ISBN-13: 978-0132671453 / ISBN-10: 013267145X

Haykin, S. (2008). Neural Networks and Learning Machines. In Pearson Prentice Hall New Jersey USA 936 pLinks (Vol. 3).

Hagan, M. T., Demuth, H. B., Beale, M. H., & De Jesus, O. (2014). Neural Network Design 2nd Edition (2014). In Neural Networks in a Softcomputing Framework.

Papoulis, A., & Hoffman, J. G. (1967). A. Papoulis and S.U. Pillai, Probability, random variables, and stochastic processes, 4th ed., McGraw-Hill, Boston, 2002. In Physics Today (Vol. 20, Issue 1).

Poularikas, Alexander D., and Zayed M. Ramadan (2006). Adaptive filtering primer with MATLAB. Boca Raton: CRC/Taylor & Francis. Print. ISBN-13: 978-0849370434. ISBN-10: 0849370434

Poularikas, A. D., & Ramadan, Z. M. (2017). Adaptive Filtering Primer With Matlab. In Adaptive Filtering Primer with Matlab. https://doi.org/10.1201/9781315221946

Poularikas, A. D., & Ramadan, Z. M. (2019). Wiener filters. In Adaptive Filtering Primer With Matlab. https://doi.org/10.1201/9781315221946-4

Proakis, J. G., & Manolakis, D. (1998). Digital Signal Processing: Algoritms and applications. Pentice Hall.

Sayed, Ali H. (2008) Adaptive filters. Hoboken, N.J: Wiley-Interscience IEEE Press. ISBN ISBN:9780470253885. https://doi.org/10.1002/9780470374122

Shaw, Z. (2018). Learn more Python 3 the hard way: the next step for new Python programmers. Boston: Addison-Wesley.

Tan, Li. (2008) Digital signal processing: fundamentals and applications. Amsterdam Boston: Academic Press, 2008. ISBN 978-0-12-374090-8

Tan, L., & Jiang, J. (2018). Digital signal processing: Fundamentals and applications. In Digital Signal Processing: Fundamentals and Applications. https://doi.org/10.1016/C2017-0-02319-4

Webb, A. R., & Copsey, K. D. (2011). Statistical Pattern Recognition: Third Edition. In *Statistical Pattern Recognition: Third Edition*. https://doi.org/10.1002/9781119952954

Páginas web

The Mathworks, Inc. MATLAB, Version 2019. MATLAB 2019b - MathWorks. In www.Mathworks.Com/Products/Matlab.

The Mathworks Inc. (2019). MATLAB (R2019a). In The MathWorks Inc.

Scikit-learn, Machine Learning in Python, https://scikit-learn.org/stable/

Pytorch,https://pytorch.org/

Tensorflow, https://www.tensorflow.org/

Artículos de Revistas

Atal, D. K., & Singh, M. (2020). Arrhythmia Classification with ECG signals based on the Optimization-Enabled Deep Convolutional Neural Network. *Computer Methods and Programs in Biomedicine*, *196*. https://doi.org/10.1016/j.cmpb.2020.105607

Abadi, M. S. E., Mesgarani, H., & Khademiyan, S. M. (2021). Two Improved Wavelet Transform Domain LMS Sign Adaptive Filter Algorithms Against Impulsive Interferences. Circuits, Systems, and Signal Processing, 40(2). https://doi.org/10.1007/s00034-020-01508-5

Al-Safi, A. (2021). ECG signal denoising using a novel approach of adaptive filters for real-time processing. International Journal of Electrical and Computer Engineering, 11(2). https://doi.org/10.11591/ijece.v11i2.pp1243-1249

Bai, Y., Wang, X., Jin, X., Su, T., Kong, J., & Zhang, B. (2020). Adaptive filtering for MEMS gyroscope with dynamic noise model. ISA Transactions, 101. https://doi.org/10.1016/j.isatra.2020.01.030

Beatriz Cuellar, H., de la Cruz-Alejo, J., Enciso Contreras, E., & Alcocer Guillermo, I. C. (2021). Design and Implementation of a LMS Adaptive Filter for Humidity and Temperature Control in a Bioreactor. https://doi.org/10.1007/978-3-030-72208-1_10

Bershad, N. J., & Bermudez, J. C. M. (2020). A switched variable step size NLMS adaptive filter. *Digital Signal Processing: A Review Journal*, *101*. https://doi.org/10.1016/j.dsp.2020.102730

Boveiri, H. R., Khayami, R., Javidan, R., & MehdiZadeh, A. R. (2020). Medical image registration using deep neural networks: A comprehensive review. In *arXiv*.

Cai, Y., Yu, Z., Mo, D., Liu, R., Chen, A., Dai, B., & Li, Y. (2020). Noise reduction with adaptive filtering scheme on interferometric fiber optic hydrophone. *Optik*, *211*. https://doi.org/10.1016/j.ijleo.2020.164648

Chandra, M., Goel, P., Anand, A., & Kar, A. (2021). Design and analysis of improved high-speed adaptive filter architectures for ECG signal denoising. *Biomedical Signal Processing and Control*, *63*. https://doi.org/10.1016/j.bspc.2020.102221

Das, L., Kumar Das, J., & Nanda, S. (2021). Effective Identification and Prediction of Breast Cancer Gene Using Volterra Based LMS/F Adaptive Filter. Advances in Intelligent Systems and Computing, 1199. https://doi.org/10.1007/978-981-15-6353-9_27

Gupta, J., Saini, S. K., & Juneja, M. (2020). Survey of denoising and segmentation techniques for MRI images of prostate for improving diagnostic tools in medical applications. Materials Today: Proceedings, 28. https://doi.org/10.1016/j.matpr.2020.05.023

Hirikude, S. M., & Patil, S. S. (2021). Design of iRLS Algorithm With/Without Pre-Filter for Antenna Beam Forming Technique. Wireless Personal Communications. https://doi.org/10.1007/s11277-021-08155-2

Jain, S., Ahirwal, M. K., Kumar, A., Bajaj, V., & Singh, G. K. (2017). QRS detection using adaptive filters: A comparative study. *ISA Transactions*, *66*. https://doi.org/10.1016/j.isatra.2016.09.023

Jakati, J. S., & Kuntoji, S. S. (2021). A noise reduction method based on modified lms algorithm of real time speech signals. WSEAS Transactions on Systems and Control,16. https://doi.org/10.37394/23203.2021.16.13

Jelfs, B., Sun, S., Ghorbani, K., & Gilliam, C. (2021). An Adaptive All-Pass Filter for Time-Varying Delay Estimation. IEEE Signal Processing Letters, 28. https://doi.org/10.1109/LSP.2021.3065889

Khan, A. A., Shah, S. M., Raja, M. A. Z., Chaudhary, N. I., He, Y., & Machado, J. A. T. (2021). Fractional LMS and NLMS Algorithms for Line Echo Cancellation. Arabian Journal for Science and Engineering. https://doi.org/10.1007/s13369-020-05264-1

Kundella, S., & Gobinath, R. (2020). Robust convolutional neural network for arrhythmia prediction in ECG signals. *Materials Today: Proceedings*. https://doi.org/10.1016/j.matpr.2020.10.579

La Rosa, A. B., Pereira, P. T. L., Ücker, P., Paim, G., da Costa, E. A. C., Bampi, S., & Almeida, S. (2021). Exploring NLMS-Based Adaptive Filter Hardware Architectures for Eliminating Power Line Interference in EEG Signals. Circuits, Systems, and Signal Processing. https://doi.org/10.1007/s00034-020-01620-6

Nagasirisha, B., & Prasad, D. V. K. D. V. (2021). EMG signal denoising using adaptive filters through hybrid optimization algorithms. Biomedical Engineering - Applications, Basis and Communications, 33(2). https://doi.org/10.4015/S1016237221500095

Pathan, A., Memon, T. D., & Sohu, F. K. (2021). FPGA based area-power-performance analysis of LMS filter using conventional and proposed multipliers. Journal of Physics: Conference Series, 1860(1). https://doi.org/10.1088/1742-6596/1860/1/012010

Salih, T. A. (2021). New approach to eliminate noise attendants ECG signal corrupted. Journal of Theoretical and Applied Information Technology, 99(3).

Sayed, A. H. (2014). Adaptive networks. *Proceedings of the IEEE*, *102*(4). https://doi.org/10.1109/JPROC.2014.2306253

Shin'ichi, K. (2021). Least mean square nonlinear regressor algorithm. European Signal Processing Conference, 2021-January. https://doi.org/10.23919/Eusipco47968.2020.9287688

Spelta, M. J. M., & Martins, W. A. (2020). Normalized LMS algorithm and data-selective strategies for adaptive graph signal estimation. *Signal Processing*, *167*. https://doi.org/10.1016/j.sigpro.2019.107326

Surekha, K. S., & Patil, B. P. (2021). Power line interference removal from electrocardiogram signal using multi-order adaptive LMS filtering. International Journal of Biomedical Engineering and Technology, 35(2). https://doi.org/10.1504/IJBET.2021.113330

Xu, W., Deng, S., Liang, D., & Cheng, X. (2021). A crown morphology-based approach to individual tree detection in subtropical mixed broadleaf urban forests using UAV lidar data. Remote Sensing, 13(7). https://doi.org/10.3390/rs13071278

Yu, Z., Cai, Y., & Mo, D. (2020). Comparative study on noise reduction effect of fiber optic hydrophone based on LMS and NLMS algorithm. Sensors (Switzerland), 20(1). https://doi.org/10.3390/s20010301

Procesamiento Avanzado de Señales en Sistemas Adaptativos y Redes Neuronales

Este libro es un resumen de mi experiencia como profesor y Doctor en Ingeniería en área del procesamiento de señales e imágenes, y tiene como objetivo introducir temas relacionados con el procesamiento avanzado de señales.

El procesamiento de señales e imágenes involucra diferentes áreas de las ciencias y de la ingeniería. Se puede utilizar en química, física, economía, automatización, ciencias sociales, biología, medicina, etc.

Si el entorno cambia los sistemas de procesamiento de datos deben ser dinámicos y modificar su funcionamiento rápidamente para adaptarse al entorno.

Se utilizan Sistemas Adaptativos que presentan muchas ventajas respecto de un sistema de respuesta fija con parámetros constantes. Muchas veces resulta indispensable utilizar un sistema adaptativo, si se plantea un sistema con filtro fijo puede fracasar el análisis.

En el presente libro se repasan conceptos de procesamiento digital de señales y se estudian sistemas adaptativos con parámetros variables mediante filtros adaptativos y redes neuronales. Se abarcan los temas mediante un marco teórico, con desarrollos matemáticos, ejercicios analíticos, aplicaciones variadas y ejercicios en Matlab® y Python.

ISBN 978-987-4998-66-8

edUTecNe

CiN REUN
Red de Editoriales de Universidades Nacionales de la Argentina

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES