CONCEPTOS BÁSICOS PDS, APLICACIONES EN MATLAB Y DSP.

Autor: Mauricio Sebastián Caggioli

GADIB: Grupo de Análisis, Desarrollo e Investigaciones Biomédicas.

Noviembre 2003

Índice

Introducción al PDS.

- Señal: Función que porta información acerca del estado o comportamiento de un sistema.
- Matemáticamente: las señales se representan como funciones de una o más variables.
- La variable independiente de dicha representación puede ser continua o discreta.
- Las señales en tiempo continuo están definidas sobre un tiempo continuo,
- Las señales en tiempo discreto están definidas sobre un conjunto discreto de valores y, por consiguiente, serán representadas por sucesiones o vectores que son como las tratadas en MATLAB.
- Las señales digitales son aquellas que tanto el tiempo, como la amplitud son discretas. Hay que notar bien esta diferencia entre señales digitales y sólo discretas.

Introducción al PDS.

• En el caso de sistemas procesados con microprocesadores, microcontroladores, DSP, etc la discretización o cuantificación de las amplitudes se deberá a la cantidad de bits con que puedo representar cada amplitud.

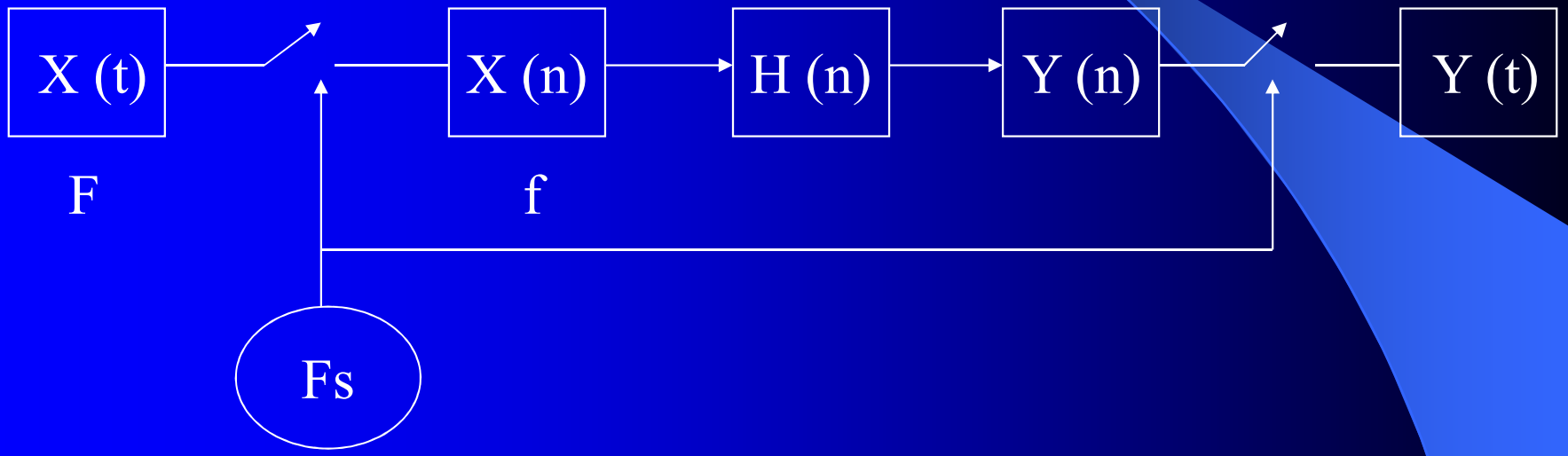
• La ventaja del tratamiento digital de señales con los elementos descriptos anteriormente es que la implementación de sistemas muy complicados de realizar en forma analógica se transforma en algo rutinario de programación para un tratamiento de este tipo.

• Cualquier cambio en el sistema será variar algún parámetro del programa, en cambio en forma analógica significa cambiar componentes, que la mayoría de las veces resulta tedioso y a veces imposible.

• La mayoría de las señales pueden ser digitalizadas y tratadas digitalmente, excepto aquellas que tengan un ancho de banda o frecuencia muy alta tal que no se pueda conseguir un oscilador para realizar el muestreo correspondiente (de por los menos el doble del ancho de banda o frecuencia máxima de la señal).

VOLVER

Muestreo y conceptos básicos.



$$t = n / Fs$$

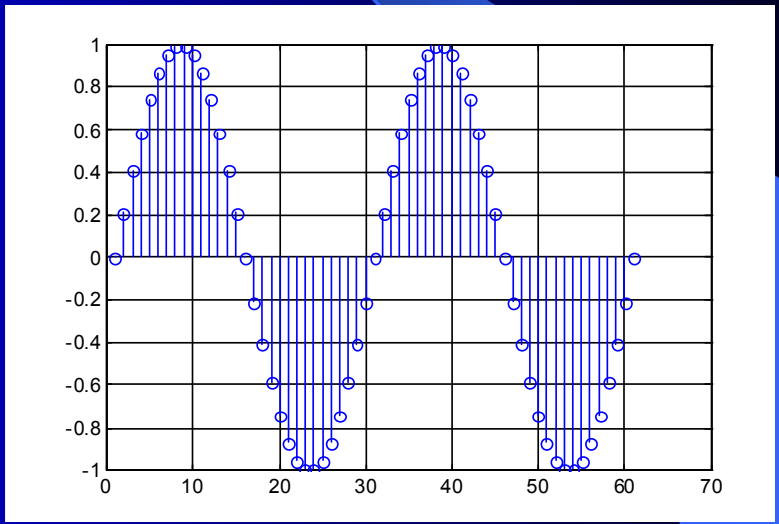
Muestreo y conceptos básicos.

- Frecuencia de muestreo (Fs)
- Frecuencia discreta f (f < 0.5)

$t = n.Ts \Rightarrow T = N.Ts = N / Fs$

$N = Fs.T = Fs / F$

$1 / N = f = F / Fs$



Como Fs > 2.F, como máximo f = F / 2.F = 0.5

- Filtro antialiasing: En forma digital o analógica.

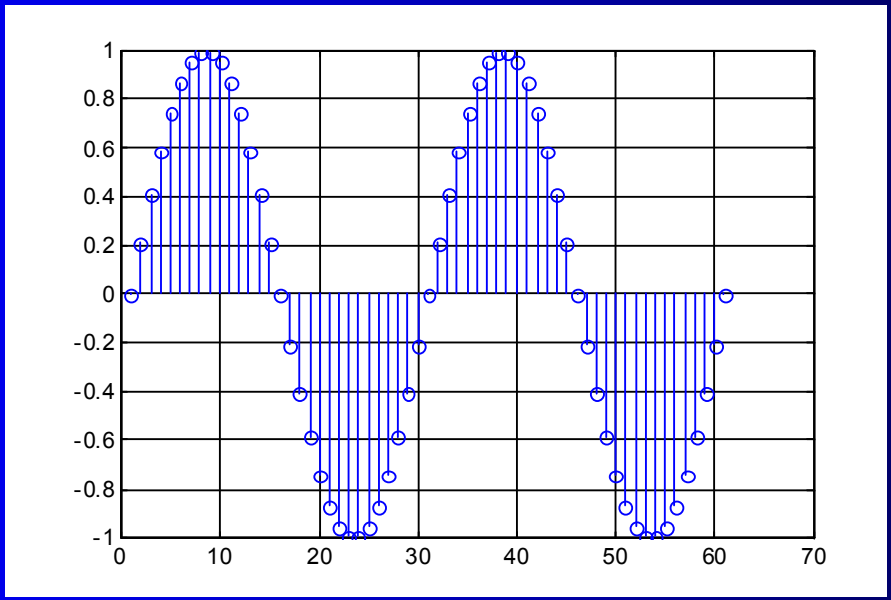
Muestreo y conceptos básicos. Actividades en MATLAB.

- Actividad 1.1: Muestreo
- Actividad 1.2: Cálculo de frecuencia discreta
- Actividad 1.3: Periodicidad y de No Periodicidad en relación de F y Fs.
- Actividad 1.4: Verificación tasa de Nyquist.

Actividad 1.1: Muestreo.

Se tiene una señal senoidal de frecuencia 1Hz, y se la muestrea con Fs=30 Hz. Ver la señal en forma continua y discretizada. Comprobar que t=n*Ts (tener en cuenta que el primer valor del vector de la señal discreta es 1, no 0).

```
» syms t;f=sin(2*pi*1*t);
» subplot(2,1,1),ezplot(f,[0 2]),grid;
» fs=30;t=0:1/fs:2-1/fs;f=sin(2*pi*1*t);
» subplot(2,1,2),stem(f),axis([1 60 -1 1]),grid;
```



VOLVER

Convolución.

• En tiempo continuo se podía modelizar cualquier sistema con la ayuda de la transformada de Laplace. Es decir se planteaban las ecuaciones integro-diferenciales que representaban al sistema y luego se realizaba la transformada de Laplace. La función transferencia se definía como la relación, en transformada de Laplace, de la salida deseada sobre la entrada. Una vez obtenida esta fc transferencia o H(s) se podían realizar diversos estudios de estabilidad del sistema, etc. Al antitransformar H(s) obteníamos h(t), que por definición si se convolucionaba con la entrada resultaba la salida y(t). Aquí había un divorcio entre la aplicación práctica y la teoría ya que en el circuito no se podía distinguir claramente que se hacía la convolución.

• Esto cambia en forma digital ya que para implementar el sistema se hace realmente la convolución, como fórmula matemática. Por lo tanto el problema se resume en encontrar los coeficientes para convolucionar con la entrada.

Convolución.

• Definición de convolución:

$$y(n) = \sum_{k=0}^{tama(h)-1} h(k) * x(n-k)$$

• Teniendo los coeficientes h(n) se puede hacer la convolución con la entrada y se obtiene la salida.

• Si de la transformada de Laplace H(s) se obtiene h(t), y se la digitaliza se tiene h(n), por lo que ya obtuve los coeficientes para hacer la convolución anterior. Esta es una implementación directa FIR, y el problema es la cantidad de coeficientes necesarios para la modelización.

• La implementación IIR de un sistema se resuelve mediante la ecuación en diferencias:

$$y(n) = -\sum_{k=1}^{N} a(k) * y(n-k) + \sum_{k=0}^{M} bk * x(n-k)$$

• Para encontrar estos coeficientes una vez obtenida H(s) se aplica la transformación bilineal para llegar a H(z), es decir la transformada Z del sistema. En el numerador tendremos los b(k), mientras que en el denominador estarán los a(k).

Convolución.

- $y(t) = x(t) * h(t)$. Sin implementación práctica.
- $y(n) = x(n) * h(n)$:
  - Implementación con Sumatorias (ciclo for) y productorias => Orígen unidad MAC + SHIFTER + ALU + Unidades en paralelo.
  - Implementación en forma recursiva y no recursiva: Filtros FIR e IIR.
  - Obteniendo el vector h(n) se implementa cualquier sistema.

Convolución. Actividades en MATLAB.

- Ver Programa de Convolución PC522.
- Actividad 1.5: Implementación de un sistema mediante h(n).
- Actividad 1.6: Implementación con lazos for.
- Actividad 1.7: Convolución x(n) y h(n).

Actividad 1.5.

- Actividad 1.5

Definir un sistema dinámico y(n) = 4*x(n-1) – 3*x(n) y obtener probando los valores h(n) para definir este sistema como una convolución.

```
» n=0:10;
» x=n*10/2;
» n=2:11;f=4*x(n-1)-3*x(n)

f =

   -15   -10    -5     0     5    10    15    20    25    30

» h=[-3,4];
» conv(h,x)

ans =

     0   -15   -10    -5     0     5    10    15    20    25
30   200
```

Actividad 1.6.

- Actividad 1.6

Realizar la convolución anterior con lazos for, que es tal como lo haríamos al programar en un PIC, DSP, etc.

```
» for n=1:11,
        suma(n)=0;
        for k=1:length(h),
        if (n>k) suma(n)=suma(n)+x(n-k)*h(k);
        end,
        end,
end;
» suma

suma =

     0     0   -15   -10    -5     0     5    10    15    20
25
```

VOLVER

Sistemas FIR e IIR.

Sistemas FIR e IIR. Implementación en forma recursiva y no recursiva

Un sistema FIR (finite impulse response) es un sistema con respuesta impulsional de duración finita, es decir h(n) tiene una cantidad finita de valores, contrariamente de lo que sucede con los sistemas IIR (infinite impulse response). Para obtener la salida y(n), teniendo la entrada x(n) y h(n) se realiza la convolución, tal como sucedía en el tiempo continuo. Para los distintos sistemas se tiene este producto (de convolución).

$h(n)=0$ para todo $n>M+1$ v $n<0$, para los sistemas FIR.

De la definición del producto de convolución se tiene que la sumatoria sería posible sólo en los sistemas FIR, no así en los IIR por ser una sumatoria de infinitos valores. Para remediar esto se implementan estos sistemas en forma recursiva. Los sistemas FIR debe tener una memoria finita, mientras que los IIR tendrían una memoria infinita si se trata en forma no recursiva.

Actividad 1.8: Sistemas FIR e IIR.

Actividad 1.8

Implementar en forma recursiva y no recursiva el sistema y(n) = 1/(n+1) * sum(x(k), k=0:n).
(En forma recursiva y(n)=n/(n+1)*y(n-1)+1/(n+1)*x(n))

```
» clear all
»
»
» Fs=20;t=1/Fs:1/Fs:1;x(t*Fs)=sin(2*pi*1*t); %19 valores

Warning: Subscript indices must be integer values.
» subplot(3,1,1),stem(x);
» for n=1:19,
        m=0;
        for k=1:n,
                m=m+x(k);
        end;
        y(n)=m/n;
end;
» subplot(3,1,2),stem(y);
```

Actividad 1.8: Sistemas FIR e IIR.

Actividad 1.8

Implementación del sistema en forma recursiva:

```
» for n=0:18,
        if (n ~=0)
                yy(n+1)=yy(n)*n/(n+1)+x(n+1)/(n+1); %aparece
                                                % +1 porque Matlab
        else        %       toma como primera muestra a la x(1)
                yy(n+1)=x(n+1)/(n+1);
        end;
end;
» subplot(3,1,3),stem(yy);
```

Se puede ver que el resultado es el mismo. La ventaja de implementación en forma recursiva es la rapidez, ya que me ahorro un ciclo for. La desventaja es la memoria que pudiera necesitar.

VOLVER

Teoría básica PDS. Actividades en MATLAB.

- Actividad 1.10: DFT de una señal discreta.
- Actividad 1.11: Componente frecuencial de una sucesión periódica.
- Actividad 1.13: Filtrar una señal con implementación FIR con 1000 valores de h(n).
- Actividad 1.14: Idem anterior con 20 valores de h(n).
- Actividad 1.15: Logro de un mejor filtro aumentando F.

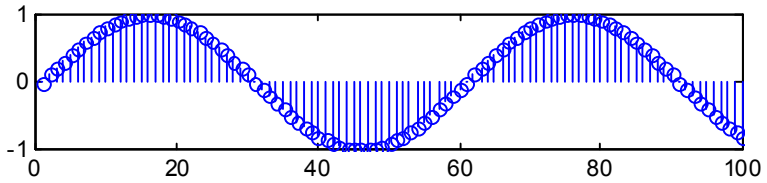
Actividades sacadas del apunte “Introducción a MATLAB”

Actividad 1.10.

- Actividad 1.10

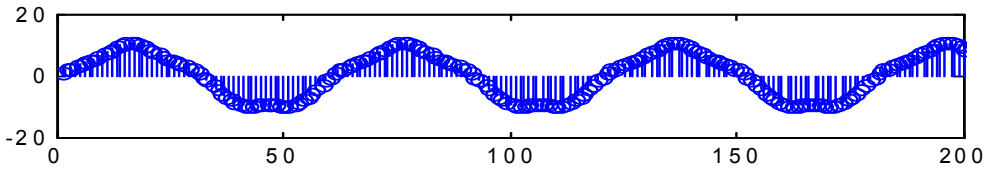
Obtener la transformada discreta de Fourier de una señal discreta. Graficar la señal y los espectros de amplitud y fase.

```
» Fs=300;t=0:1/Fs:1;f=sin(2*pi*5*t);
» fw=fft(f);
» afw=abs(fw);
» ffw=angle(fw);
» ffwgrados=ffw*180/pi;
» subplot(3,1,1),stem(f(1:100));
» subplot(3,1,2),stem(afw/length(afw));
» subplot(3,1,3),stem(ffwgrados);
```



Actividad 1.10.

```
» Fs=300;t=0:1/Fs:1;f=10*sin(2*pi*5*t)+cos(2*pi*20*t);
» fw=fft(f);
» afw=abs(fw);
» ffw=angle(fw);
» ffwgrados=ffw*180/pi;
» subplot(3,1,1),stem(f(1:200));
» subplot(3,1,2),stem(afw/length(afw));
» subplot(3,1,3),stem(ffwgrados);
```



Actividad 1.10.

Lectura del eje de las frecuencias:

- Primero hay que decir que estamos frente a frecuencias discretas (F/Fs). La frecuencia discreta se definía como f=F/Fs=1/N, es decir la inversa de la cantidad de muestras que entran en un ciclo.
- Si tengo un ciclo la cantidad de muestras que entran en un ciclo serán N. Ahora si tengo P ciclos de la señal, entonces la cantidad de muestras que me entrarán serán P*N, que es la cantidad de muestras totales.
- Por lo tanto f= 1/N * Pciclos/Pciclos = Pciclos / cant. de muestras totales.
- Resumiendo tengo que dividir el valor que está en Matlab por la cantidad de muestras para saber la frecuencia discreta y multiplicar por Fs si necesito conocer la frecuencia continua. La segunda mitad de las gráficas de los espectros “está de más” ya que me muestra el contenido frecuencial para f>1/2, algo que en la realidad no lo tengo por la tasa de Nyquist (Fs>=2F => f<=1/2).

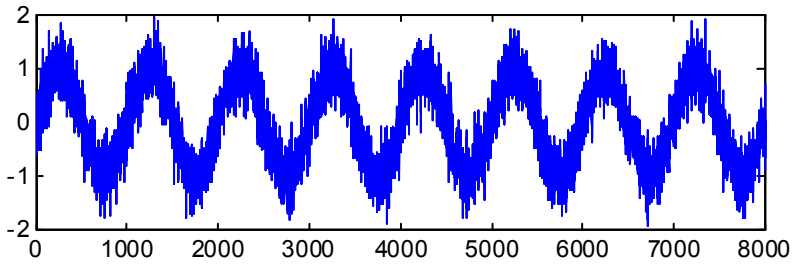
Actividad 1.13.

Actividad 1.13

Se tiene una señal senoidal de 1 Hz con ruido aleatorio, Fs = 1000 Hz. Dibujar la señal y su espectro de amplitud. Filtrar el ruido lo mejor posible sin usar técnicas de diseño de filtros.

```
» clear all;
» Fs=1000;t=0:1/Fs:10-1/Fs;
» x=sin(2*pi*1*t);
» y=x+0.3*randn(size(t));
» n=1:length(yw);filtrow(n)=0;
» n=5:12;filtrow(n)=1;   % f=0.001. 10000 muestras => en 10
                          %(10/10000=0.001) tengo la componente
» filtrow(10000-n)=1;
» filtron=real(ifft(filtrow));
» senalf=conv(filtron,y);
» subplot(2,1,1),plot(y(1:8000));
» subplot(2,1,2),plot(senalf(1:8000));
```

Actividad 1.13.



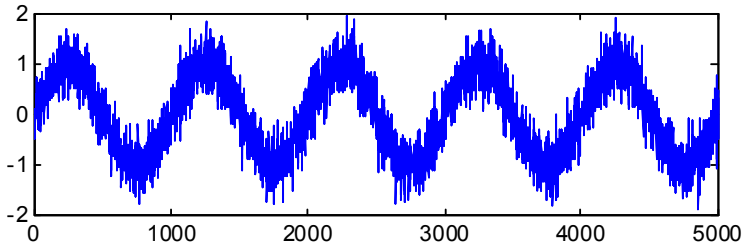
• Este filtrado es efectivo pero el problema es que h(n) posee muchos valores (10000). Por esto hay técnicas para el diseño de filtros más efectivas. Este filtro así diseñado tiene una implementación no recursiva y es FIR.

Actividad 1.14.

Actividad 1.14

Realizar el mismo filtro anterior, sólo que h(n) tenga 20 valores, y no 10000. Sacar conclusiones porque el filtrado no es tan bueno. Implementar con lazos for que es cómo se implementaría en un sistema microprocesado.

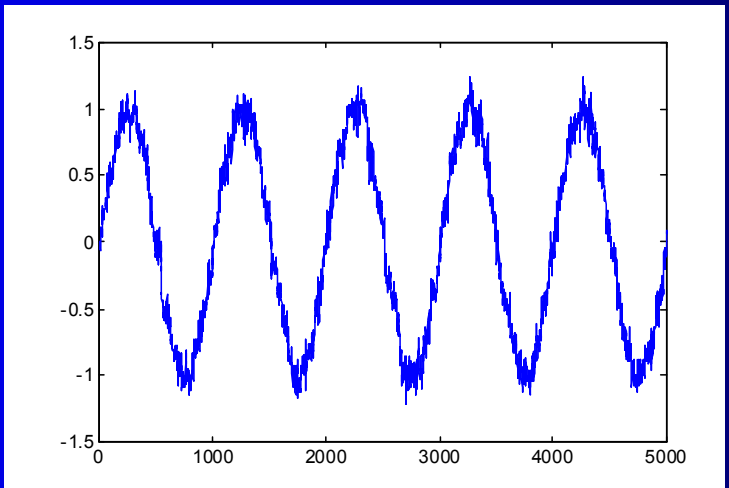
```
» n=1:20;filtro(n)=0;
» filtro(1)=1;filtro(20)=1;
» filtron=real(ifft(filtro));
» senalf=conv(filtron,y);
» subplot(2,1,1),plot(y(1:5000));
» subplot(2,1,2),plot(senalf(1:5000));
```



Actividad 1.14.

La señal filtrada no es tan buena porque al tener 20 valores la frecuencia discreta más chica que puede tener es 1/20 = 0.05. La frecuencia discreta de mi señal es F/Fs = 0.001. La cantidad mínima de valores de h(n) para tener un filtrado “perfecto” sería 1000 => 1/1000=0.001.
Al hacer que filtro(1) = 1 entonces dejamos pasar todas las frecuencias menores a 1/20 = 0.05.

```
» for n=1:length(y),
        suma(n)=0;
        for k=1:length(filtron),
                if (n-k)>0 suma(n)=suma(n)+filtron(k)*y(n-k);
                end;
        end;
        end;
» figure(2),plot(suma(1:5000));
```



Teoría básica PDS. Actividades en MATLAB.

- Ejemplo 4: Obtener coeficientes FIR e IIR para modelizar retardo.
- Ejemplo 5: Implementación IIR

Actividades sacadas del apunte “Convolución y transformada Z usando MATLAB”

Ejemplo 1.

- **Ejemplo 1.**

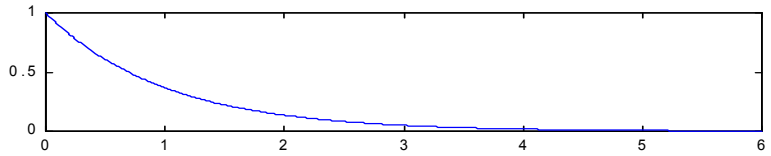
Para un circuito RC serie, hallar Vc para entradas escalón, impulso, y senoidal utilizando convolución (**implementación FIR**) y las herramientas de Matlab para hallar tales respuestas.

$$\frac{Vc(s)}{Vi(s)} = \frac{1}{RCs+1}$$

```
%Ejemplos de convolucion en circuitos
%ejemplo 1: cálculo de la tensión en el C en un circuito RC con una
tensión de
%entrada escalón.
clear all;
syms t s;
r=1;c=1;
%Calculando la respuesta al impulso, escalón y senoidal según
%funciones de MATLAB:
num=[1];
den=[r*c,1];
sistema=tf(num,den)
subplot(311),impulse(sistema),ylabel('Respuesta al impulso');
subplot(312),step(sistema),ylabel('Respuesta al escalón');
t=0:0.01:6;vis=sin(t);
subplot(313),lsim(sistema,vis,t),ylabel('Respuesta al seno');
```

Ejemplo 1.

```
%Calculando h(t), digitalizando y convolucionando:
ilaplace(1/(r*c*s+1))
h=c/r*exp(-t*c/r);
%impulso;
vi(1:length(t))=0;vi(1)=1;
imp=conv(vi,h);
figure(2);
subplot(311),plot(t,imp(1:length(t))),ylabel('Conv. Resp al impulso');
%escalón;
vi(1:length(t))=1;
ste=conv(vi,h);
subplot(312),plot(t,ste(1:length(t))),ylabel('Conv. Resp al escalón');
%Seno
sen=conv(vis,h);
subplot(313),plot(t,sen(1:length(t))),ylabel('Conv. Resp al seno');
```



Conclusiones Ejemplo 1.

Aquí se ve una diferencia en las amplitudes y es debido a que, luego de realizar la convolución de las señales, se debe multiplicar este resultado por Ts, o dividir por la frecuencia de muestreo:

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

Discretizando y analizando en una muestra (en un instante):

Si $k * \Delta t = \tau$ y $n * \Delta t = t$ =>

$$y(k*\Delta t) = \int_{k*\Delta t}^{(k+1)*\Delta t} h(k*\Delta t) * x((n-k)*\Delta t)$$

En ese intervalo los h y x son constantes, entonces para calcular la integral definida queda:

$$y(n) = \Delta t * \sum_{k=0}^{M-1} h(k*\Delta t) * x((n-k)*\Delta t) = Ts * \sum_{k=0}^{M-1} h(k) * x(n-k)$$

Es decir se debe multiplicar al resultado de la convolución con Ts (1/Fs) y también normalizar los coeficientes h(n).

Ejemplo 3.

%Ejemplo 3.

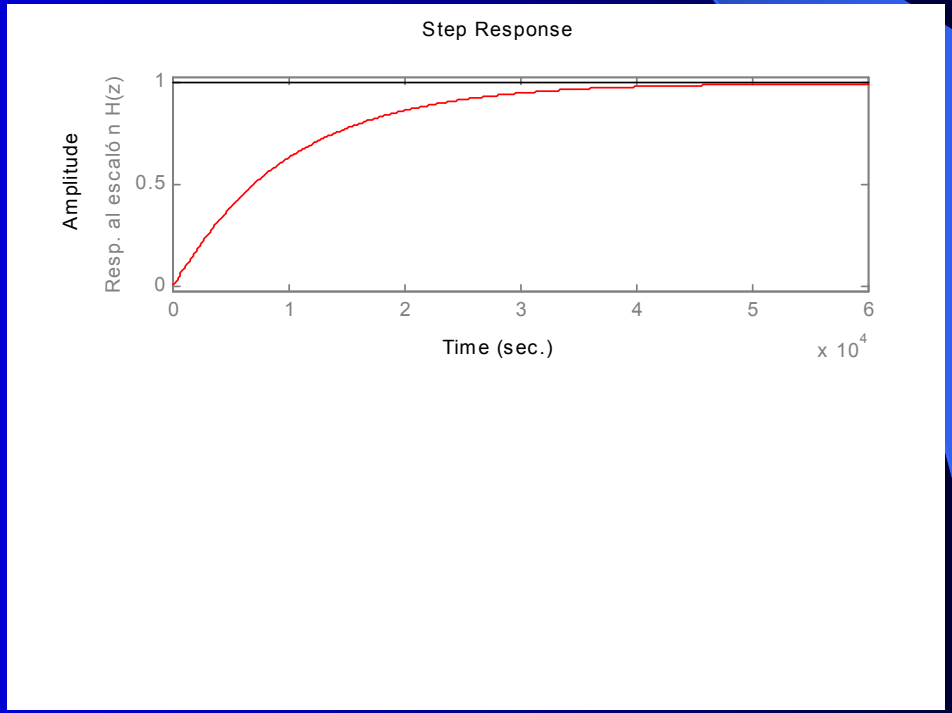
Teniendo la función transferencia H(s) = I(s)/Vi(s) de un circuito RC serie obtener su símil en transformada Z utilizando la transformada bilineal. Comparar la respuesta al impulso de H(s) y H(z). De H(z) obtener los factores a(k) y b(k) (**implementación IIR**) y convolucionar con un escalón. Comparar esta respuesta con el ejemplo 1.

```
%Transformada Bilineal
%Vamos a usar la función transferencia de un filtro pasa alto, es decir
s/s+1
clear all;
r=1;c=1;fs=100;
n=[1];
d=[r*c 1];
sis=tf(n,d)
figure(1),bode(sis);
[nu de]=bilinear(n,d,fs);
[h w]=freqz(nu,de,256);              %resp en frecuencia del sistema en Z

figure(2),plot(w,abs(h));
sisz=tf(nu,de,fs)
figure(3);
subplot(311),plot(abs(h)),ylabel('Resp en frec.');
subplot(312),impulse(sis),ylabel('Resp al impulso H(s)');
subplot(313),impulse(sisz),ylabel('Resp. al imp. H(z)');
```

Ejemplo 3.

```
%Con los coeficientes del numerador y denominador tengo los
coeficientes
%a(k) y b(k), por lo que puedo probar convolucionar esto con una
%señal de entrada.
t=0:1/fs:6;
vi(1:length(t))=1;
figure(4);
subplot(211),step(sisz),ylabel('Resp. al escalón H(z)');
subplot(212),plot((filter(nu,de,vi))),ylabel('Resp. al escalón conv');
%Respuesta al escalón convolucionando.
```

```
Transfer function:
  1
-----
s + 1


Transfer function:
0.004975 z + 0.004975
---------------------
      z - 0.99

Sampling time: 100
```



Ejemplo 4.

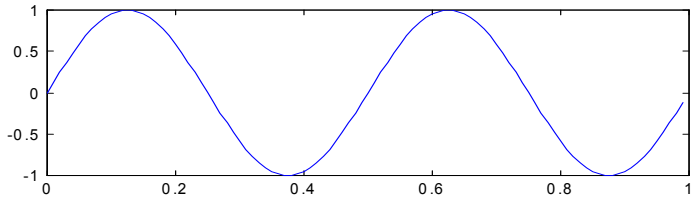
Ejemplo 4.

Obtener los coeficientes FIR e IIR para modelizar retardo:

$$y(t) = x(t - t0) \qquad Y(s) = X(s) * e^{s*t0}$$

La transformada inversa de Laplace de exp(s*t0) es un impulso unitario en t0, por lo tanto h(n) sera un vector con todos ceros, excepto en el valor K (K*Ts segundos) donde quiero que se retrase.

```
fs=100,
h(1:40)=0;h(41)=1;  %Retardo de 40 muestras * 0.01 seg = 0.4 seg.
t=0:1/fs:1-1/fs;
vi=sin(2*pi*2*t);
salida=conv(vi,h);
subplot(211),plot(t,vi),ylabel('Entrada');
subplot(212),plot(t,salida(1:length(vi))),ylabel('Entrada retardada');
```



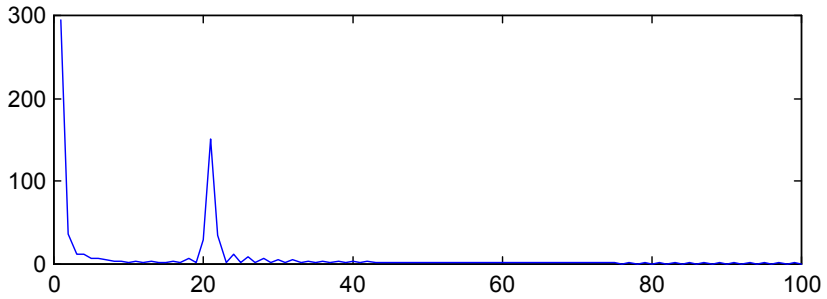
Ejemplo 5.

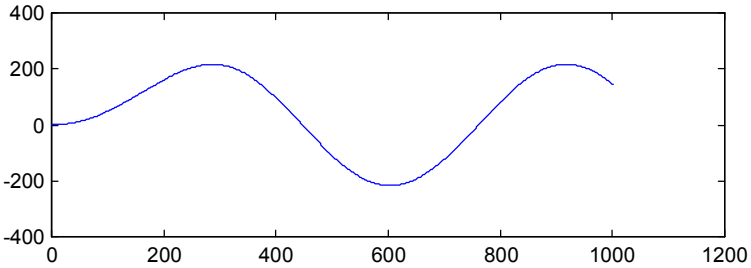
Ejemplo 5.

Obtener coeficientes para una implementación IIR a partir de los coeficientes de una implementación FIR. Suponer que H(s) es tal que h(t)= exp(-t)+sin(t)+cos(2*pi*10*t). Comparar las respuestas en frecuencia H(z) con los coeficientes FIR e IIR.

```
tt=0:1/100:3;
h=exp(-tt)+sin(tt)+cos(2*pi*10*tt);
t=0:1/100:10;
[hh,ww]=freqz(h,1,100);
figure(1),subplot(211),plot(abs(hh));
%Variar los polos y ceros y ver como varia la resp en frec con la
original.
[biir,aiir]=invfreqz(hh,ww,6,6)
[hhiir,wwiir]=freqz(biir,aiir,100);
subplot(212),plot(abs(hhiir))
vi=sin(t);
figure(2),subplot(211),plot(filter(h,1,vi))
sis=tf(biir,aiir);
figure(2),subplot(212),plot(filter(biir,aiir,vi))
```

Ejemplo 5.





VOLVER

Implementación FIR e IIR con coeficientes calculados con MATLAB.

FIR (función y filtro tipo remez)

$$y(n) = \sum_{k=0}^{M-1} b(k) * x(n-k)$$

$$H(z) = \sum_{k=0}^{M-1} b(k) * z^{-k}$$

IIR (función y filtro tipo yulewalk)

$$y(n) = -\sum_{k=1}^{N} a(k) * y(n-k) + \sum_{k=0}^{M} bk * x(n-k)$$

$$H(z) = \frac{\sum_{k=0}^{M} b(k) * z^{-k}}{1 + \sum_{k=1}^{N} a(k) * z^{-k}}$$

Ejemplo 6.

Ejemplo 6.

Diseño de filtros digitales dando como parámetros la amplitud para las diferentes frecuencias normalizadas (1=Fs/2). Calcular los coeficientes de un filtro FIR y los de un IIR si Fs=100 Hz, fc1= 16.66 Hz y fc2 = 23.33 Hz. Probar con una señal de entrada utilizando este filtro.

```
%Cálculo de filtros digitales
clear all;
fs=100;
f=0:1/15:1; %f=0:fs/(2*15):fs/2 Hz.
m=[0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0];%16.66Hz=5/15*50Hz
                                    %23.33Hz=7/15*50Hz
%El tamaño de m, y por lo tanto f dependerá de la resolución requerida
%para el filtro.
%Definición de la señal de entrada.
t=0:1/fs:3;
senal=sin(2*pi*16*t)+sin(2*pi*2*t);
figure(1),plot(t,senal);

%Diseño del filtro IIR (40 coeficientes 20 a y 20 b)

[b,a]=yulewalk(20,f,m);
[H,wt]=freqz(b,a,150);
figure(2),subplot(211),plot(wt/pi*fs/2,abs(H)),ylabel('Implementacion IIR');
subplot(212),plot(t,filter(b,a,senal));
```

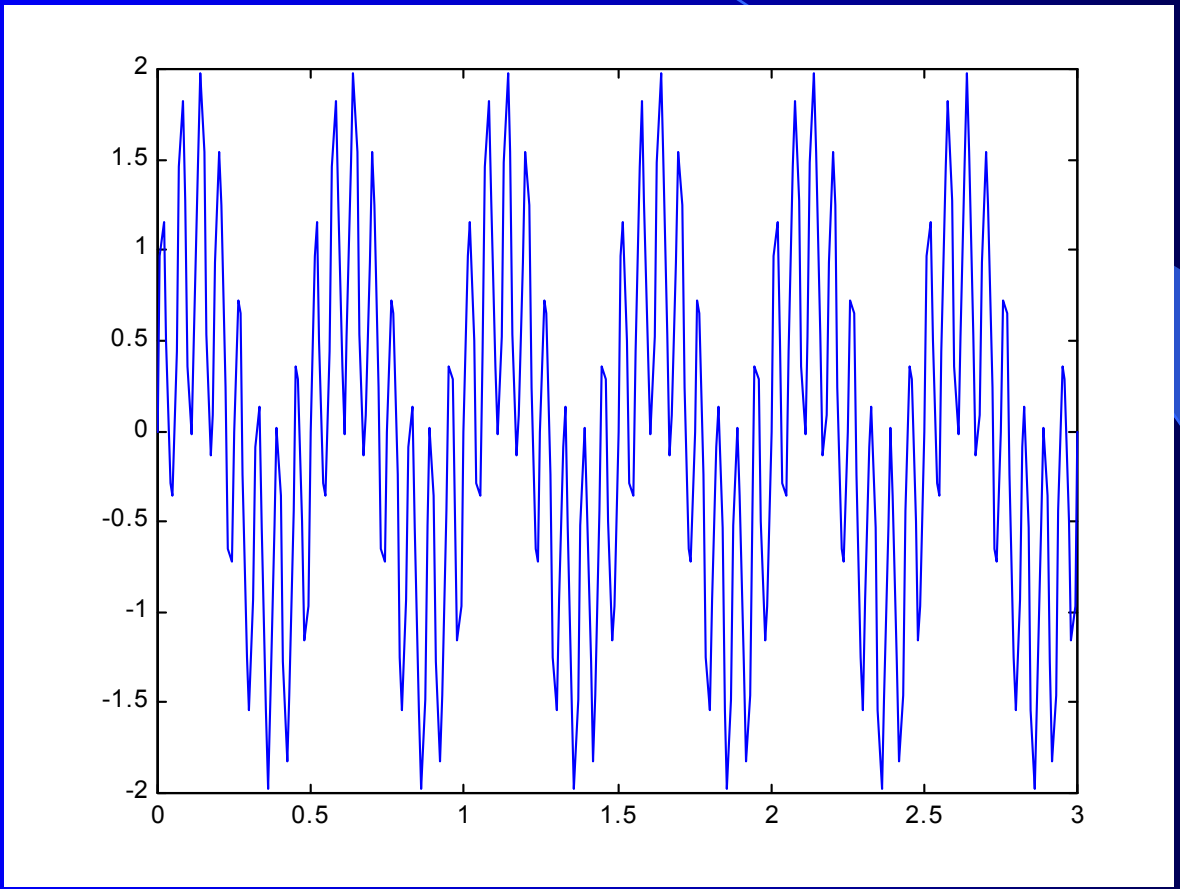
Ejemplo 6.

Ejemplo 6.

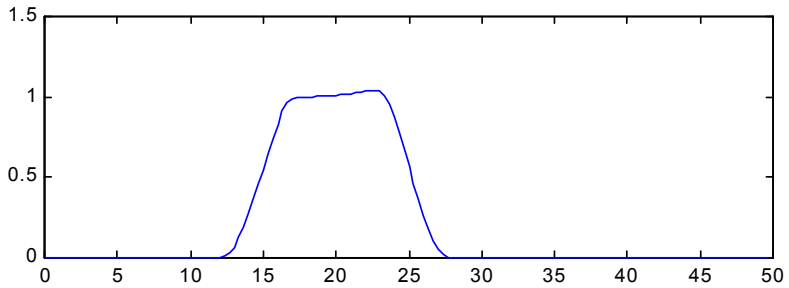
Diseño de filtros digitales dando como parámetros la amplitud para las diferentes frecuencias normalizadas (1=Fs/2). Calcular los coeficientes de un filtro FIR y los de un IIR si Fs=100 Hz, fc1= 16.66 Hz y fc2 = 23.33 Hz. Probar con una señal de entrada utilizando este filtro.

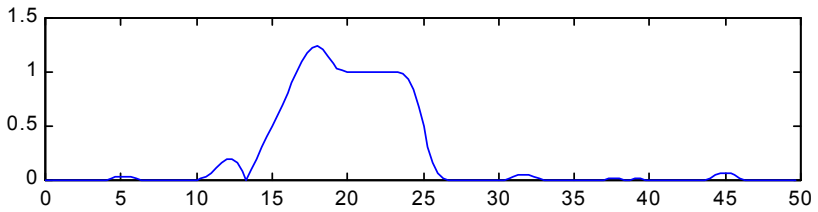
```
%Diseño del filtro FIR (100 coeficientes)

clear H wt;
h=remez(100,f,m);
[H,wt]=freqz(h,1,150);
figure(3),subplot(311),plot(wt/pi*fs/2,abs(H)),ylabel('Implementacion FIR');
%Para aplicar el filtro anterior definido por b y a a x se hace:
%Con convolución
res=conv(h,senal);
subplot(312),plot(t,res(1:length(t)));
%Con filter
subplot(313),plot(t,filter(h,1,senal));
```

Ejemplo 6.



Ejemplo 6.





VOLVER

Programación en el kit SHARC.

```
/************************************************************/
/*
/************************************************************/

/* ADSP-2106x System Register bit definitions */
#include <def21060.h>
#include <21060.h>
#include <signal.h>
#include <sport.h>
#include <macros.h>
#include <math.h>
#include <filters.h>
#include <stdio.h>
#include <conio.h>   //para el kbhit()
                                                    /* DMA Chain pointer bit definitions */
#define CP_PCI 0x20000          /* Program-Controlled Interrupts bit */
#define CP_MAF 0x1ffff          /* Valid memory address field bits   */

#define SetIOP(addr, val)  (* (int *) addr) = (val)
#define GetIOP(addr)       (* (int *) addr)

/************************************************************/

#define NUM_TAPS 1000

float pm coeffs[NUM_TAPS] =
{
#include "fir.h"
};

float dm state[NUM_TAPS+1];
int dm opcion;    //elijo entre ruido o codec
/************************************************************/

#
```

Programación en el kit SHARC.

```
/*********************************************************/
/*                                                       */
/* Serial port receive DMA complete                      */
/*                                                       */
/*********************************************************/
void spr0_asserted( int sig_num )
{
    float filter_input;
    if (opcion==1)    //si la opcion es 1, entonces la entrada es el codec, sino es ruido
                    filter_input = rx_buf[1];
    else
                    filter_input = rand() & 0x00003fff; //aca está la entrada de ruido
    if (opcion==3)
            tx_buf[1]=rx_buf[1];
    else
            // Filter sample and output.
            tx_buf[1] = fir( filter_input, &coeffs[0], &state[0], (int)NUM_TAPS );
     tx_buf[2] = tx_buf[1];
    //printf("Valor: %f: ",filter_input);
    //fid=fopen("senal.txt","a+");
    //fprintf(fid,"%f",tx_buf[1]);
    //fclose(fid);

}
```

Programación en el kit SHARC.

```
/*********************************************************/
void main ( void )
{
   int i;
   int x;


   puts("\n Bienvenido al programa de prueba de SHARC");
   puts("\n ---------- -- -------- -- ------ -- -----");
   puts("\n \t Elija opción: ");
   scanf("%i",&opcion);
   //printf("\n Opción elegida: %i",opcion);
    // Initialize state array for FIR filter.
   for( i=0 ; i<NUM_TAPS+1 ; i++ )
      state[i] = 0.0;

   // Initialize some SHARC registers.
   init_21k();

   // Reset the Codec.
   set_flag( SET_FLAG0, CLR_FLAG );    /* Put CODEC into RESET  */
   for( x=0 ; x<0xffff ; x++ )         /* Hold CODEC in RESET */
      ;
   set_flag( SET_FLAG0, SET_FLAG );    /* Release CODEC from RESET */

   // Configure SHARC serial port.
   setup_sports();

   // Send setup commands to CODEC.
   send_1847_config_cmds();
```

Programación en el kit SHARC.

```
    // Turn on the timer.
    timer_on();


    // Loop forever.
    for(;;)
    {
      if (kbhit() != 0)      //en cualquier momento puedo interrumpit presionando una tecla
        {
        puts("\n Ingrese nuevamente una opcion: ");
        scanf("%i",&opcion);
        }
      idle();
    };
}
/*********************************************************/

// End of file bp.c .

/*********************************************************/
```

Compilación

```
g21k -I..\include -o filtro.21k filtro.c ..\libdh.a
```

Bajarlo al DSP desde DOS

```
set adsp0=0,23,5,1
..\dh21k -b0 filtro.21k
```

VOLVER