Maestría en Ingeniería Eléctrica especialización Telecomunicaciones

Comunicaciones Digitales

Tarea #0

Luis Emilio Tonix Gleason

Fernando Alberto Madera Torres

20/03/2022

Dr. Ramon Michel Parra

Tabla de contenido

Ejercicio 1

Relación entre las series de Fourier (FS), la transformada de Fourier (FT), La transformada de Fourier en tiempo discreto (FTDT) y la transformada discreta de Fourier (FDT).

Series de Fourier

Cualquier forma de onda periódica está formada por un componente promedio y una serie de ondas senoidales y cosenoidales relacionadas armónicamente, donde una armónica es un múltiplo entero de la frecuencia fundamental.

$$f(t) = A_0 + A_1 \cos\theta + A_2 \cos 2\theta + A_1 \cos 3\theta \ldots + A_n \cos n\theta + B_1 \sin\delta + B_2 \sin 2\delta + B_3 \sin 3\delta \ldots + B_n \sin n\delta \qquad donde \quad \delta = \theta$$

$$x(t) = \sum_{k=-\infty}^{\infty} c_k \, e^{jk\omega_0 t}$$

$$c_k = \frac{1}{T_0}\int_T x(t) e^{-jk\omega_0 t}\, dt$$

La frecuencia fundamental es la primera armónica y es igual a la frecuencia o rapidez de repetición de la forma de onda. La frecuencia fundamental es la mínima necesaria para representar la forma de onda.

La serie de Fourier se utiliza en el análisis de señales para cambiar una señal en el dominio del tiempo a una señal en el dominio de la frecuencia, se puede obtener una serie de Fourier para cualquier función periódica.

Transformada de Fourier

Puede ser aplicada a una señal aperiódica, este limitando un tiempo de observación y dando por hecho que esa ventana observable sea tomada como un periodo.

$$F(\omega) = \int_{-\infty}^{\infty} x(t) e^{-j\omega t}\, dt$$

$$X(t) = X^{-1}(\omega) = \frac{1}{2\pi}\int_{-\infty}^{\infty} F(\omega) e^{j\omega t}\, d$$

Transformada de Fourier en Tiempo Discreto

Es la transformada para una señal discreta, tiene relación con la transformada z, la transformada de tiempo discreto vive en el circulo unitario de la transformada Z.

$$X(\omega) = \sum x[n] e^{-j\omega n}$$

Transformada Discreta de Fourier

Se muestrea una señal en el dominio del tiempo, en tiempo discretos. Las muestras se guardan en una memoria para realizar un algoritmo que calcula la transformación.

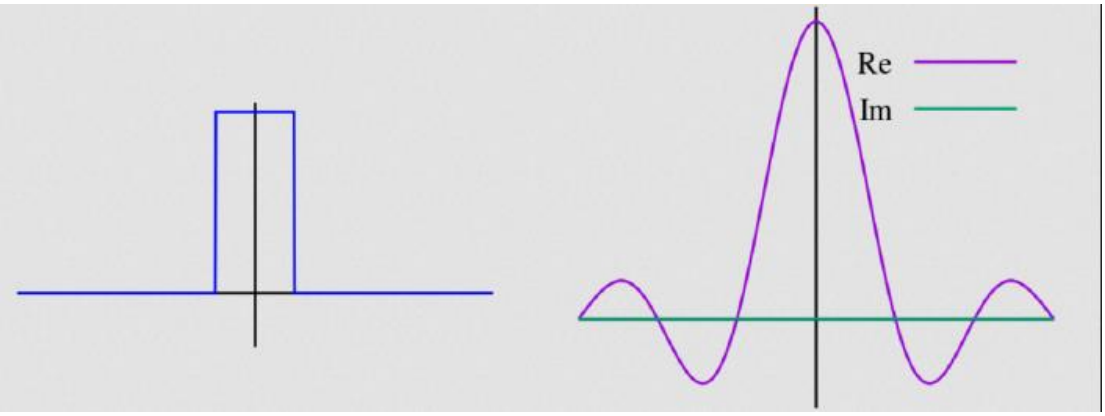
$$X_k = \sum x_n \, e^{-j\frac{2\pi}{N}kn}$$

Ejercicio 2

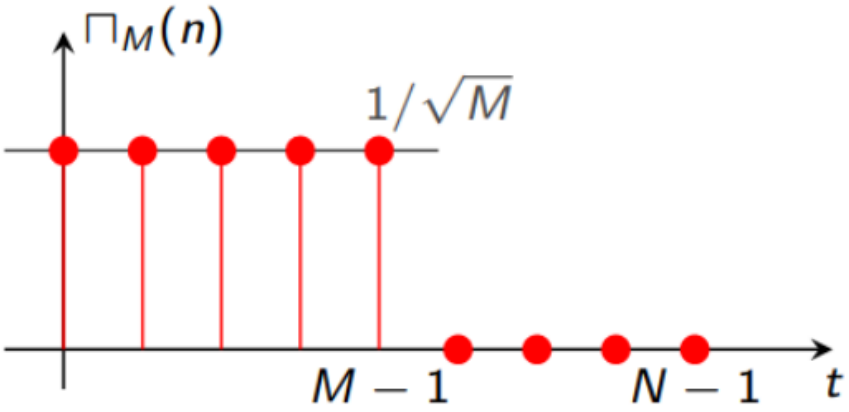
Dos funciones temporales (No senoidales) y obtener su TF mediante el cálculo directo y compruebe su aproximación mediante la TDF correctamente escalada.

$$F(\omega) = \int_{-1}^{1} e^{-j\omega t}\, dt = -\frac{e^{-j\omega t}}{j\omega} = \frac{1}{j\omega} e^{-j\omega t|_{-1}^{1}} =$$

$$F(\omega) = -\frac{1}{j\omega}\left[e^{-j\omega} - e^{j\omega}\right] = \frac{1}{\omega}\left[\frac{e^{j\omega} - e^{-j\omega}}{j}\right] = \frac{sin(\omega)}{\omega} = sinc(\omega)$$



TDF



La norma de este pulso es $\frac{1}{\sqrt{M}}$ este sería el escalamiento según su longitud

$$X(k) = \frac{1}{\sqrt{N}} \sum \frac{1}{\sqrt{M}} e^{-j\frac{2\pi}{N}kn} = \frac{1}{\sqrt{MN}} \sum_{n=0}^{M-1} e^{-j\frac{2\pi}{N}kn}$$

Ejercicio 3

Diseñar un filtro pasa altas de 51 coeficientes a partir de 8pi/10

Filtro en la frecuencia de $-\pi$ a π con el filtro hasta $\omega_f > \frac{8\pi}{10}$

Generacion de ventana cuadrada

Se define un tiempo de observacion de 625 segundos por ser $-\pi, \pi$ el rango es 2π

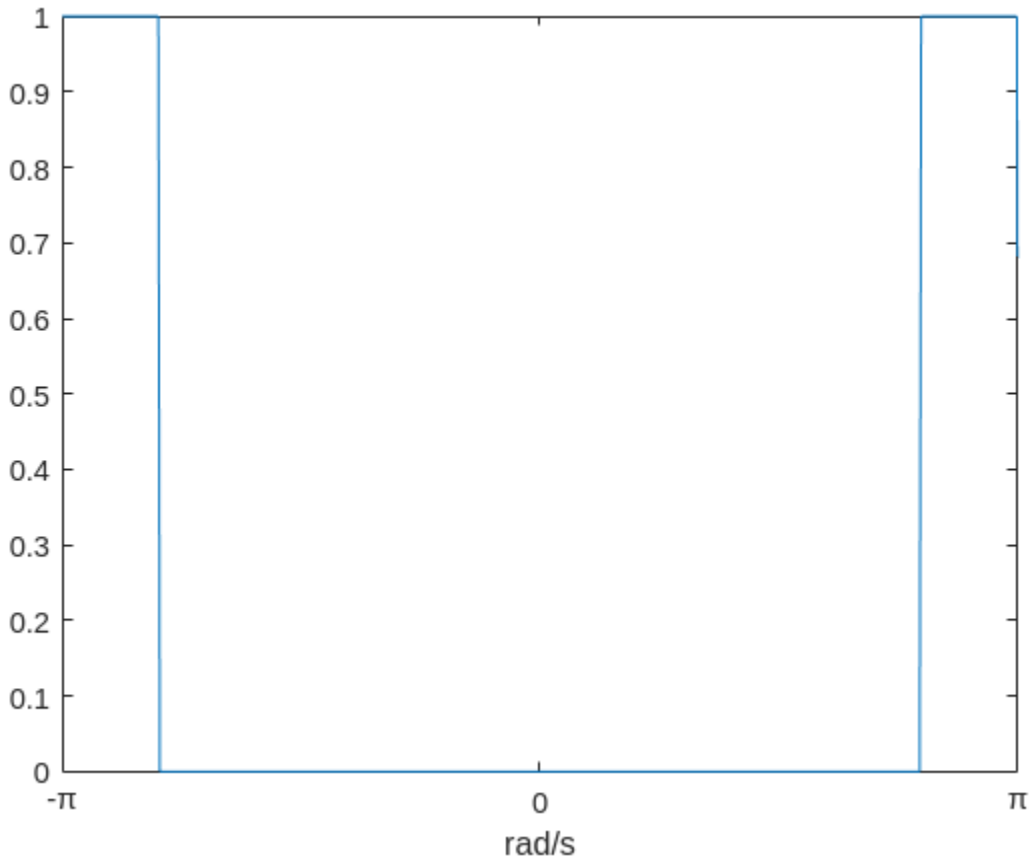
```
bound =[-pi,pi]; % 2pi range lenght
obs = 100 % 100 segundos
```

```
obs = 100
```

$[-\pi, -8\pi/10,\ 8\pi/10, \pi]$

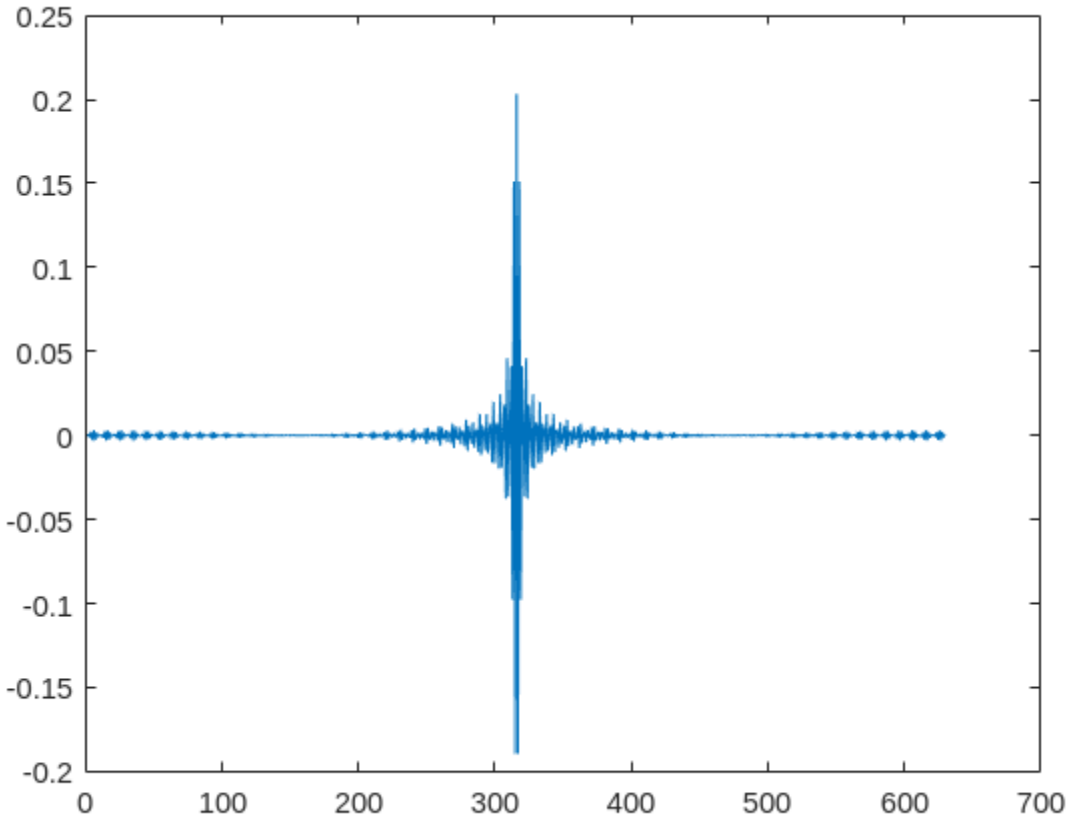
```
%Puntos de corte del filtro
cut_points =  [-pi -8*pi/10 8*pi/10 pi];


% tiempo de observacion 100s * 2pi = 625 muestras
space = bound(1):1/obs:bound(2)+1/100;
%inicia señal de filtro en 0
rad_space = zeros(1,length(space));


%parametros limite en la señal
%cut_freq_id se utilizara despues para el corte en tiempo tambien
cut_freq_id = [];
%Counter for cutting points
cnt = 0;


%Iterate over all cutting points and when reach a pair of poins mod(cnt,2)
%assign them the frecuency square window.


for section = cut_points % Ventana limits
   temp = find(space == interp1(space,space,section,'nearest'));%index in
linespace
   cut_freq_id = [cut_freq_id temp]; % add index of limits
   cnt = cnt +1;
   if(mod(cnt,2) == 0) % Cnt is a pair of cutting values
       rad_space(cut_freq_id(cnt-1):cut_freq_id(cnt)) = 1;
   end
end


%Parametros de la grafica
plot(space,rad_space)
```

```
xlim([-pi pi])
xticks([-pi 0 pi 2*pi 3*pi])
xticklabels({'-\pi','0','\pi'})
xlabel('rad/s')
```



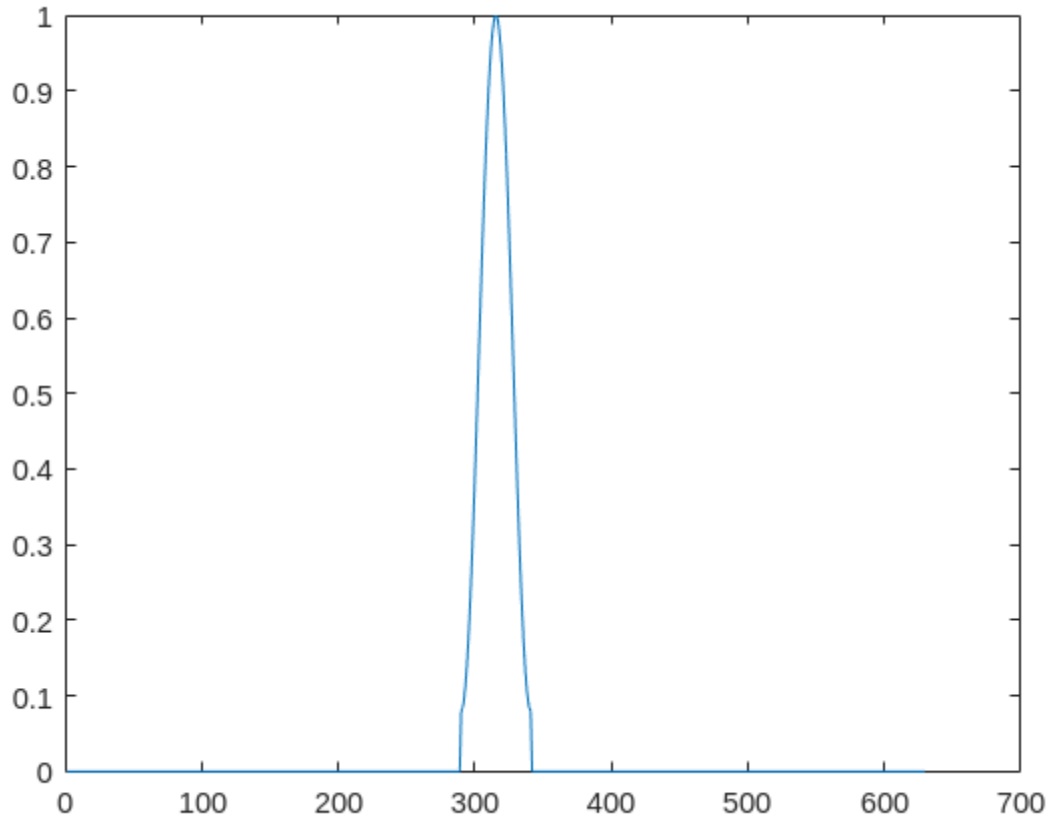
Converimos las ventanas de frecuencia a tiempo.

```
ventana_tiempo = real(ftot(rad_space));
plot(ventana_tiempo)
```

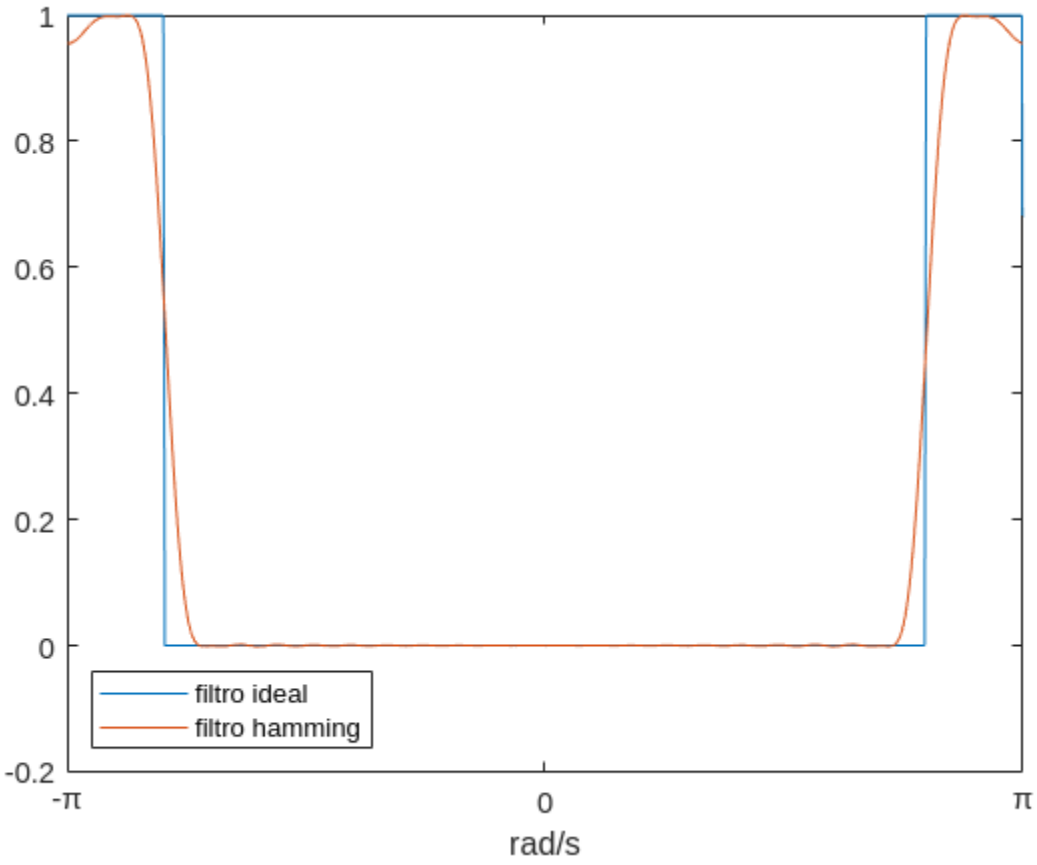


Generacion de ventanas en tiempo, con 51 coeficientes. Y una venta de hamming

```
ventanacuad = zeros(1,length(space));
coeff       = 51;
middle      = (length(space))/2;
left        = int32((middle-coeff/2));
right       = int32((middle+coeff/2));
ventanacuad(left:right) = hamming(coeff+1)';
plot(ventanacuad)
```




```
filtro_cuad=ventana_tiempo.*ventanacuad;
plot(space,rad_space);
hold on
plot(space,real(ttof(filtro_cuad))/max(real(ttof(filtro_cuad))))
xlim([-pi pi])
xticks([-pi 0 pi 2*pi 3*pi])
xticklabels({'-\pi','0','\pi'})
xlabel('rad/s')
legend({'filtro ideal','filtro hamming'},'Location','southwest')
hold off
```


1
0.8
0.6
0.4
0.2
0
-0.2
-π
0
π
rad/s
filtro ideal
filtro hamming

Ejercicio 4

Diseñar 4 filtros de 45 coeficientes, cada uno con las frecuencias positivas de corte ideales en

$$\text{filtro1}= \left[0, \frac{\pi}{4}\right]$$

$$\text{filtro2}= \left[\frac{\pi}{4}, \frac{\pi}{2}\right]$$

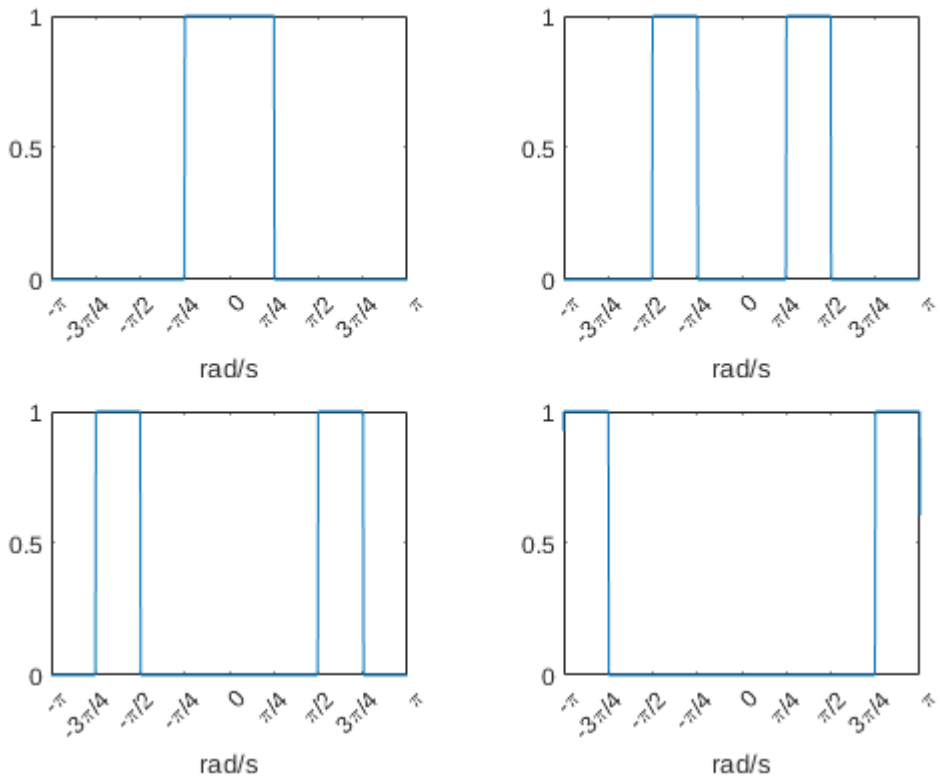
$$\text{filtro3}= \left[\frac{\pi}{4}, \frac{3\pi}{4}\right]$$

$$\text{filtro4}= \left[\frac{3\pi}{4}, \pi\right]$$

Definimos un tiempo de observación de 1 seg. Esto quiere decir que si muestreamos a 8000 muestras por segundo tendremos 1/8000 como tiempo de observación.

Radian domain

```
bound =[-pi,pi];
space = bound(1)-1/100:1/100:bound(2)+1/100;
%generate 4 spaces
rad_space = zeros(4,length(space));
range = 0:pi/4:pi;
% 4 graphs to be ploted
figure(1)
t = tiledlayout(2,2);
side =[-1,1];
for n=2:1:5
    %index in space
    for m=side % negative, positve side
        start_filter = find(space == interp1(space,space,m*range(n-
1),'nearest'));
        end_filter   = find(space == interp1(space,space,m*range(n),'nearest'));
        %4d tensor of filters, n-1 acces to layer of tensor.
        rad_space(n-1,min(start_filter,end_filter):max(start_filter,end_filter))
= 1;
    end
    %plotting stuff
    nexttile
    plot(space,rad_space(n-1,:));
    xlim([-pi pi])
    xticks([-pi -pi*3/4 -pi/2 -pi/4 0 pi/4 pi/2 pi*3/4 pi])
    xticklabels({'-\pi','-3\pi/4','-\pi/2','-
\pi/4','0','\pi/4','\pi/2','3\pi/4','\pi'})
    xlabel('rad/s')
end
```



Sampling Frequency domain

Dividimos el eje por 2π. Esto para hacer la conversion de $\frac{\text{rad}}{s}$ a $\text{Hz} = \frac{1}{s}$

$$\omega = 2\pi f \ , \ f = \frac{\omega}{2\pi}$$

Posteriormente multiplicamos por la frequencia de muestreo $F_S = 8000\text{Hz}$ para obtener el **maximo espacio libre de alias .**

Det tal forma que la venta queda de

$$\left[-\frac{F_s}{2}, \frac{F_s}{2}\right]$$

```
Fs    = 8000; % sampling frequency
Coeff = 45; % num of coeffiientes
```

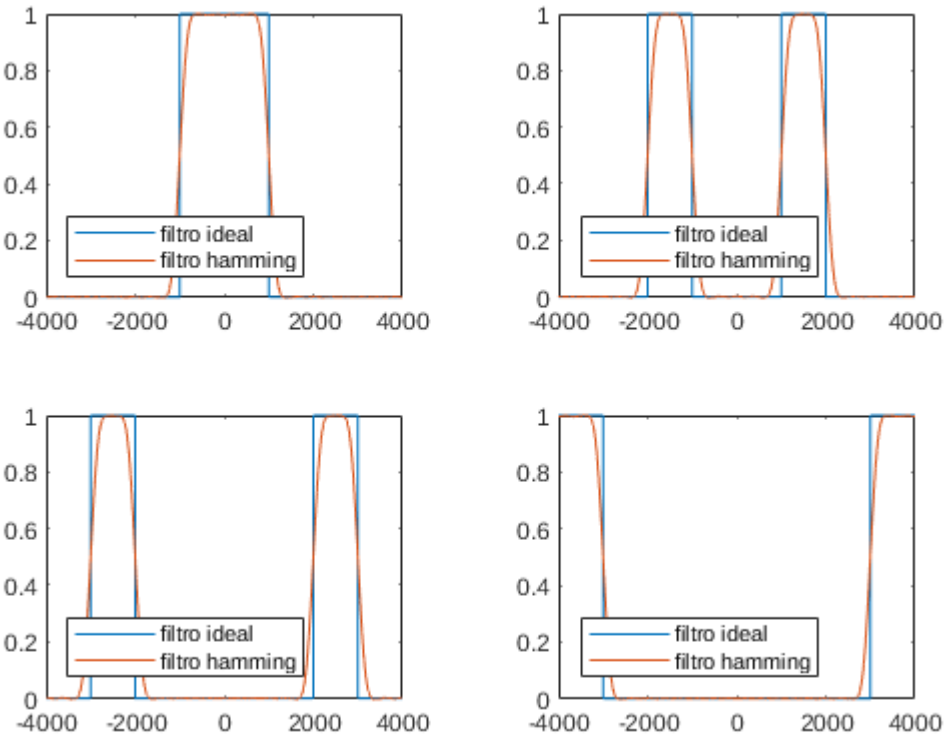
$F_s = \frac{\omega}{2\pi} * F_s$

```
bound_fs = (bound./(2*pi)).*(Fs);
```

1 second of observation time

```
space_fs = bound_fs(1):1:bound_fs(2); % 1 second
%tensors space
filter_fs   = zeros(4,length(space_fs));
filter_time = zeros(4,length(space_fs));
windows     = zeros(4,length(space_fs));
```

```
range_fs = (range./(2*pi)).*Fs;


% 4 graphs to be ploted
figure(2)
dt_fs = tiledlayout(2,2);
side =[-1,1]; % positive and negative side of the spectra
for n=2:1:5
    %index in space
    for m=side % negative, positve side
        start_filter = find(space_fs == interp1(space_fs,space_fs,m*range_fs(n-
1),'nearest'));
        end_filter   = find(space_fs ==
interp1(space_fs,space_fs,m*range_fs(n),'nearest'));
        %4d tensor of filters, n-1 acces to layer of tensor.
        filter_fs(n-1,min(start_filter,end_filter):max(start_filter,end_filter))
= 1;
    end
    %time signal of square filter
    filter_time(n-1,:) = real(ftot(filter_fs(n-1,:)));
    %in this case time signal is centered at 0 so 45 coeff means -22,0,22
    middle      = (length(space_fs))/2;
    left        = int32((middle-Coeff/2));
    right       = int32((middle+Coeff/2));
    %inserting hamming filtering
    windows(n-1,left:right) = hamming(Coeff+1)';
    %convolution
    windows(n-1,:) = windows(n-1,:).*filter_time(n-1,:);
    %normalization
    windows(n-1,:) = real(ttof(windows(n-1,:)))/max(real(ttof(windows(n-1,:))));
    %plotting stuff
    nexttile
    plot(space_fs,filter_fs(n-1,:));
    hold on;
    plot(space_fs,windows(n-1,:));
    legend({'filtro ideal','filtro hamming'},'Location','southwest')
    hold off;
end
```

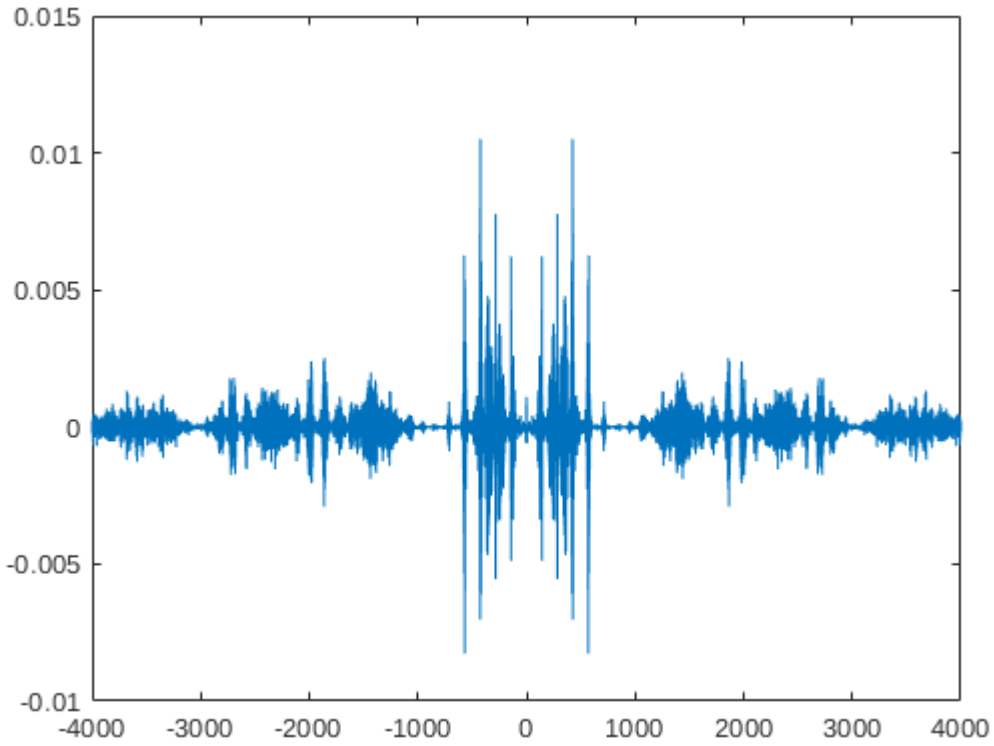


Lectura de Audio

```
figure(7)
[y, audioFS] = audioread("Audios/grape-juice.wav");
```

Ajustamos el sample rate para garantizar los 8000 samples/sec

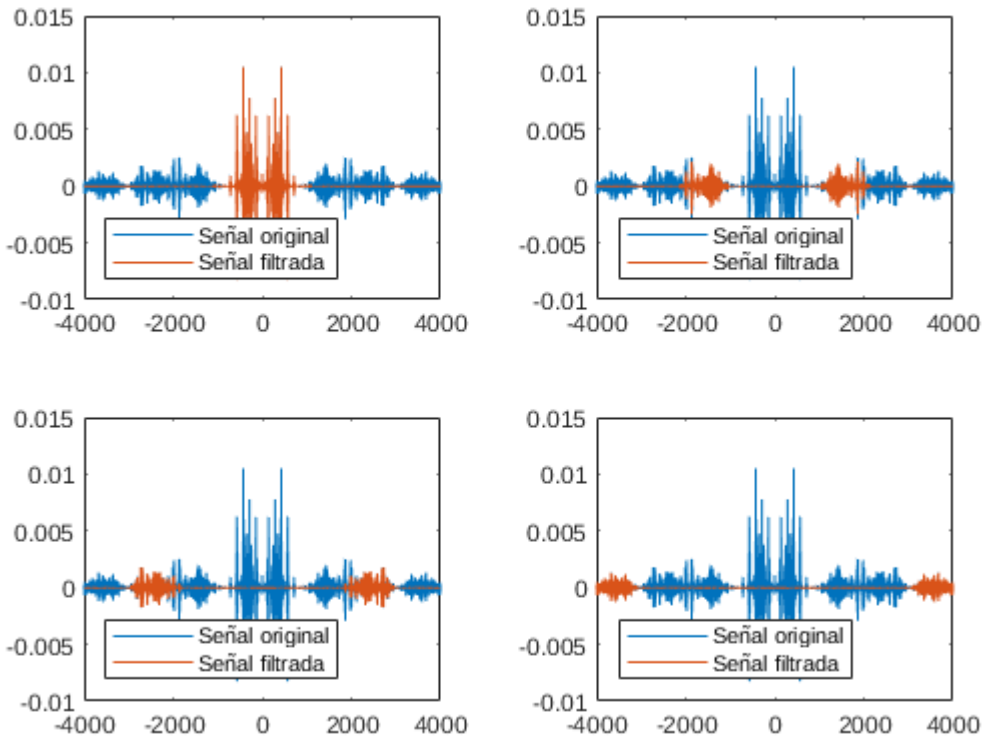
```
s_down_resample = resample(y, Fs, audioFS);
m_t = s_down_resample(1:(Fs+1)); %cut one second final message signal
m_t_f = real(ftot(m_t));
plot(space_fs,m_t_f);
```



Aplicacion de filtros en audio

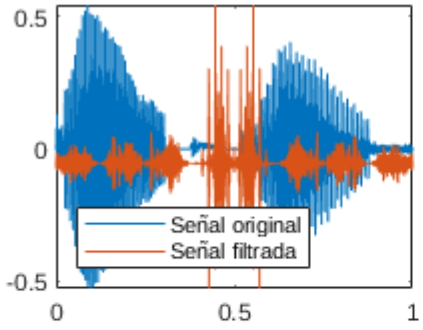
Simplmente queda como la multiplicacion en frecuencia de las dos señales.

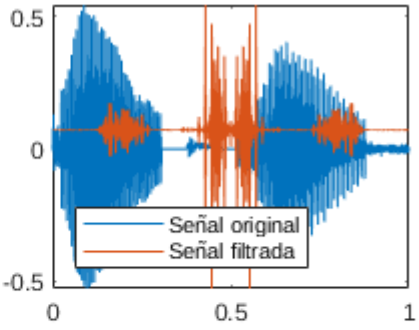
```
figure(4)
filtered_signals = [m_t_f m_t_f m_t_f m_t_f]';
dt_filtered = tiledlayout(2,2);


for n=1:1:4
    filtered_signals(n,:) = filtered_signals(n,:).*windows(n,:);
    %plotting stuff
    nexttile
    plot(space_fs,m_t_f);
    hold on;
    plot(space_fs,filtered_signals(n,:));
    legend({'Señal original','Señal filtrada'},'Location','southwest')
    hold off;
end
```

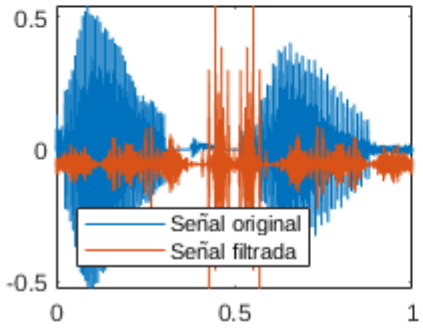


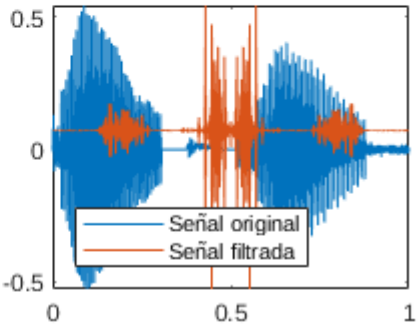
Conversión y comparación en tiempo

```
figure(5)
filtered_time = real (ftot(filtered_signals));
dt_filtered_time = tiledlayout(2,2);
time  = 0:1/Fs:1;


for n=1:1:4
    filtered_time(n,:) = rescale(filtered_time(n,:),min(m_t),max(m_t));
    %plotting stuff
    nexttile
    plot(time,m_t);
    hold on;
    plot(time,filtered_time(n,:));
    legend({'Señal original','Señal filtrada'},'Location','southwest')
    hold off;
end
```


0.5
0
-0.5
0
0.5
1
Señal original
Señal filtrada


0.5
0
-0.5
0
0.5
1
Señal original
Señal filtrada

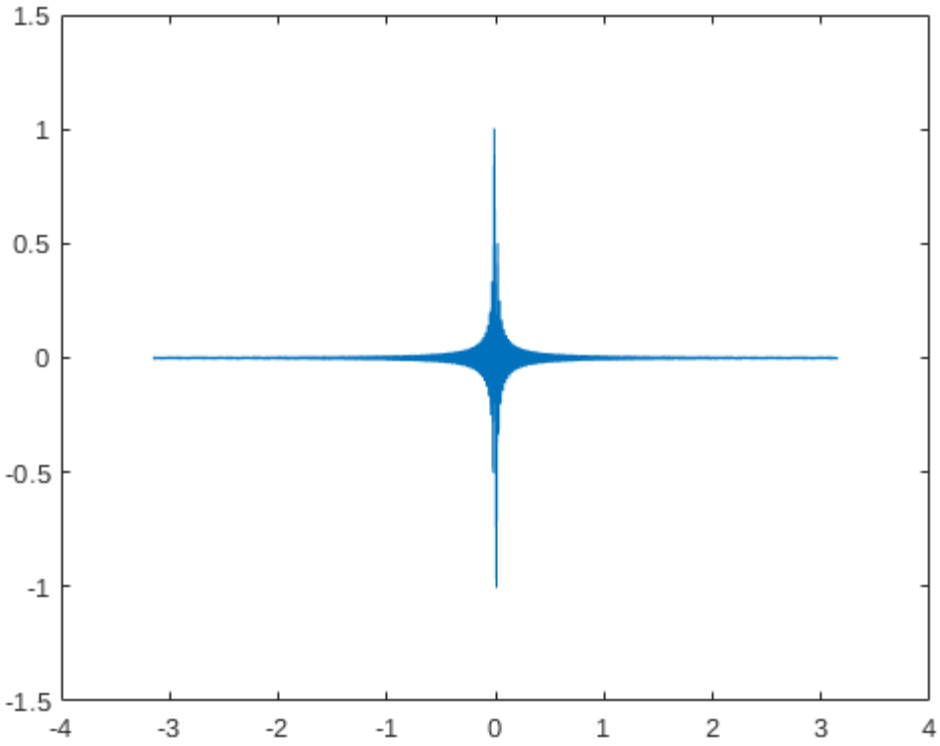

0.5
0
-0.5
0
0.5
1
Señal original
Señal filtrada


0.5
0
-0.5
0
0.5
1
Señal original
Señal filtrada

Ejercicio 5

Un diferenciador realiza la operación y(t) = dx(t)/dt, que en la frecuencia es la multiplicación por jw, por lo tanto, la operación de diferenciación se puede ver como el filtrado con un filtro cuya función de transferencia es jw.

```
bound = [-pi,pi];
w     = bound(1)-1/100:1/100:bound(2)+1/100;
derivator = real(ftot(1i*w));
plot(w,derivator);
```

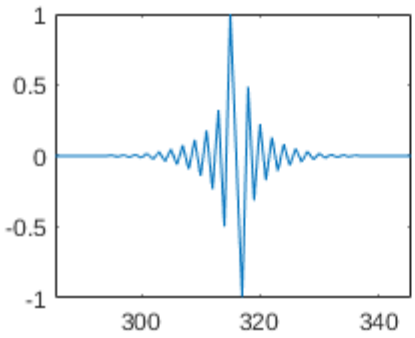


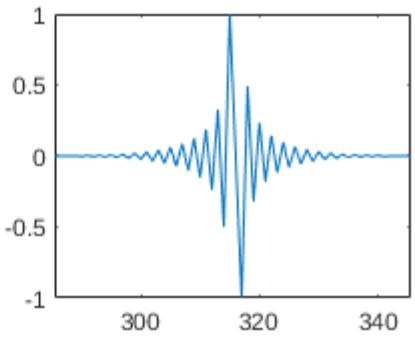
1. Diseñe un filtro FIR de 41, 51 y 61 coeficientes que haga una operación de derivada. (puede utilizar cualquier ventana)

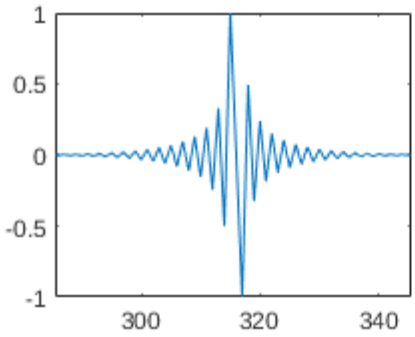
Se crear un filtro de hamming muestreando 41,51,61 coeffiecientes.

```
Coeff    = [41,51,61];
FIR      = zeros (length(Coeff),length(w));
%inserting hamming filtering
middle = (length(w))/2;
figure(1);
dt_fs = tiledlayout(2,2);
for n=1:1:3
    left   = int32((middle-Coeff(n)/2));
    right  = int32((middle+Coeff(n)/2));
    FIR(n,left:right) = hamming(Coeff(n)+1)';
```

```
    FIR(n,:) = derivator.*FIR(n,:);
    nexttile
    plot(FIR(n,:));
    axis([-30+middle middle+30 -1 1])
    %normalization
    FIR(n,:) = ttof(FIR(n,:))/max(ttof(FIR(n,:)));
end
```

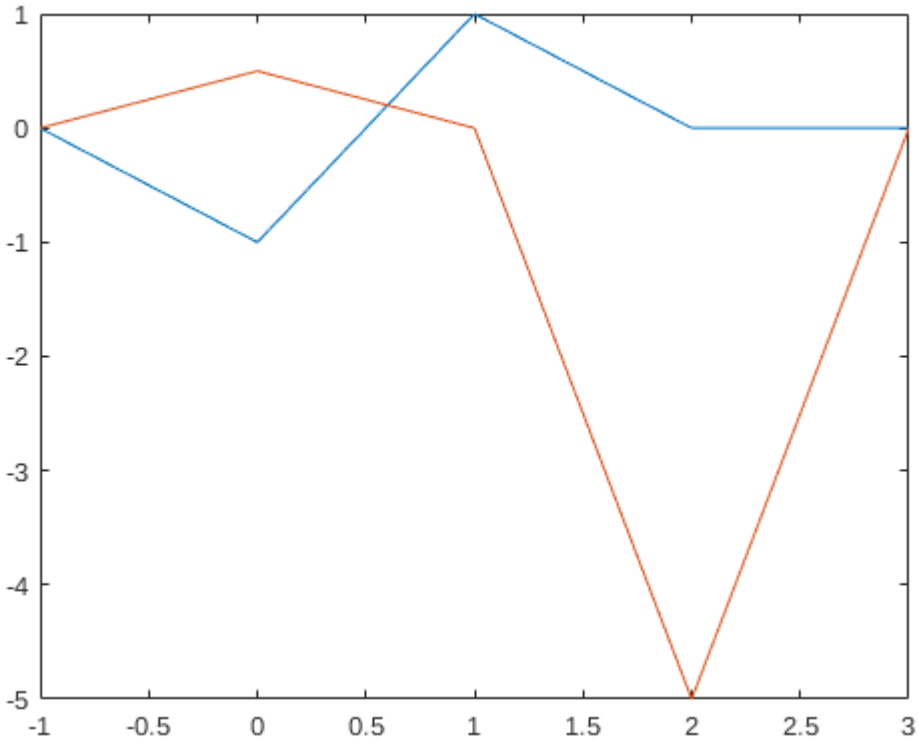






2. Dada las aproximaciones con los filtros diferenciadores h1[n] = {-1,1}, h2[n] = {0.5, 0,-0,5}. Con el cero en el primer coeficiente en ambos filtros. Obtenga de ambos su función de transferencia analítica.

```
figure(3)
time = -1:1:3;
h_1 = [0,-1,1,0,0];
h_2 = [0,.5,0,-5,0];
plot(time,h_1);
hold on;
plot(time,h_2);
hold off;
```



3. Compare las funciones de transferencia obtenida por a) y b) y concluya que tanto se aproximan al derivador real.

4. Introduzca una señal coseno en múltiplos de $\frac{\pi}{10}$ desde 0 a π (es decir, su frecuencia angular discreta) a los filtros anteriores, y verifique a partir de las señales generadas por simulación si los filtros efectivamente realizan la operación de derivación: justifique sus conclusiones. (grafique la señal de salida y la entrada en un mismo cuadro para apoyar sus observaciones)

$$T = \frac{\pi}{10}$$

$$f = \frac{10}{\pi}$$

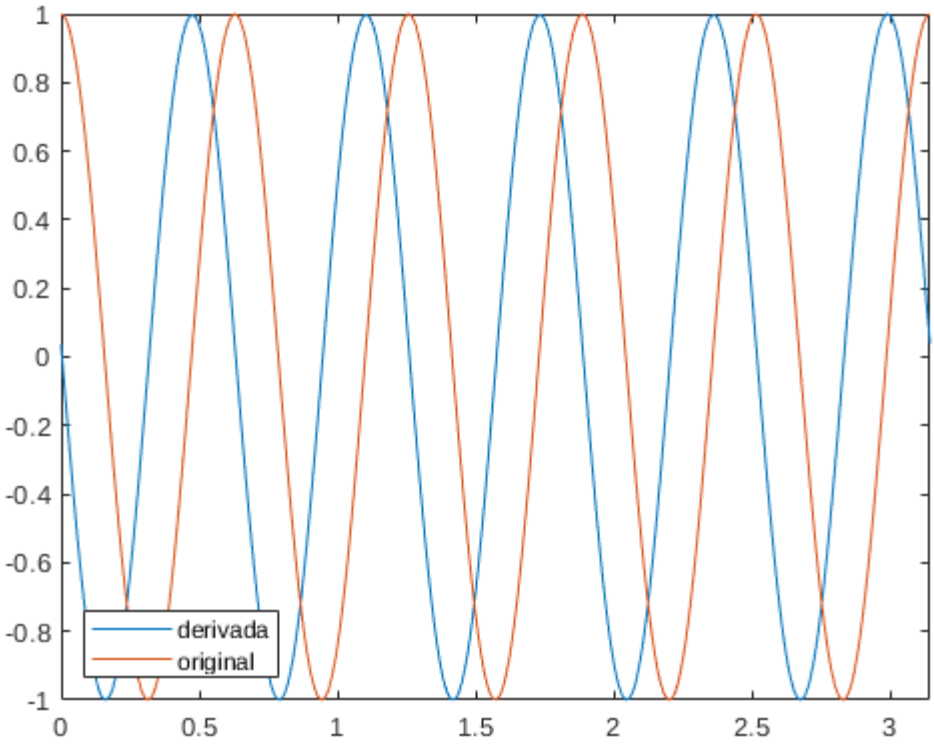
$$\omega = 2\pi \frac{10}{\pi}$$

```
signal = cos(10*w);
```

Aplicacion de filtro derivador

```
figure(6)
derivative= real(ftot(ttof(signal).*FIR(1,:)));
derivative= rescale(derivative,-1,1);
plot(w,derivative);
hold on;
```

```
plot(w,signal);
hold off;
axis([0 pi -1 1])
legend({'derivada','original'},'Location','southwest')
```



Observaciones

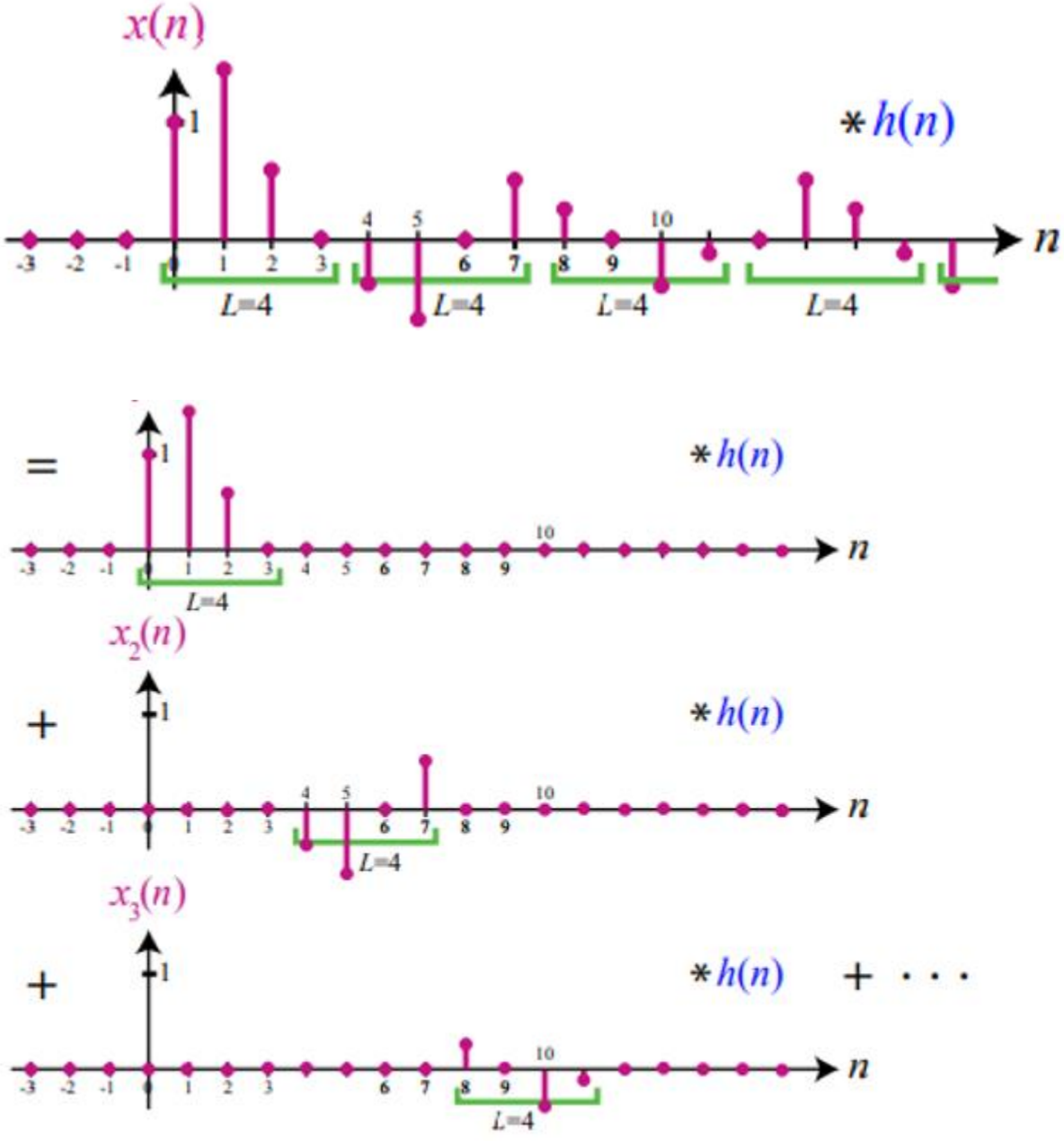
Pese al filtro estar normalizado en frecuencia y tener un factor de coeficientes considerables fue necesario un escalamiento de la señal, dado que la señal derivada salía muy pequeña.

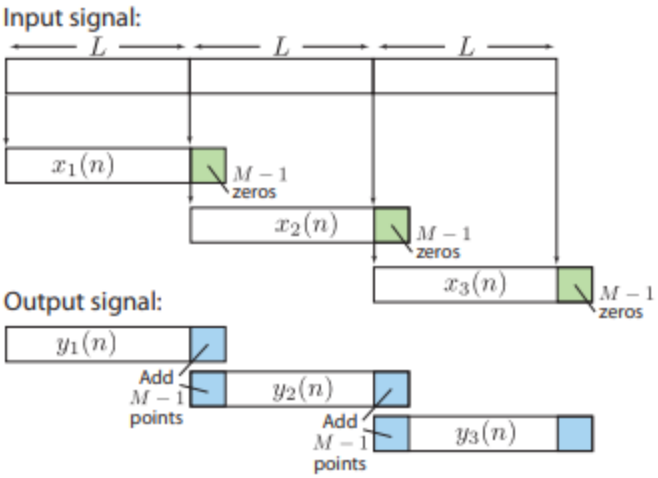
Ejercicio 6

Explicar en qué consiste la técnica *overlap and add* y *overlap and save* para el filtrado de secuencias arbitrariamente largas mediante el filtrado por multiplicación en la frecuencia.

overlap and add

La entrada se divide en bloques que no se solapan entre si $x_m(n)$ cada longitud L. Cada bloque de entrada $x_m(n)$ es individualmente filtrado y recibido para producir la salida del bloque $y_m(n)$

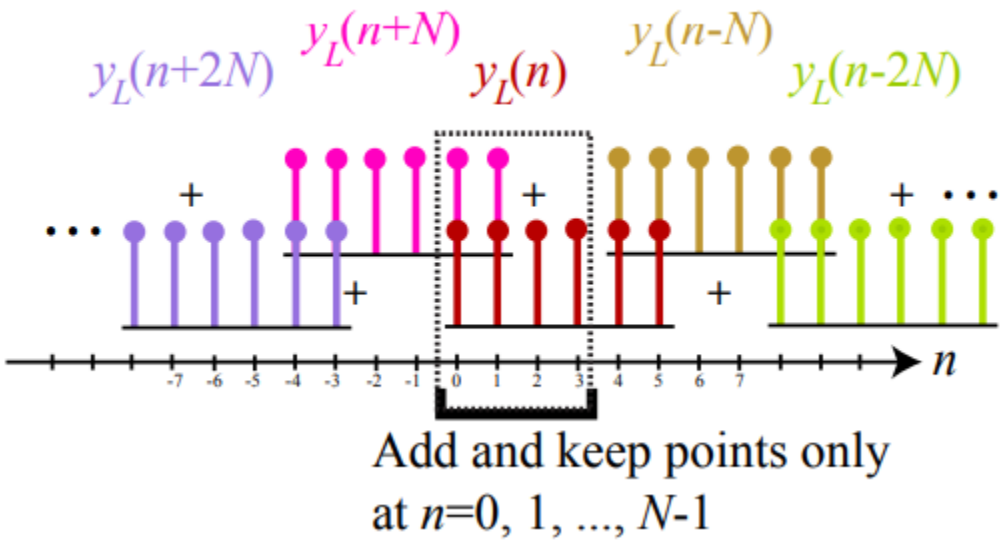


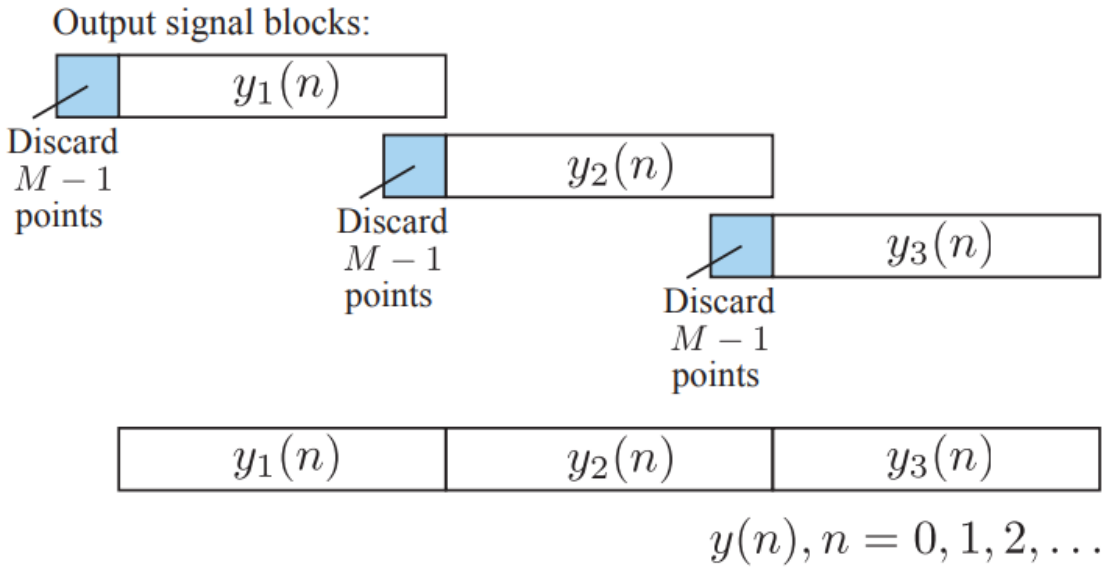


Algoritmo

1. Separar la señal x(n) en bloques que no se solapen entre si $x_m(n)$ de longitud L.
2. Hacer Zero Pad h(n) para tener la longitud N=L+M-1
3. Tomar la N-DFT de h(n) para dar H(k), k=0,1,2,3, … N-1
4. Para cada boque m:
    a. Zero pad $x_m(n)$ para que sea de longitud N=L+M-1
    b. Obetenr la N-DFT de $x_m(n)$ para tener $X_m(k)$, k=0,1,2,3, … N-1
    c. Multiplicar: $Y_m(k) = X_m(k)^* H_m(k)$, k=0,1,…N-1
    d. Obtener la N-IDFT de $Y_m(k)$ para dar $y_m(n)$ n=0,1,…,N-1
5. De y(n) con superposición del ultimo M-1 muestras de $y_m(n)$ con el primer M-1 muestras de $y_{m+1}(n)$ y sumando el resultado.

overlap and save





Algoritmo

1. Insertar M-1 ceros en el principio de la secuencia de entrada x(n).
2. Separar en partes la señal de entrada en bloques superposicionados $x_m(n)$ de longitud N= L+M-1 donde la longitud del solapamiento es M-1
3. Zero pad h(n) para que sea de longitud N=L+M-1
4. Obtener N-DFT de h(n) para poder obtener H(k), k=0,1, … N-1
5. Para cada bloque m:
    a. Obtener la N-DFT de $x_m(n)$ para dar $X_m(k)$ n=0, 1,…,N-1
    b. Multiplicar: $Y_m(k) = X_m(k)^* H_m(k)$, k=0, 1,…N-1
    c. Obtener la N-IDFT de $Y_m(k)$ para tener $y_m(n)$, k=0,1,2,3, … N-1
    d. Descartar la primera M-1 puntos de cada bloque de salida $y_m(n)$.
6. Formar y(n) sumando el restante (por ejemplo, del último) L muestras por cada bloque de $y_m(n)$.