Práctica 1: Procesado en Tiempo Discreto de Señales Continuas

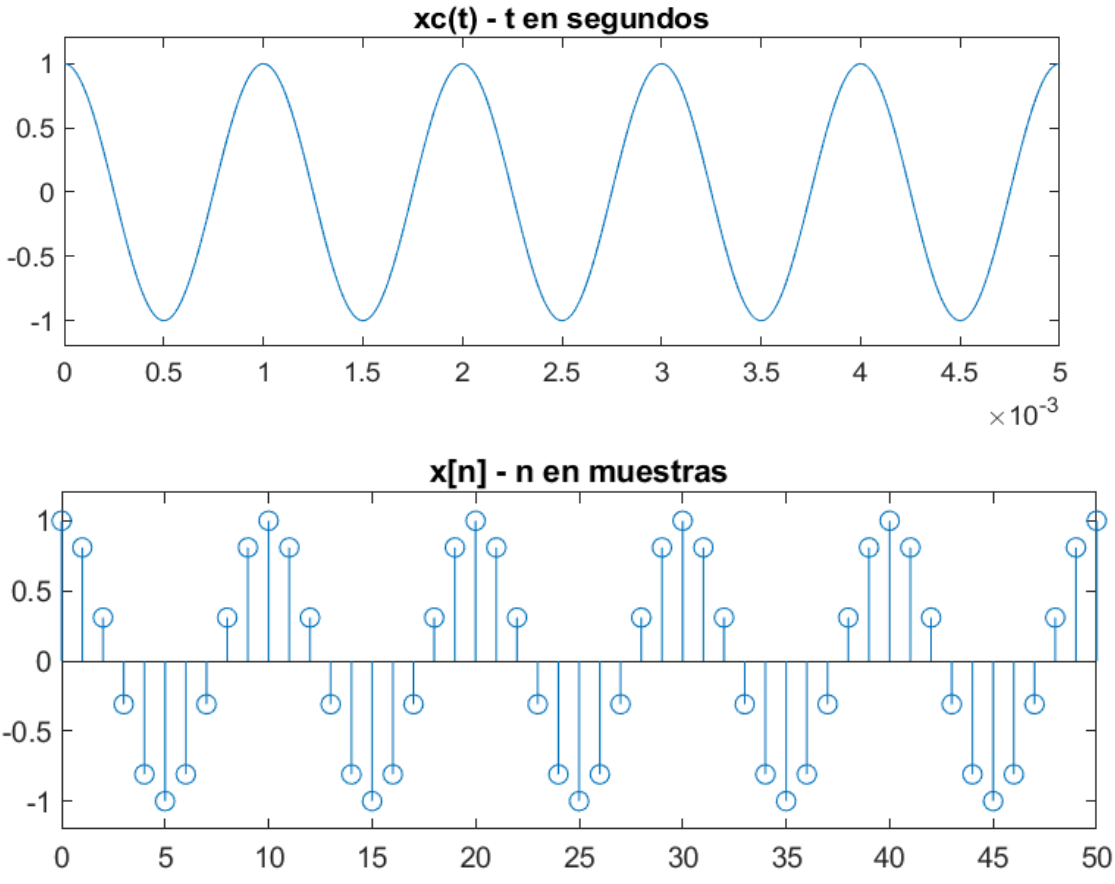
1. Muestreo de una Señal Continua

1.1. Construya la señal continua $x_c(t) = \cos(\omega_0 t)$, donde $\omega_0 = 2000\pi$ rad/s, usando $t = 0 : T_c : 1$ con $T_c = 10^{-5}$ s. Obtenga a continuación la secuencia discreta $x[n] = x_c(nT_s)$ muestreando $x_c(t)$ con una frecuencia de muestreo $f_s = 10$ kHz. Calcule la dimensión que debe tener el vector n de forma que $x[n]$ cubra toda la longitud de la señal $x_c(t)$

```
Tc= 1E-5;
t=0:Tc:1;
xc=cos(2000*pi*t);

subplot(211);
plot(t,xc);
title('xc(t) - t en segundos');

fs=10000;
Ts=1/fs;
n=0:1:fs;
xcn=cos(2000*pi*Ts*n);

subplot(212);
stem(n,xcn);
title('x[n] - n en muestras');
```

1.2. Represente $x_c(t)$ usando la función **plot** y $x[n]$ mediante la función stem. Utilice la funcion **subplot**, de modo que pueda observar ambas señales de forma simultánea, pero cada una según su eje de tiempos, para comprobar el efecto del muestreo en el tiempo.Nota: Para realizar esta representación es conveniente utilizar sólo las muestras correspondientes a los 5 primeros milisegundos de ambas señales.

```
Tc= 1E-5;
t=0:Tc:1;
xc=cos(2000*pi*t);

subplot(211);
plot(t,xc);
axis([0 5E-3 -1.2 1.2]);
title('xc(t) - t en segundos');

fs=10000;
Ts=1/fs;
n=0:1:fs;
xcn=cos(2000*pi*Ts*n);

subplot(212);
stem(n,xcn);
axis([0 50 -1.2 1.2]);
title('x[n] - n en muestras');
```



1.3. Utilizando la señal x[n] **completa**, encuentre la señal "reconstruida", $y_r(t)$, haciendo uso de la función **interp**.

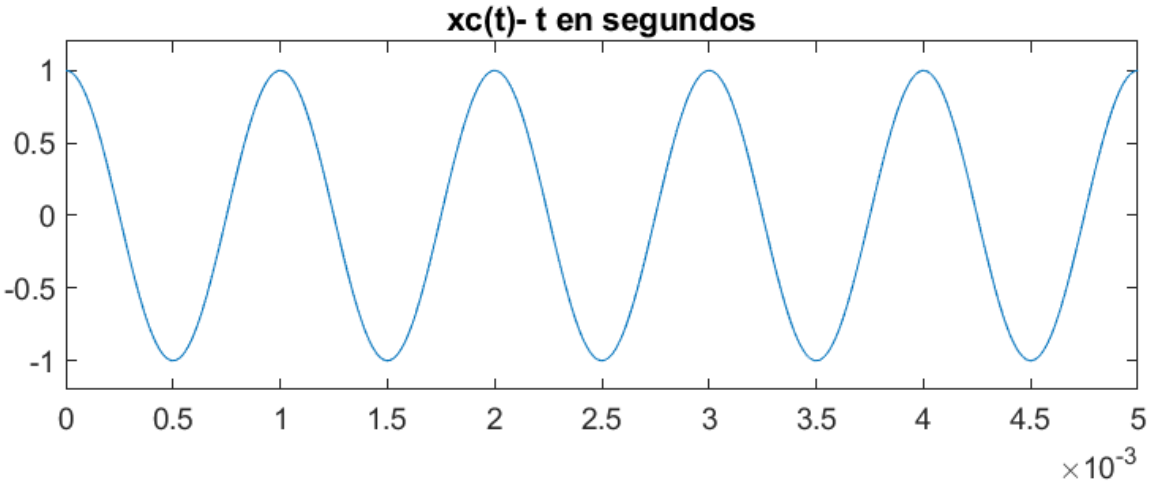
Nota: Tenga en cuenta que el factor de interpolación que debe utilizar es el que le permita pasar de la frecuencia de muestreo, f_s, a la utilizada para simular la señal continua con una rejilla suficientemente fina, $f_c = 1/T_c$.

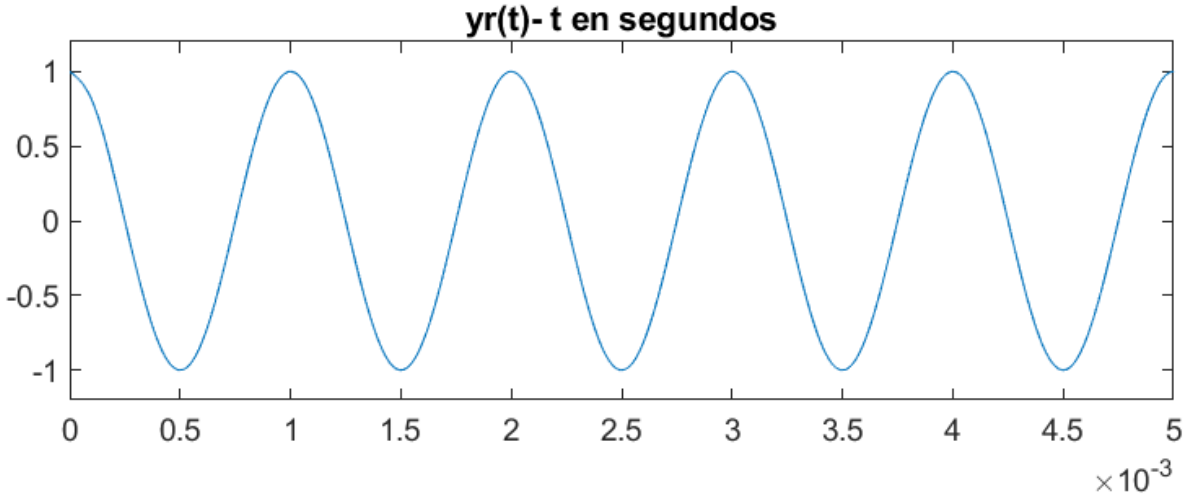
1.4. Utilice la función sound para escuchar las señales $x_c(t)$ e $y_r(t)$. ¿Encuentra alguna diferencia? ¿Coincide este resultado con lo esperado? Refuerce sus afirmaciones representando, con la función **plot**, ambas señales en el intervalo de 0 a 5ms.

Nota: Al utilizar la función sound, no olvide definir correctamente la frecuencia de muestreo. Además, para realizar esta parte es recomendable disponer de unos auriculares.

A primera vista no hay direncias, con la representacion grafica se confirma que son iguales

```
% Apartados 1.3 y 1.4
P= Ts/Tc;
yr=interp(xcn, P);

t=0:Tc:1;
yr=yr(1:length(t));

%%sonido señal xc
sound(xc,1/Tc);


%%sonido señal yr
sound(yr,1/Tc);


subplot(211);
plot(t,xc);
axis([0 5e-3 -1.2 1.2]);
title('xc(t)- t en segundos');


subplot(212);
plot(t,yr);
axis([0 5e-3 -1.2 1.2]);
title('yr(t)- t en segundos');
```

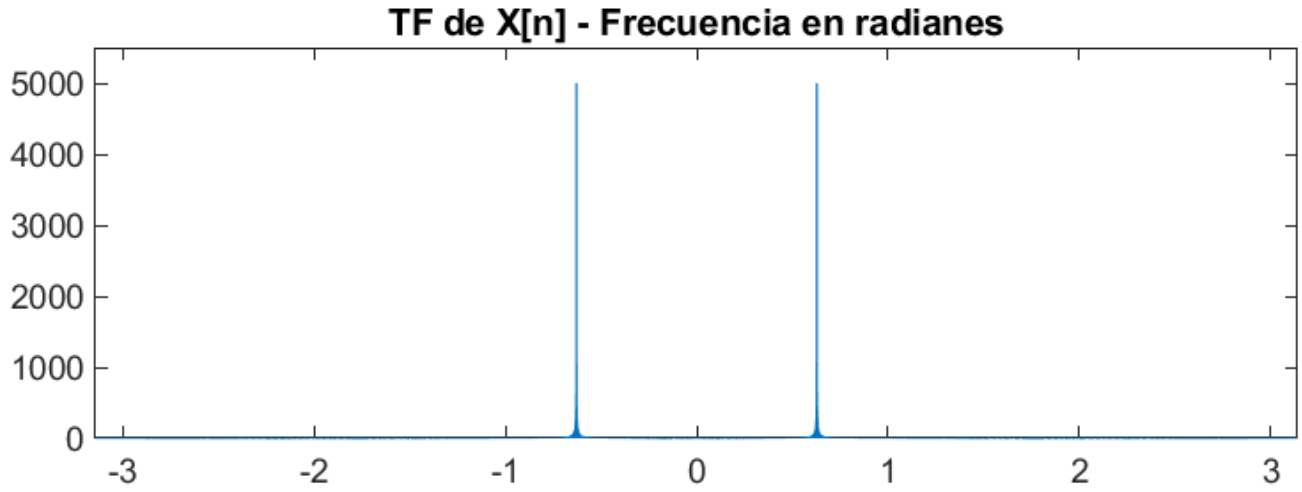




1.5. Haciendo uso de la función **fft**, obtenga el espectro de $x[n]$, $X(e^{j\Omega})$, y represente su módulo entre $-\pi$ y π rad.

Nota: Utilice la función **fftshift** y recuerde definir correctamente el eje de frecuencias, tal y como se vio en la práctica anterior.

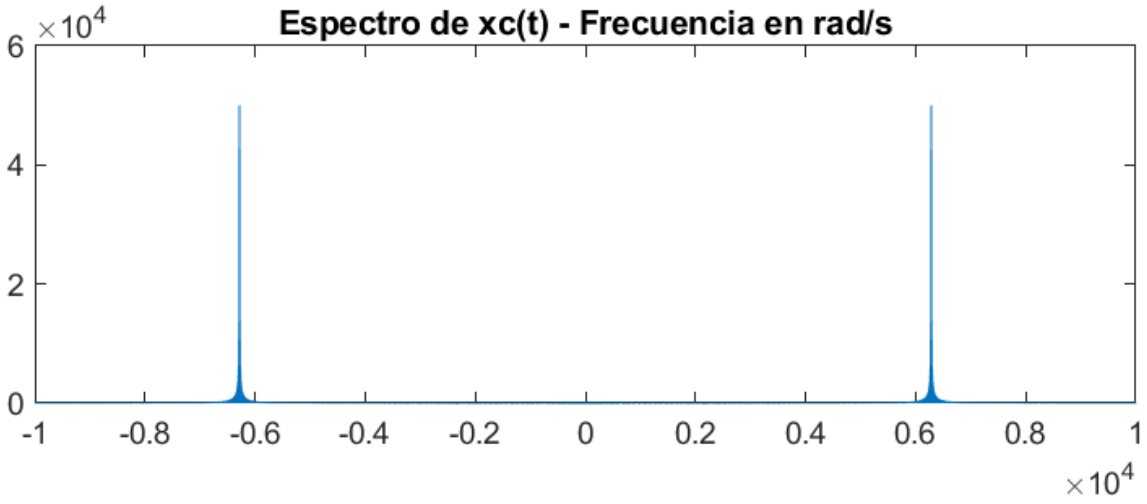
```
n=0:1:10000;
fs=10000;
xcn=cos(2000*pi*1/fs*n);

N=20000;
Om = -pi: 2*pi/N: pi-2*pi/N;
XF = fft(xcn,N);
XF = fftshift(XF);

plot(Om,abs(XF));
 axis([-pi pi 0 5500]);

title ('TF de X[n] - Frecuencia en radianes')
```

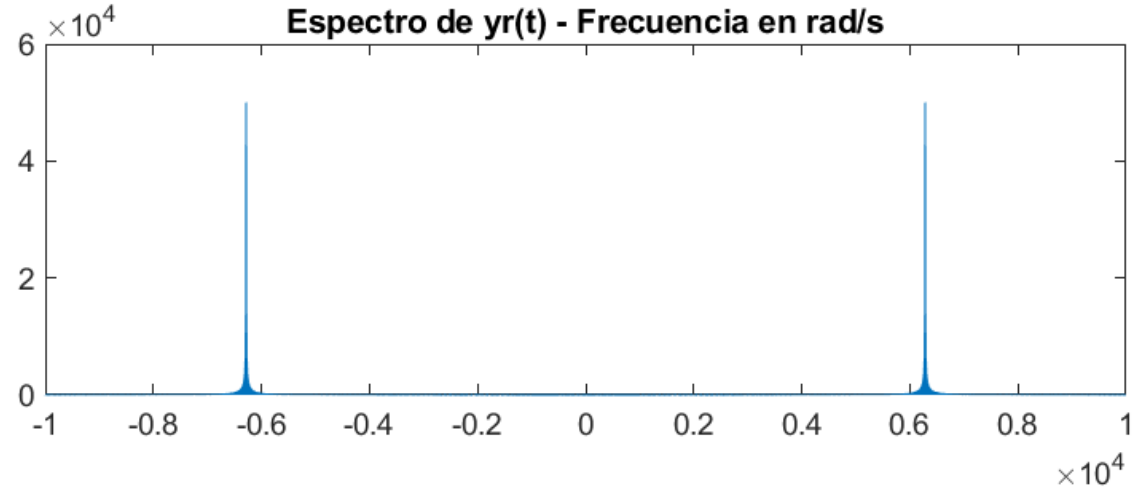


1.6. Represente el módulo del espectro de las señales $x_c(t)$ e $y_r(t)$, $X_c(j\omega)$ e $Y_r(j\omega)$, haciendo uso de las funciones plot y subplot. Nótese que, para representar correctamente el eje frecuencial correspondiente a las señales continuas simuladas, debe tener en cuenta la relación entre frecuencias continuas y discretas, $\Omega = \omega \times T_c$, siendo T_c el periodo de muestreo utilizado para simular las señales continuas. ¿Existen diferencias entre los espectros obtenidos? ¿Coincide este resultado con lo esperado? Represente el eje de frecuencias en Hz.

```
fs=10000;
n=0:1:fs;
Tc=1E-5;
N=200000;
P= Ts/Tc;
xcn=cos(2000*pi*1/fs*n);
yr=interp(xcn, P);

Om = -pi: 2*pi/N: pi-2*pi/N;
Omc = Om/Tc;

xc_espectro = fft(xc,N);
XC_espectro = fftshift(xc_espectro);

subplot (211);
plot(Omc,abs(XC_espectro));
axis([-1E4 1E4 0 6E4]);
title ('Espectro de xc(t) - Frecuencia en rad/s');
```

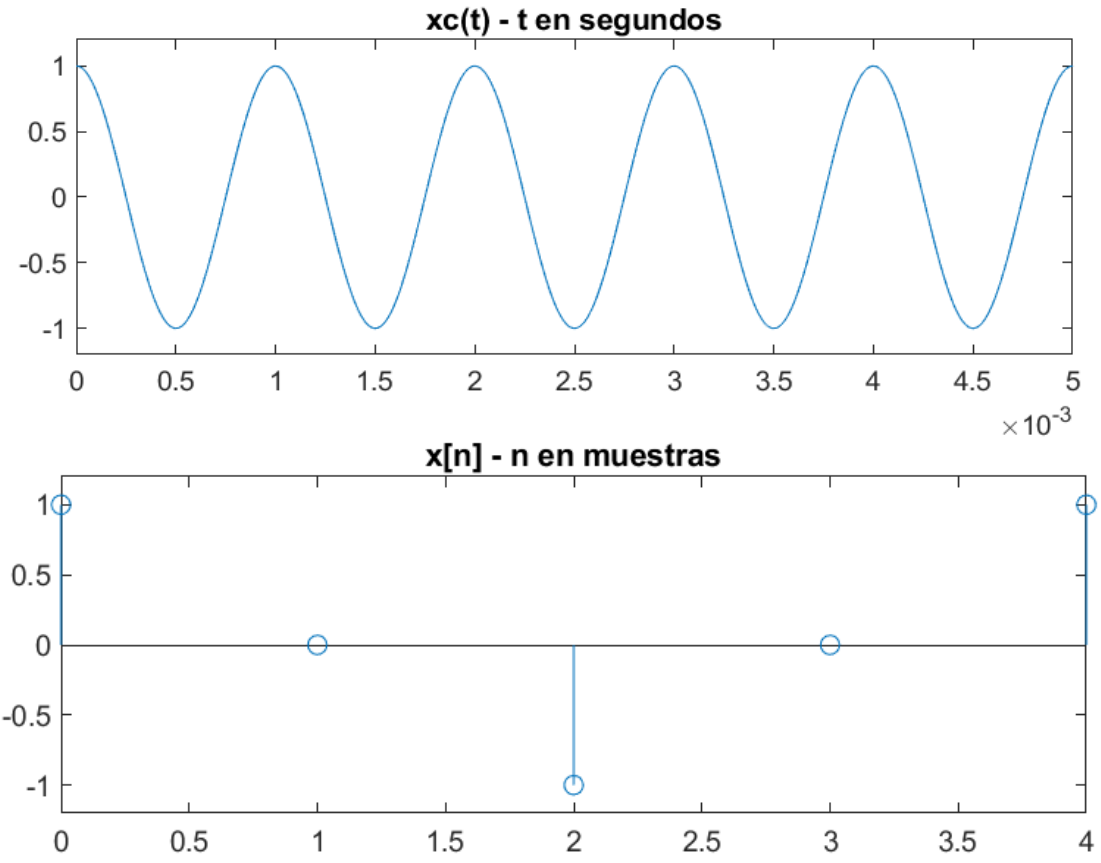
```
yr_espectro = fft(yr,N);
Yr_espectro = fftshift(yr_espectro);
subplot (212);
plot(Omc,abs(Yr_espectro));
axis([-1E4 1E4 0 6E4]);
title ('Espectro de yr(t) - Frecuencia en rad/s');
```



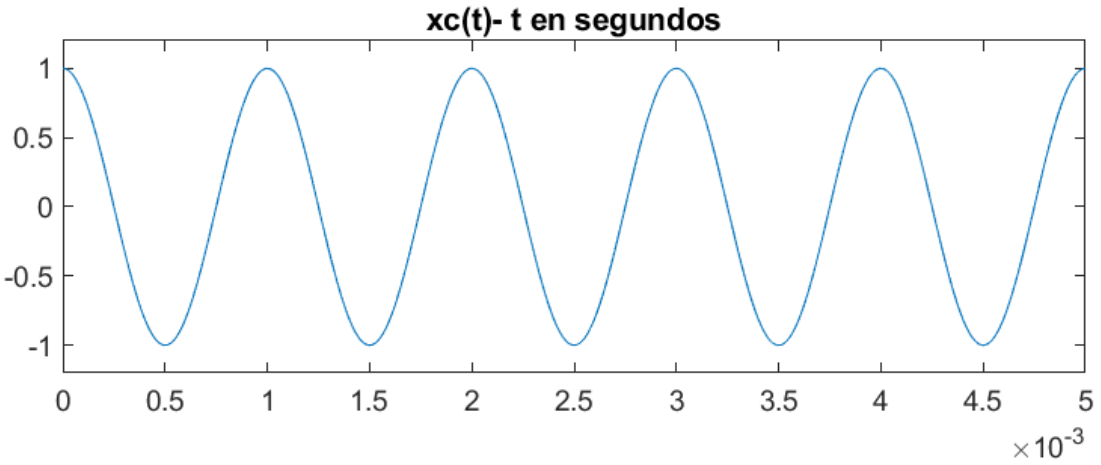


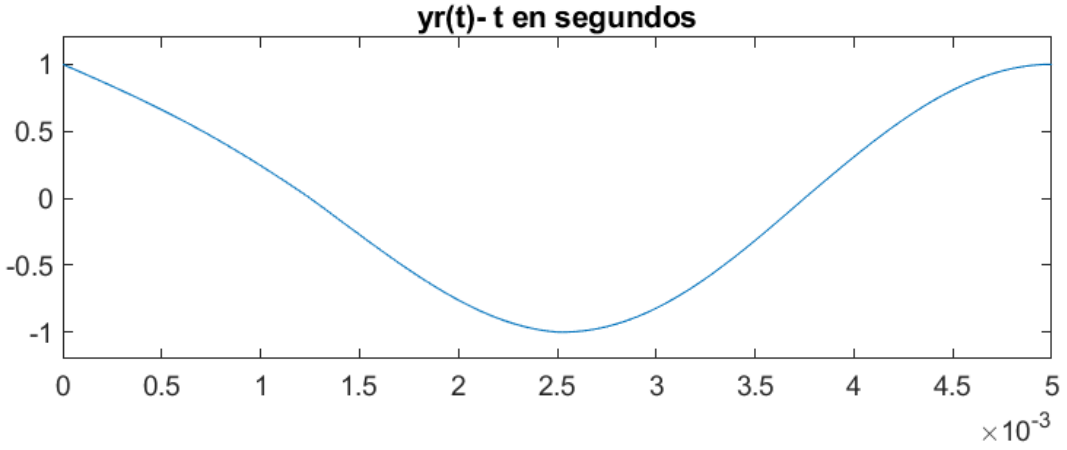
1.7. Repita todo el proceso anterior (Ejercicios 3.1 – 3.6) para una frecuencia de muestreo $f_s = 800$ Hz. ¿Encuentra diferencias con el proceso anterior? ¿Qué efecto se ha producido?

```
%Apartado 1.7
%1.1 y 1.2
Tc= 1E-5;
t=0:Tc:1;
xc=cos(2000*pi*t);

subplot(211);
plot(t,xc);
axis([0 5E-3 -1.2 1.2]);
title('xc(t) - t en segundos');

fs=800;
Ts=1/fs;
n=0:1:fs;
xcn=cos(2000*pi*Ts*n);

subplot(212);
stem(n,xcn);
axis([0 4 -1.2 1.2]);
title('x[n] - n en muestras');
```

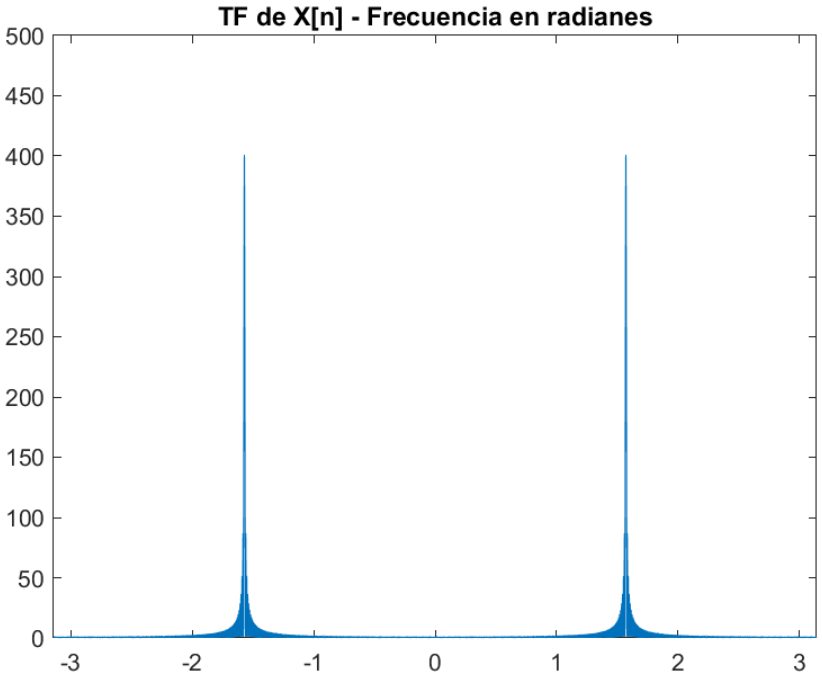


```
%Apartado 1.7
%1.3 y 1.4
P= 125;
yr=interp(xcn, P);
t=0:Tc:1;
yr=yr(1:length(t));

sound(xc,1/Tc);
sound(yr,1/Tc);

subplot(211);
plot(t,xc);
title('xc(t)- t en segundos');
axis([0 5e-3 -1.2 1.2]);

subplot(212);
plot(t,yr);
title('yr(t)- t en segundos');
axis([0 5e-3 -1.2 1.2]);
```

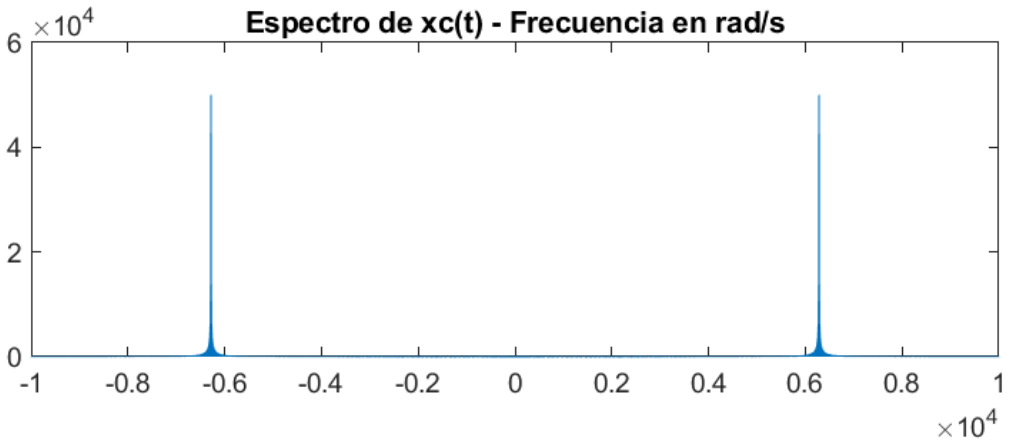


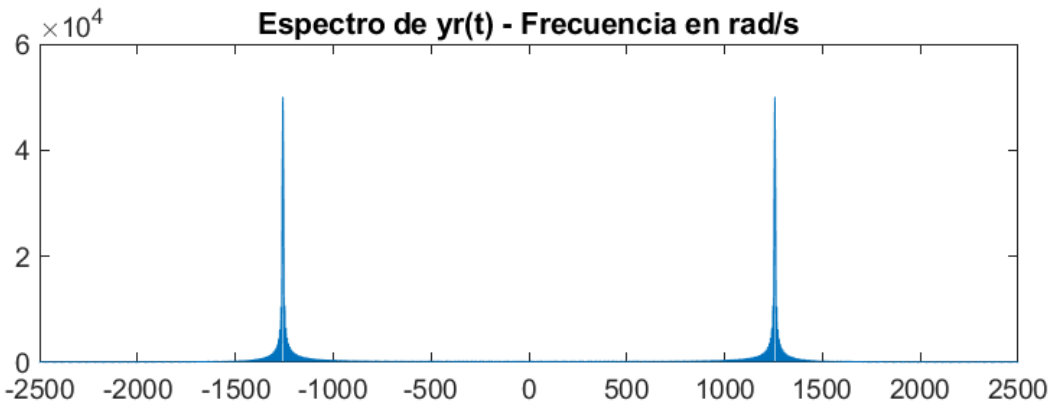


```
%Apartado 1.7
%1.5
fs=800;
n=0:1:fs;
xcn=cos(2000*pi*1/fs*n);

N=20000;
Om = -pi: 2*pi/N: pi-2*pi/N;
XF = fft(xcn,N);
XF = fftshift(XF);

plot(Om,abs(XF));
 axis([-pi pi 0 500]);

title ('TF de X[n] - Frecuencia en radianes')
```



```
%Apartado 1.7
%1.6
fs=800;
n=0:1:fs;
Tc=1E-5;
N=200000;
P= Ts/Tc;
xcn=cos(2000*pi*1/fs*n);
yr=interp(xcn, P);

Om = -pi: 2*pi/N: pi-2*pi/N;
Omc = Om/Tc;

xc_espectro = fft(xc,N);
XC_espectro = fftshift(xc_espectro);

subplot (211);
plot(Omc,abs(XC_espectro));
axis([-1E4 1E4 0 6E4]);
title ('Espectro de xc(t) - Frecuencia en rad/s');

yr_espectro = fft(yr,N);
Yr_espectro = fftshift(yr_espectro);
subplot (212);
plot(Omc,abs(Yr_espectro));
axis([-2500 2500 0 6E4]);
title ('Espectro de yr(t) - Frecuencia en rad/s');
```





2. Procesado Digital de Señales Continuas

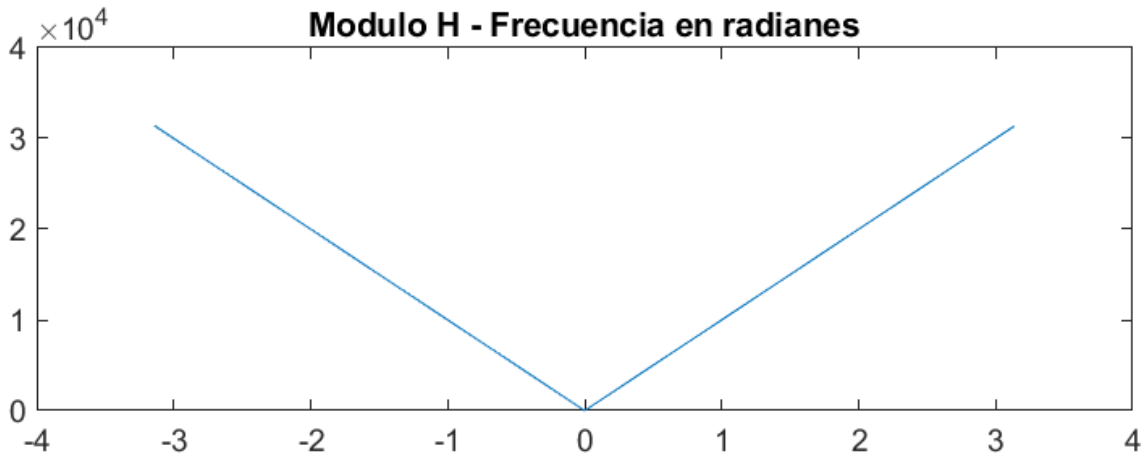
En esta sección se pretende revisar el diseño del sistema discreto equivalente a un sistema continuo dado en el dominio frecuencial. Para ello se va a abordar el diseño de un diferenciador.

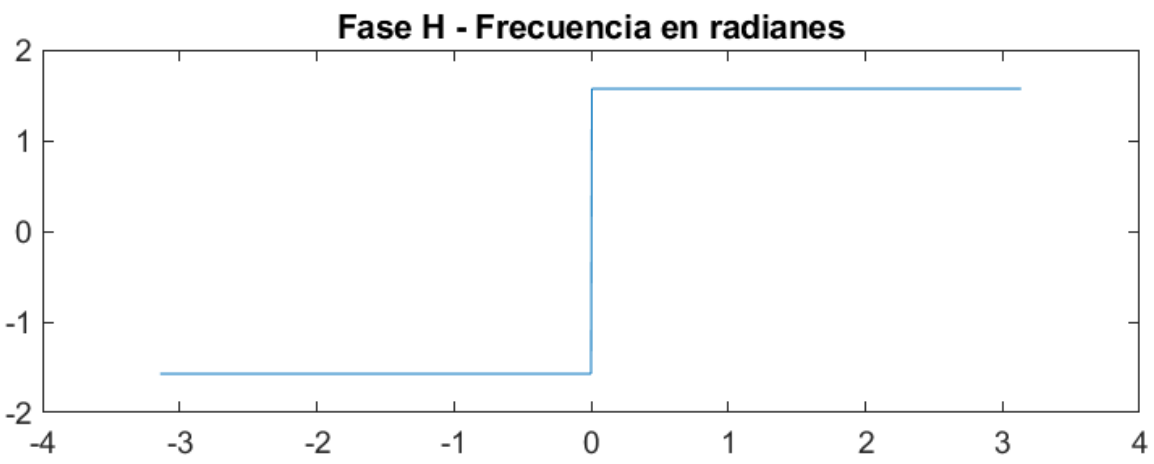
2.1. La relación entrada-salida de un diferenciador en tiempo continuo es

$$y(t) = \frac{dx(t)}{dt}.$$

Encuentre **teóricamente** la respuesta en frecuencia de este sistema continuo, $H(j\omega)$, as´ı como la respuesta en frecuencia del filtro discreto "equivalente", $H(e^{j\Omega})$, que permite implementarlo usando una frecuencia de muestreo f_s = 10 kHz. Represente el módulo y la fase de $H(e^{j\Omega})$ entre $-\pi$ y π rad.

```
fs=10000;
N = 1000;
Ts=1/fs;
n=0:1:fs;
Om = -pi: 2*pi/N: pi-2*pi/N;
H= 1i.*(Om/Ts);

subplot(211);
plot(Om, abs(H));
axis([-4 4 0 4E4]);
title('Modulo H - Frecuencia en radianes');

subplot(212)
plot(Om, angle(H));
title('Fase H - Frecuencia en radianes');
```





2.2. Utilice el filtro obtenido en el apartado anterior para encontrar la señal diferenciada, $y(t)$, cuando la señal de entrada es $x(t) = \cos(\omega_0 t)$ con $\omega_0 = 2000\pi$ rad/s. Para ello, realice el siguiente proceso en Matlab:

a) Obtenga $x[n]$ tomando exactamente $D = 1000$ muestras de la señal $x(t)$ con una frecuencia de muestreo $f_s = 10$ kHz.

b) Encuentre el espectro de la señal $x[n]$ usando la función **fft**.

c) Halle el espectro de la señal de salida en el dominio discreto, $Y(e^{j\Omega})$, haciendo pasar $X(e^{j\Omega})$ por el filtro obtenido en el apartado anterior en el dominio frecuencial:

$$Y(e^{j\Omega}) = X(e^{j\Omega})H(e^{j\Omega}).$$

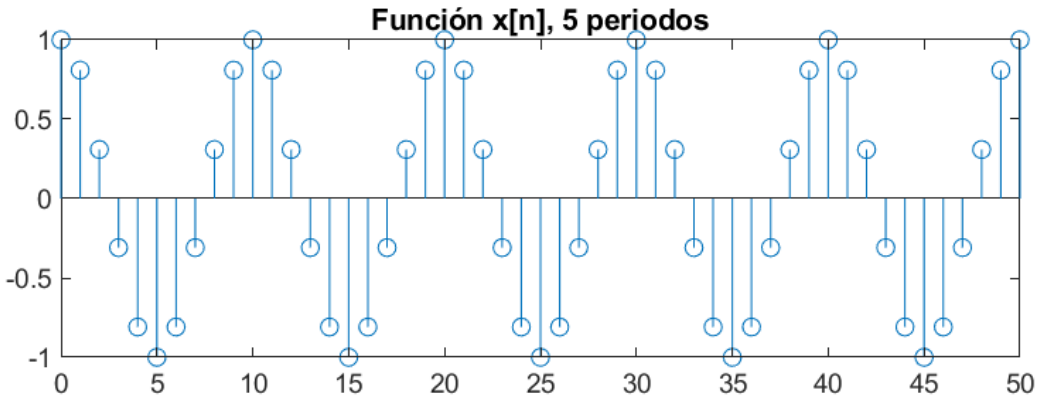
Nota: Tenga en cuenta que, para que este producto tenga sentido, tanto $X(e^{j\Omega})$ como $H(e^{j\Omega})$ tienen que estar definidos entre 0 y 2π.

d)) Calcule $y[n]$ como la transformada de Fourier inversa de $Y(e^{j\Omega})$, utilizando el comando **ifft**.

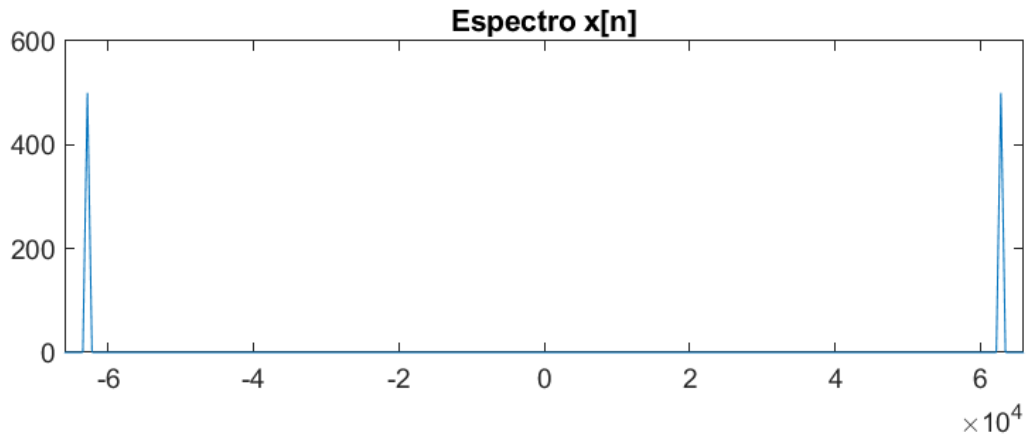
Nota: Al utilizar la función **ifft** es conveniente usar la opción 'symmetric' para evitar que aparezca una pequeña parte imaginaria en $y[n]$ debido a errores numéricos.

e) Obtenga $y(t)$ realizando la reconstrucción de la señal $y[n]$ de forma que la rejilla de simulación del plano continuo sea lo suficientemente fina (por ejemplo, usando $T_c = 10^{-5}$ s).

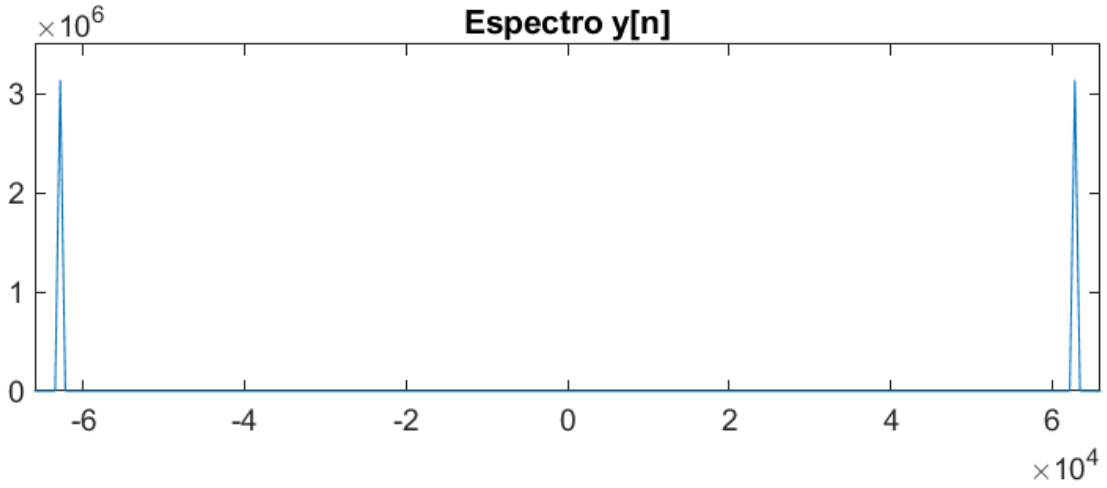
```
%Apartado 2.2
a)

D = 1000;
fs = 10000;
n = 0:1:fs;
Ts = 1/fs;

xcn=cos(2000*pi*Ts*n);
n=0:1:fs;

subplot(211);
stem(n,xcn);
axis([0 50 -1 1]);
title ('Función x[n], 5 periodos');
```



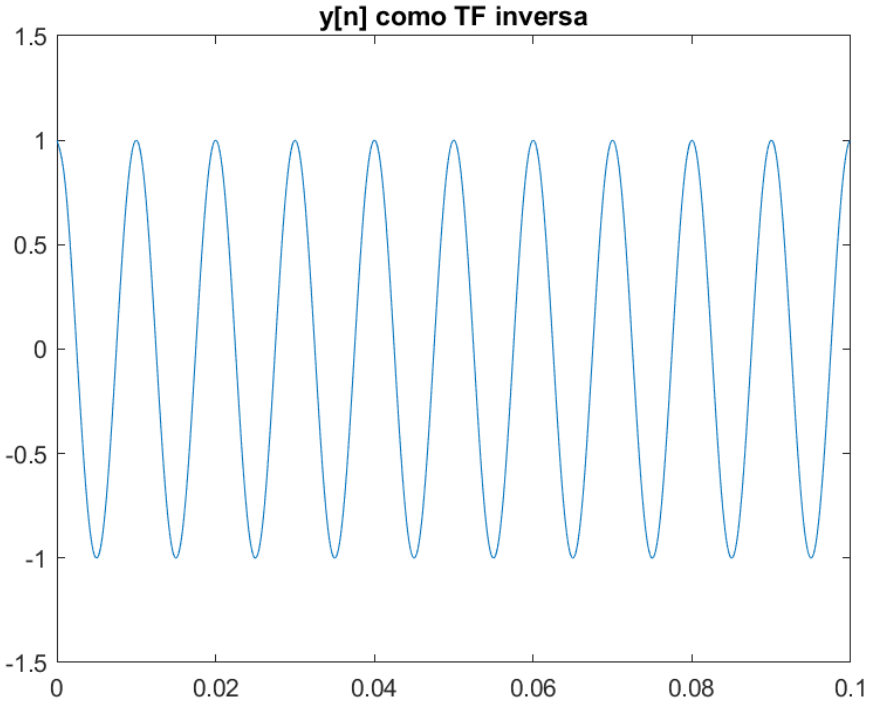
```
%Apartado 2.2
b)
Tc = 1E-5
D = 1000;
fs = 10000;
n = 0:1:fs;
Ts = 1/fs;

Om = -pi: 2*pi/N: pi-2*pi/N;
Omc= Om/Tc

xcn=cos(2000*pi*Ts*n);
XF = fft(xcn,D);
XF = fftshift(XF);

subplot(211);
plot(Omc,abs(XF));

axis([-21000*pi 21000*pi 0 600]);
title ('Espectro x[n]');
```



```
%Apartado 2.2
c)

YF = XF.*H;
plot(Omc,abs(YF));

axis([-21000*pi 21000*pi 0 35*10^5]);
title ('Espectro y[n]');
```

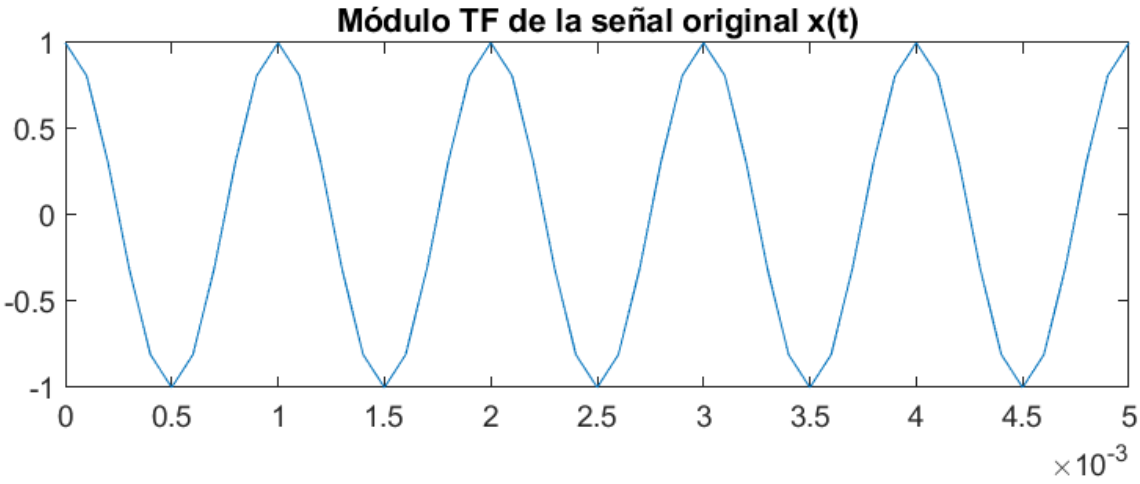


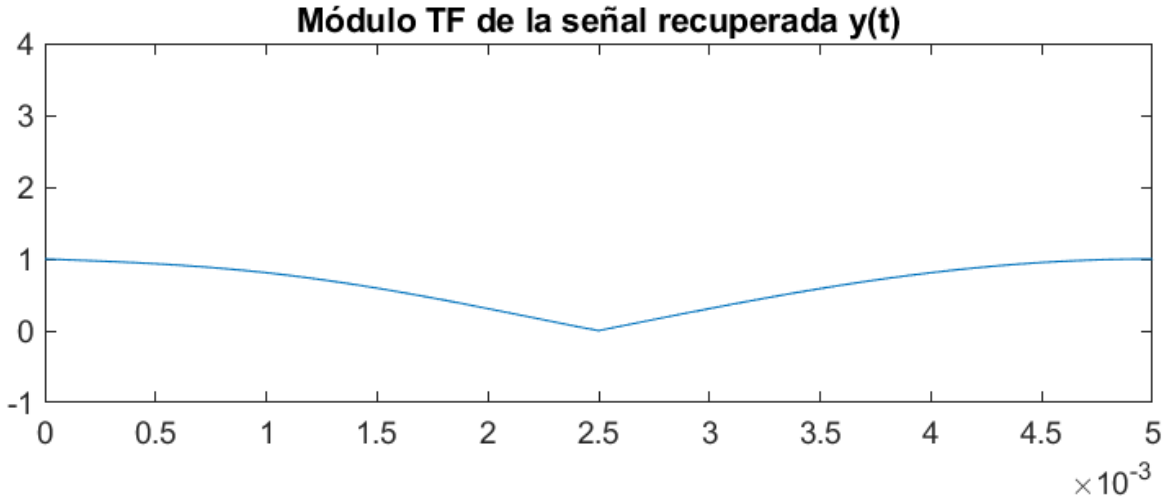

```
%Apartado 2.2
d)
Tc = 10E-5
t =0:Tc:(0.1-Tc);

YF_inv =ifft(YF); % ifft para calcular TF inversa
y =yr(1:length(YF_inv));

plot(t,y);
title ('y[n] como TF inversa');
```



2.3. Represente $x(t)$ e $y(t)$. Compare el resultado obtenido mediante la simulación con el resultado teórico esperado (obtenido calculando de forma teórica la derivada de la señal de entrada). ¿Coinciden?

```
xcn=cos(2000*pi*Ts*n);
XF = fft(xcn,D);
XF = fftshift(XF);

YF = XF.*H;

t =0:Tc:(0.1-Tc);
xcn=cos(2000*pi*Ts*n);

YF_inv =ifft(YF); % ifft para calcular TF inversa
y =yr(1:length(YF_inv));

xc=cos(2000*pi*t);

subplot (2,1,1);
plot(t,xc);
axis([0 0.005 -1 1]);
title ('Módulo TF de la señal original x(t)');

subplot (2,1,2);
plot(t,abs(y));
axis([0 0.005 -1 4]);
title ('Módulo TF de la señal recuperada y(t)');
```


Módulo TF de la señal original x(t)
1
0.5
0
-0.5
-1
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
×10-3

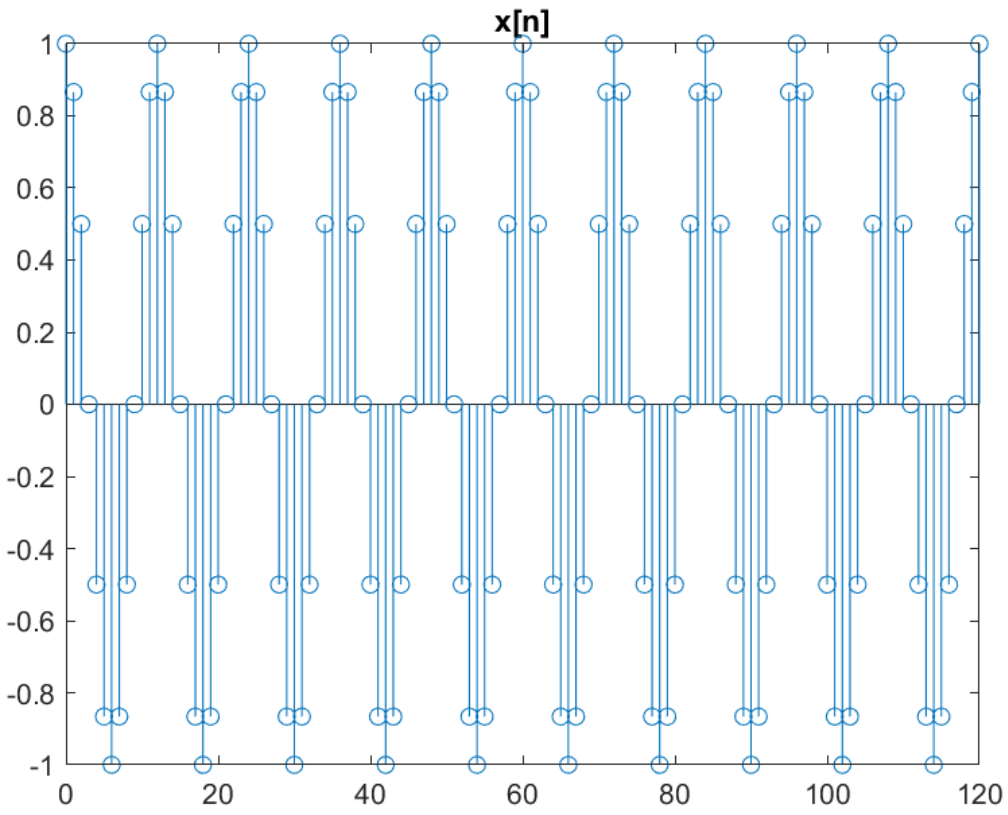

Módulo TF de la señal recuperada y(t)
4
3
2
1
0
-1
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
×10-3

3. Interpolación y Diezmado

En esta sección se revisa el cambio en la velocidad de muestreo sin salir del dominio discreto mediante técnicas de interpolación y diezmado.

3.1. Se ha muestreado una señal continua $x_c(t) = \cos(2000\pi t)$, con duración de 0,1 segundos, utilizando una frecuencia de muestreo $f_1 = 12000$ Hz, obteniéndose $x[n]$. Genere $x[n]$ en Matlab con un número de muestras tal que se cubra toda la longitud de $x_c(t)$.

```
Tc = 1E-5
t=0:Tc:0.1;
xc =cos(2000.*pi.*t);
f1=12000;
n=0:f1;
xn=cos(2000.*pi.*(1/f1).*n);
stem(n,xn);
axis([0 120 -1 1]);
title ('x[n]');
```



3.2. Se desea obtener una nueva secuencia $x_2[n]$, muestreada a una tasa de f_2 Hz, sin salir del dominio discreto. Encuentre los factores de interpolación y diezmado necesarios, y obtenga la señal remuestreada, para cada uno de los casos siguientes:

a) f_2 = 24 kHz.

b) f_2 = 6 kHz.

c) f_2 = 1 kHz.

d)) f_2 = 44 kHz.

Deberá obtener un resultado similar realizando el proceso de interpolación en dos pasos: primero, añadiendo ceros, y segundo, filtrando paso bajo. Para obtener los coeficientes del filtro FIR, h[n], deberá utilizar la función `fir1`.

Nota 1: Aunque el tema de diseño de filtros se verá más adelante, la utilización de fir1 es bastante sencilla:

```
h=fir1(Nh,1/P);
y=filter(h,1,x);
```

siendo P el factor de interpolacion y Nh el orden del filtro (par). Recuerde que el orden de un filtro FIR es el número de coeficientes menos uno.

Nota 2: En Matlab se puede realizar en una única instrucción el sobremuestreo y la adición de ceros. Piense cómo y hágalo.

Nota 3: Represente un intervalo de 10 ms de cada una de las señales anteriores, utilizando las muestras necesarias en cada caso. Tenga en cuenta que la frecuencia final es distinta en cada caso, por lo que el número de muestras correspondiente a 10 ms también será distinto.

Nota 4: Puede calcular los factores de diezmado e interpolación de manera automática utilizando la función `rat`, o mejor aún, hágalo numéricamente y compruebe que los resultados coinciden.

Nota 5: Para interpolar puede utilizar la función `interp`, mientras que para diezmar puede usar la función `decimate`. Cuando se requieran ambas operaciones, puede utilizar la función `resample`. Lo ideal es que este cambio de velocidad por un número no entero se haga en diferentes pasos. Primero, interpolar: se sobremuestrea la señal intercalando ceros y se filtra paso bajo con frecuencia de corte

$$\Omega_c = \frac{\pi}{P},$$

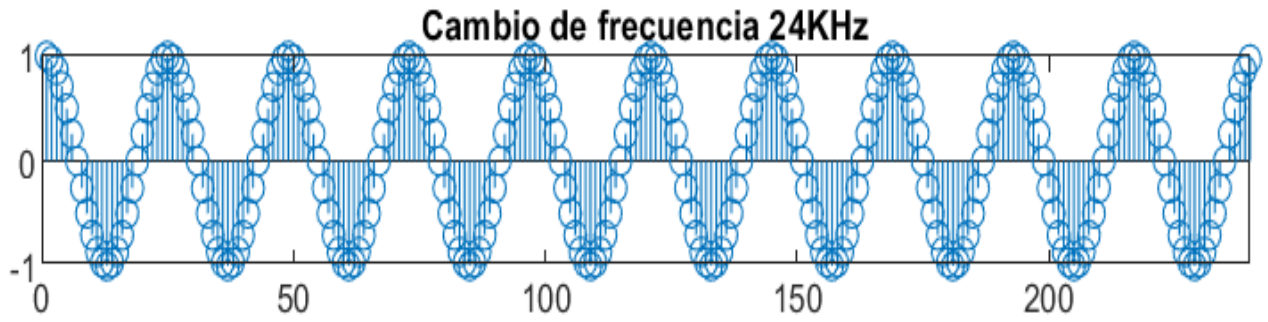
siendo P el factor de interpolación. Para diezmar, si no es necesario filtrar se diezma directamente descartando las muestras correspondientes. Pero si fuese necesario filtrar, se filtrar´ıa paso bajo con frecuencia de corte

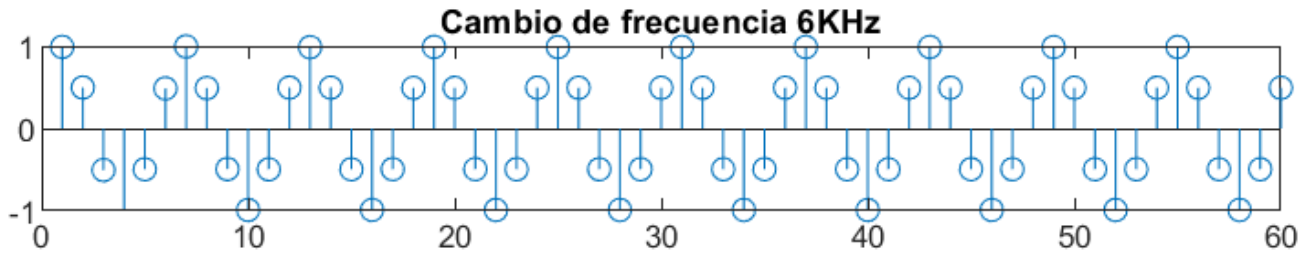
$$\Omega_c = \frac{\pi}{Q},$$

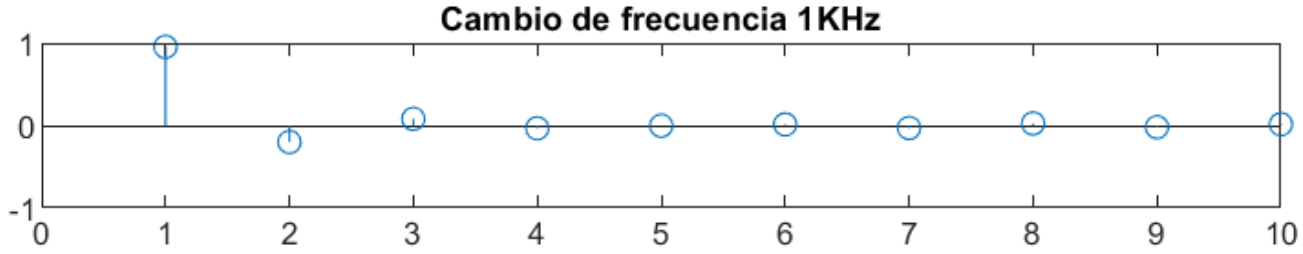
siendo Q el factor de diezmado.

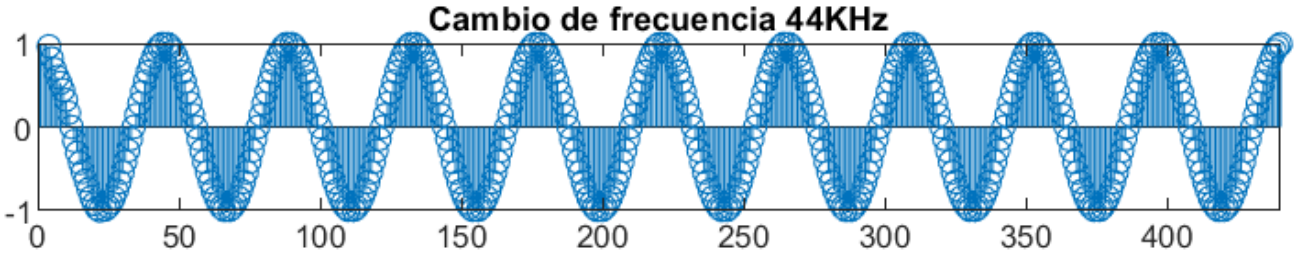
Nota 5: Para cambiar la frecuencia de muestreo por un factor no entero se debe de realizar en tres etapas: primero interpolando, segundo filtrando paso bajo (utilizando el filtro más restrictivo de los dos, es decir, el de mayor valor de P o Q), y finalmente diezmando.

```
P_a=2;
x_a=interp(xn,Pa); % interpolar
Qb=2;
x_b=decimate(xn,Qb); % diezmar
Qc=12;
x_c=decimate(xn,Qc);
Pd=11;
Qd=3;
x_d=resample(xn,Pd,Qd); % interpolación + diezmado

subplot (4,1,1);
stem(x_a);
axis([0 240 -1 1]); % 10ms de la señal
title ('Cambio de frecuencia 24KHz');

subplot (4,1,2);
stem(x_b);
axis([0 60 -1 1]); % 10ms de la señal
title ('Cambio de frecuencia 6KHz');

subplot (4,1,3);
stem(x_c);
axis([0 10 -1 1]); % 10ms de la señal
title ('Cambio de frecuencia 1KHz');

subplot (4,1,4);
stem(x_d);
axis([0 40*11 -1 1]); % 10ms de la señal
title ('Cambio de frecuencia 44KHz');
```


Cambio de frecuencia 24KHz
1
0
-1
0
50
100
150
200


Cambio de frecuencia 6KHz
1
0
-1
0
10
20
30
40
50
60


Cambio de frecuencia 1KHz
1
0
-1
0
1
2
3
4
5
6
7
8
9
10


Cambio de frecuencia 44KHz
1
0
-1
0
50
100
150
200
250
300
350
400

3.3. Escuche, mediante el uso de la función sound, tanto la señal $x[n]$ como cada una de las señales $x_2[n]$ generadas para las distintas frecuencias f_2 en el ejercicio anterior. ¿Se escuchan todas las señales igual? ¿Por qué?

Nota 6: Tenga en cuenta que cada señal se corresponde con una frecuencia de muestreo diferente al utilizar la función sound.

```
sound(xn, 12001);
sound(x_a, 24002);
sound(x_b, 6001);
sound(x_c, 1001);
sound(x_d, 44004);
```

No se escuchan iguales ya que sus frecuencias han sido modificadas

3.4. Convierta una señal de voz muestreada a 8 kHz en la misma señal, pero muestreada a 44.1 kHz (frecuencia de muestreo utilizada en Compact Disc). Visualice su espectro antes, durante y después del proceso de conversión. Utilice tres etapas de interpolación, filtro paso bajo y diezmado en cascada. Puede utilizar la función resample de Matlab, aunque no tendrá acceso al vector al que se le han añadido ceros antes de filtrar.

Nota: En Moodle dispone de dos ficheros de voz en formato Matlab, voz1_8khz.mat y voz2_8khz.mat.