Informe Octave PSTD20

June 28, 2020

1 Introducción

En éste documento se realizan los prácticos de laboratorio propuestos en la asignatura de posgrado Procesamiento de señales en tiempo discreto.

En ésta versión usamos el código de ejemplo provisto por el docente, con algunas modificaciones y agregados para hacerlo funcionar en **Octave** y embebido en Jupyter-Lab. También es posible usar MATLAB embebido en Jupyter-Lab. De todos modos tódo el código aquí presentado con pequeñas modificaciones es **compatible con MATLAB**.

De ésta manera se puede presentar el material teórico relevante a cada tema tratado, junto con el código separado en etapas para generar las imágenes a analizar.

2 Laboratorio N°1

Un filtro muy usado en comunicaciones digitales es el filtro de caída cosenoidal definido por :

$$g(t) = \text{sinc}(t/T)\frac{\cos(\pi\beta t/T)}{1-4\beta^2t^2/T^2}$$

Siendo T el tiempo entre símbolos y β el factor de roll-off:

$$0 \leq \beta \leq 1$$

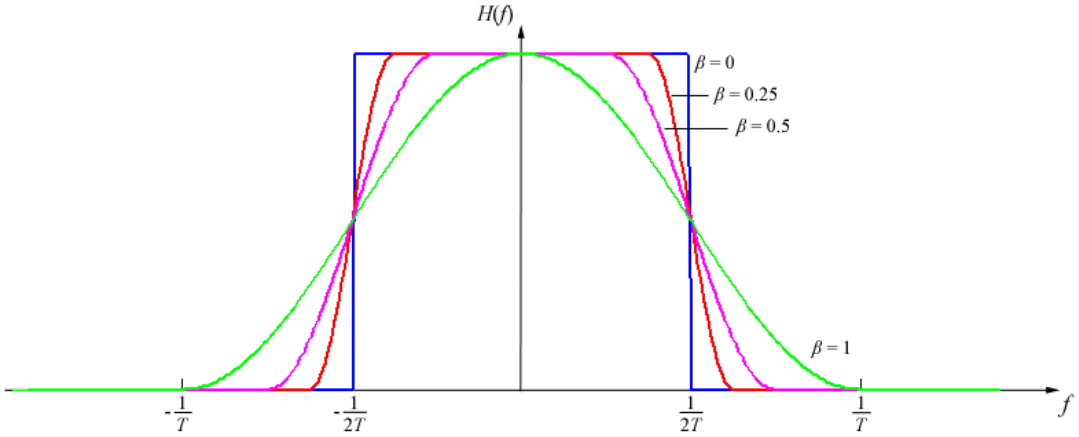
- **Generar** $g[n] = g(nT_s)$ con $1/T_s$ siendo la frecuencia de sobremuestreo ($T/T_s = M$ con M entero)
- Utilizar la autofunción $e^{j2\pi fT_sn}$, para **obtener por simulación la respuesta** $|G(e^{j2\pi fT_sn})|$, en función de $2\pi fT_s$ con $|f| < 1/T$, para $\beta = 1$.
- Sea $x[n] = \sum_{k=0}^{L-1} a[k]\delta[n-kM]$ con $a[k] \in \{\pm 1\}$, un proceso aleatorio blanco con símbolos equiprobables. **Determinar por simulación la señal de salida y generar el diagrama de ojo** . Repetir para $\beta = 0.1$ y $\beta = 0.5$

2.1 Filtro de caída cosenoidal

Respuesta en frecuencia del coseno realzado:

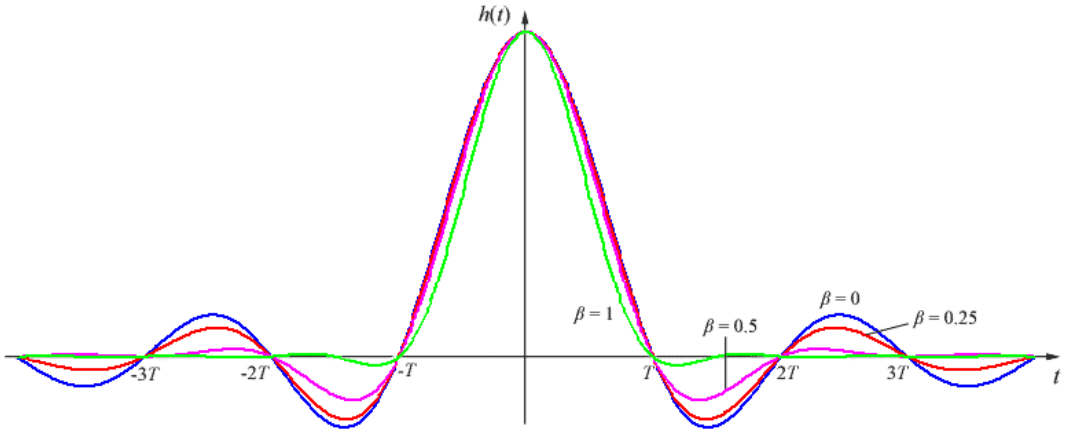
$$|H(f)| = \begin{cases} 1.0, & |f| \leq \frac{1-\beta}{2T} \\ \frac{1}{2}\left[1 + \cos\left(\frac{\pi T}{\beta}\left[|f| - \frac{1-\beta}{2T}\right]\right)\right], & \frac{1-\beta}{2T} < |f| \leq \frac{1+\beta}{2T} \\ 0, & \text{resto} \end{cases}$$

$$0 \leq \beta \leq 1$$



Respuesta al impulso:

$$g(t) = \text{sinc}(t/T)\frac{\cos(\pi\beta t/T)}{1 - 4\beta^2 t^2/T^2}$$



Consideraciones y trade-off

Aquí puede verse que a medida que $\beta \approx 0$ la respuesta en frecuencia se parece a la de un filtro pasa-bajos ideal, siendo ésto una ventaja desde el punto de vista del ancho de banda utilizado. La desventaja de éste caso es que, como se ve en la respuesta al impulso hacen falta muchos mas coeficientes para describir el filtro ya que las colas decaen mucho mas lentamente. Otra desventaja que veremos de éste caso es que, como se podrá ver en los diagramas de ojo, la sensibilidad a la

fase de muestreo es mayor (existe una mayor probabilidad de error ante pequeñas variaciones del instante de muestreo).

En cambio a medida que $\beta \approx 1$ aumenta el exceso de ancho de banda respecto al filtro pasabajos ideal , pero se tiene la ventaja de que son necesarios menos coeficientes para representar el filtro ya que las colas caen mucho mas rápidamente. Otra ventaja de aumentar el factor de roll-off es que, como se verá en los diagramas de ojo, hay una menor sensibilidad a la fase de muestreo, que se nota en el gráfico en la menor probabilidad de error ante pequeñas variaciones del instante de muestreo.

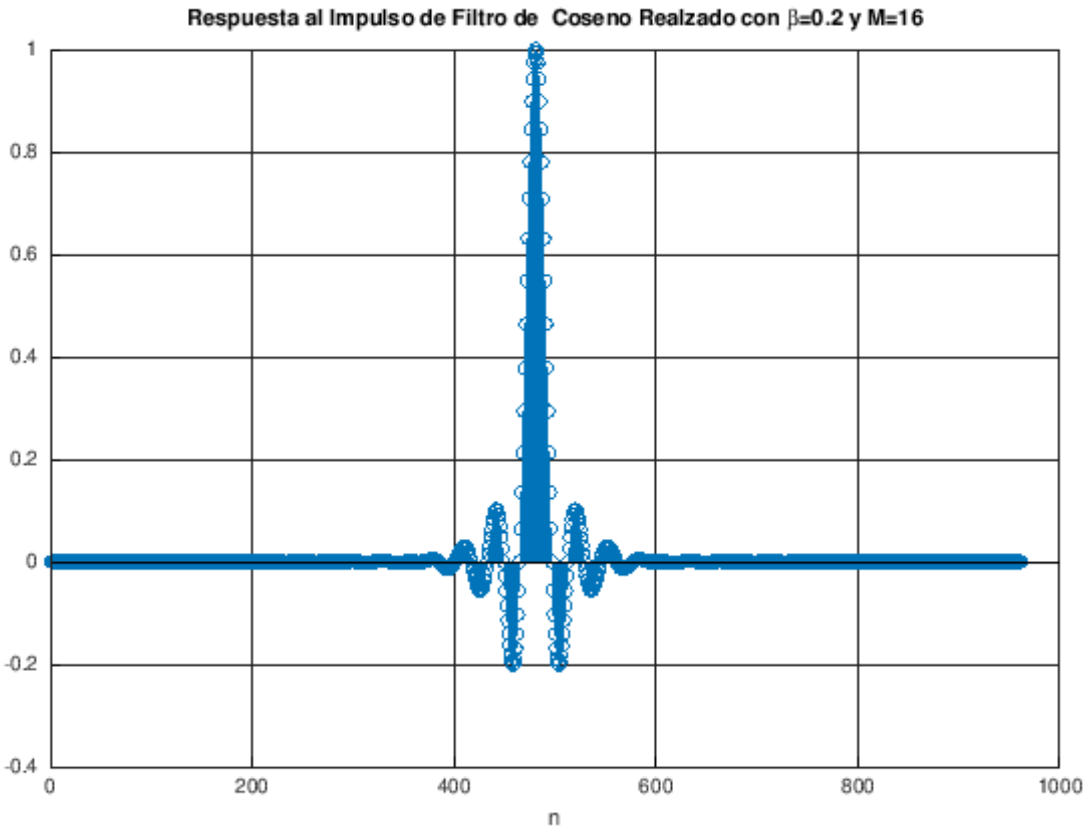
2.2 Etapa de generación de respuesta al impulso

En éste apartado mostramos la respuesta al impulso en el dominio del tiempo-discreto .

En el práctico de laboratorio adjunto con python, se puede probar el efecto de modificar β interactivamente

2.2.1 Filtro de caída cosenoidal con $\beta = 0.2$

[1]:
```
pkg load control
pkg load signal
pkg load communications
warning off
%==========================================
% Generacion de la Respuesta al Impulso
%==========================================
fB = 32e9;         % Velocidad de simbolos (baud rate)
T = 1/fB; % Tiempo entre simbolos
M = 16;  %Factor de sobremuestreo
fs = fB*M;         % Sample rate


%Definimos parámetros del filtro

beta = .200001; %Factor de roll-off
L = 30;  % 2*L*M+1 es el largo del filtro sobremuestreado
n_delay_filter = L*M; %Retardo del filtro
t = [-L:1/M:L]*T;

gn = sinc(t/T).*cos(pi*beta*t/T)./(1-4*beta^2*t.^2/T^2);


h = stem(gn);
title('Respuesta al Impulso de Filtro de  Coseno Realzado con \beta=0.2 y␣
 ↪M=16');
xlabel('n');
grid
```



2.2.2 Filtro pasa-bajos ideal - Filtro de caída cosenoidal con $\beta = 0$

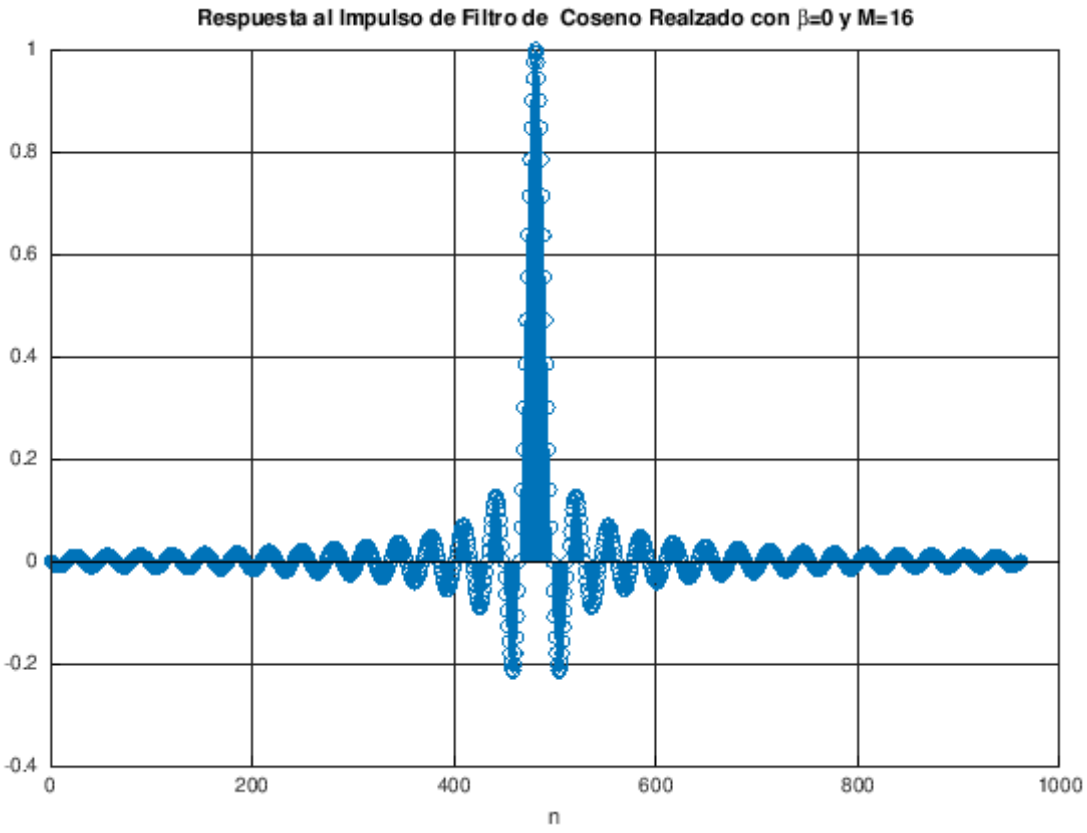
Notamos que para $\beta = 0$ el filtro se convierte en :

$$g(t) = sinc(\frac{t}{T})$$

Siendo ésta la respuesta al impulso de un filtro pasa-bajos ideal. Su ancho de banda es mínimo ya que desaparece la parte de caída cosenoidal.

[2]:
```
%Definimos parámetros del filtro
M = 16;  %Factor de sobremuestreo
beta = 0; %Factor de roll-off
L = 30;  % 2*L*M+1 es el largo del filtro sobremuestreado
t = [-L:1/M:L]*T;

gn = sinc(t/T).*cos(pi*beta*t/T)./(1-4*beta^2*t.^2/T^2);


h = stem(gn);
```

```
title('Respuesta al Impulso de Filtro de  Coseno Realzado con \beta=0 y M=16');
xlabel('n');
grid
```



2.3 Respuesta en Frecuencia

Se nos pide utilizar la autofunción $e^{j2\pi fT_s n}$, para **obtener por simulación la respuesta en frecuencia**:

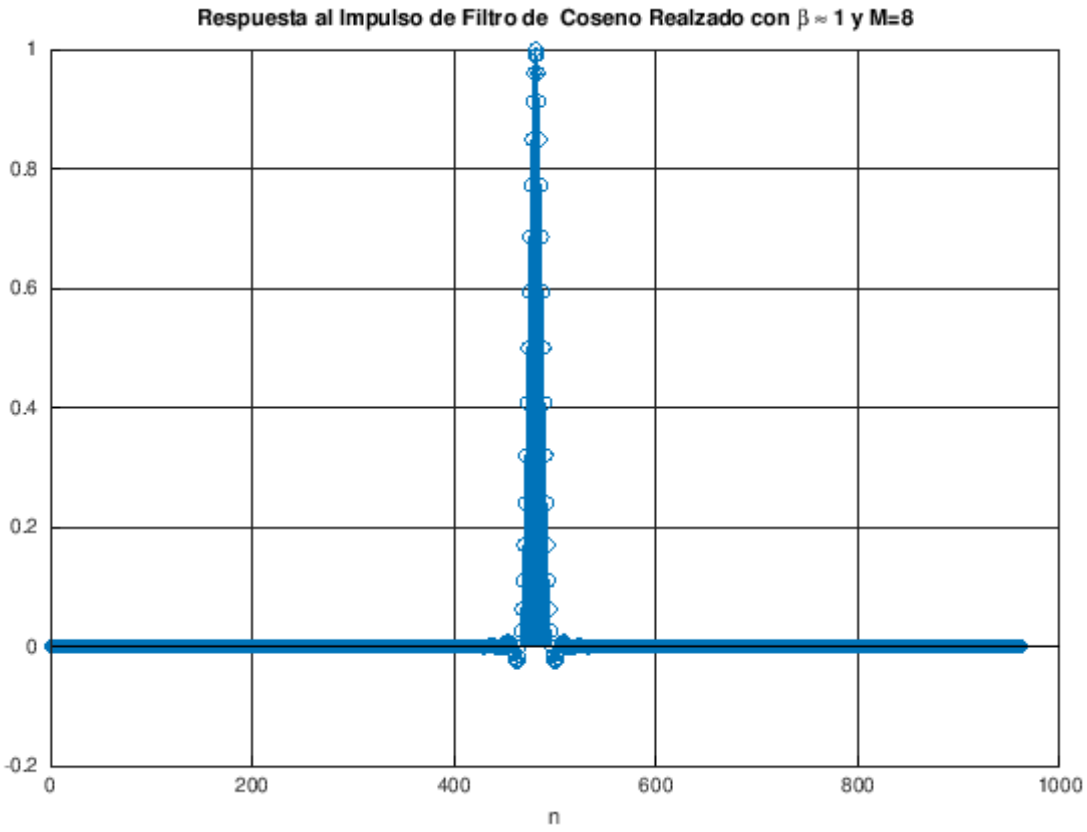
$$|G(e^{j2\pi fT_s n})|$$

En función de $2\pi fT_s$ con $|f| < 1/T$, para $\beta = 1$

2.3.1 Filtro de caída cosenoidal con $\beta \approx 1$

[3]:
```
%Definimos parámetros del filtro
M = 16;  %Factor de sobremuestreo
beta = 0.9999991; %Factor de roll-off
```

```
L = 30;  % 2*L*M+1 es el largo del filtro sobremuestreado
t = [-L:1/M:L]*T;

gn = sinc(t/T).*cos(pi*beta*t/T)./(1-4*beta^2*t.^2/T^2);


h = stem(gn);
title('Respuesta al Impulso de Filtro de  Coseno Realzado con \beta \approx 1 y␣
 ↪M=8');
xlabel('n');
grid
```



2.3.2 Cálculo de la Respuesta en Frecuencia con $\beta \approx 1$

Aquí definimos la señal de entrada x_n como una autofunción :

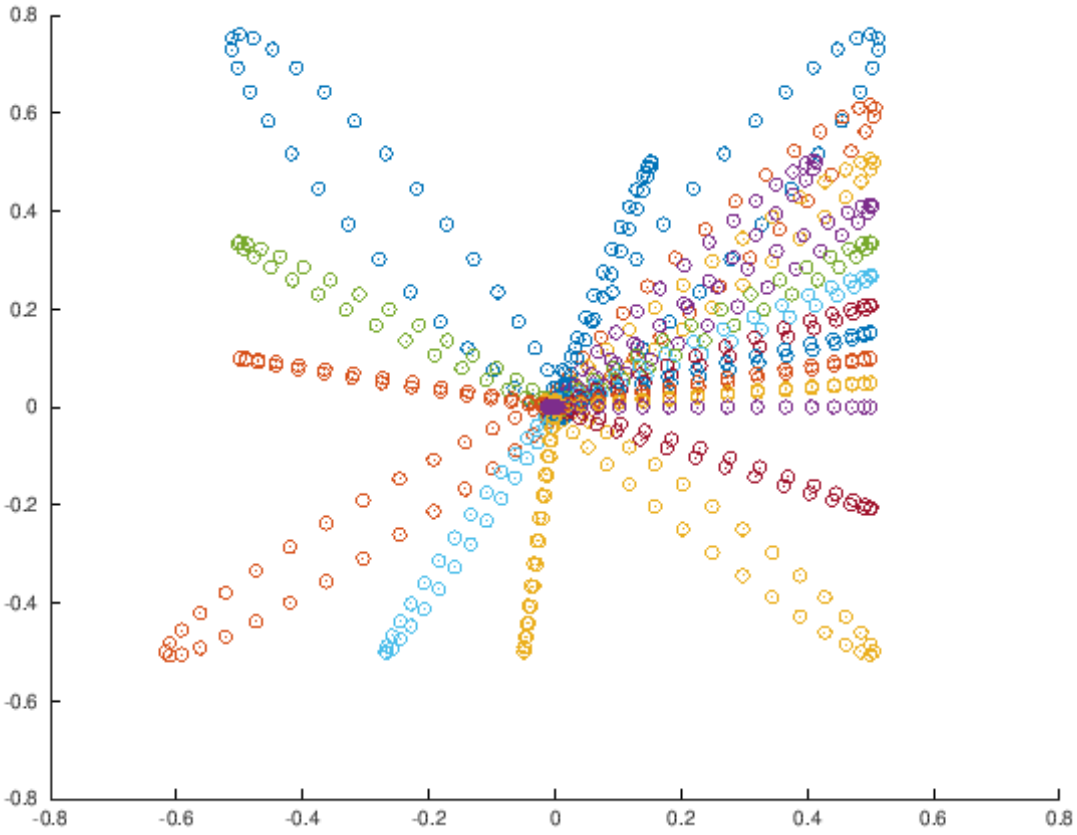
$$x[n] = e^{j2\pi f T_s n}$$

Y hacemos un barrido en frecuencia con un paso de $1/256$

Los primer y segundo plots circulares muestran el resultado del barrido en el plano complejo(cada color corresponde a una frecuencia distinta). El primer plot circular hace un barrido con un paso mayor (1/16) al segundo (1/256), notándose mejor la relación amplitud fase para cada frecuencia. Si bien éstos plots circulares no estaban incluídos en el programa compartido por el docente, se agregan aquí por ser de interés en la propuesta de investigación del estudiante, relacionada con el estudio de entrenadores de redes neuronales para estimación de series temporales multi-variadas .

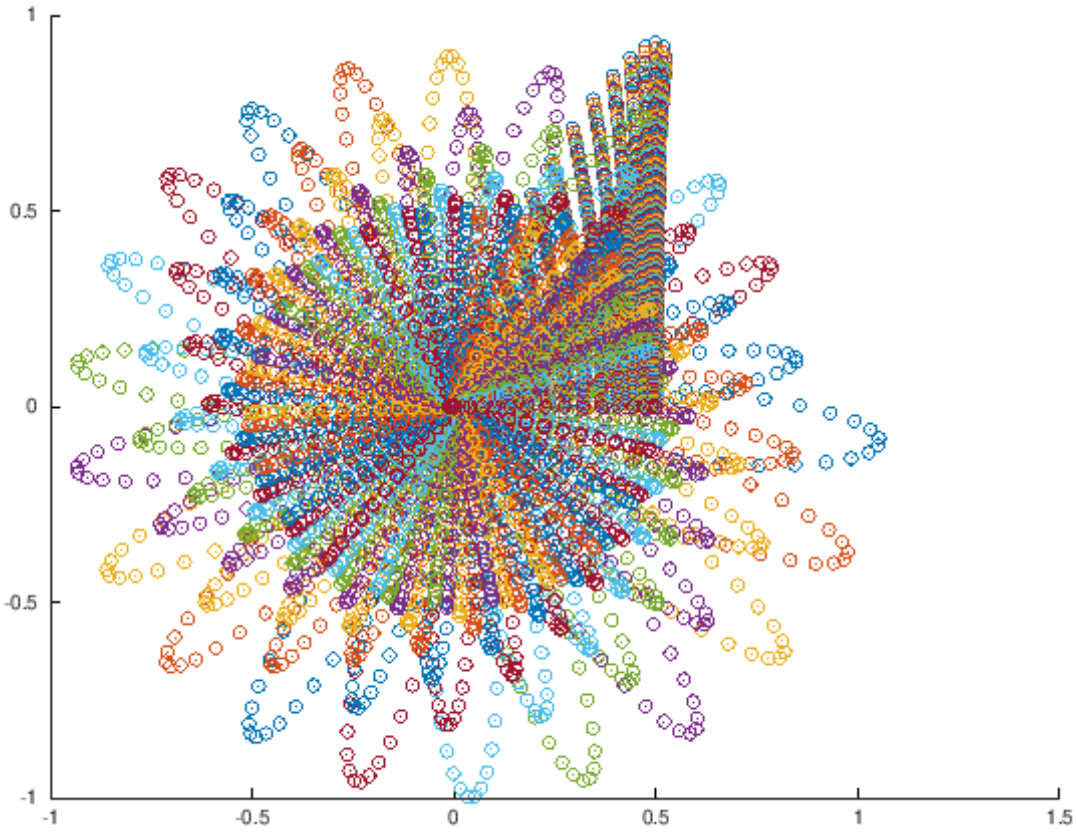
El resto de los plots son los tradicionales mostrando el resultado del mismo barrido en módulo y fase .

[4]:
```
Omega = [0:1/2^4:1]*pi;
N = 1000;
n = [0:N];
index = 1;
figure();

for omega=Omega
    xn = exp(j*omega*n);
    yn = conv(xn,gn);
    if (omega>1)
        plot(real(yn),imag(yn),'o'); % Plot polar de respuesta
    end
    hold on
    H_Mag1(index) = abs(yn(N/2));
    H_Fase1(index) = angle(yn(N/2)*conj(xn(N/2-n_delay_filter)));
    index = index+1;
end
```

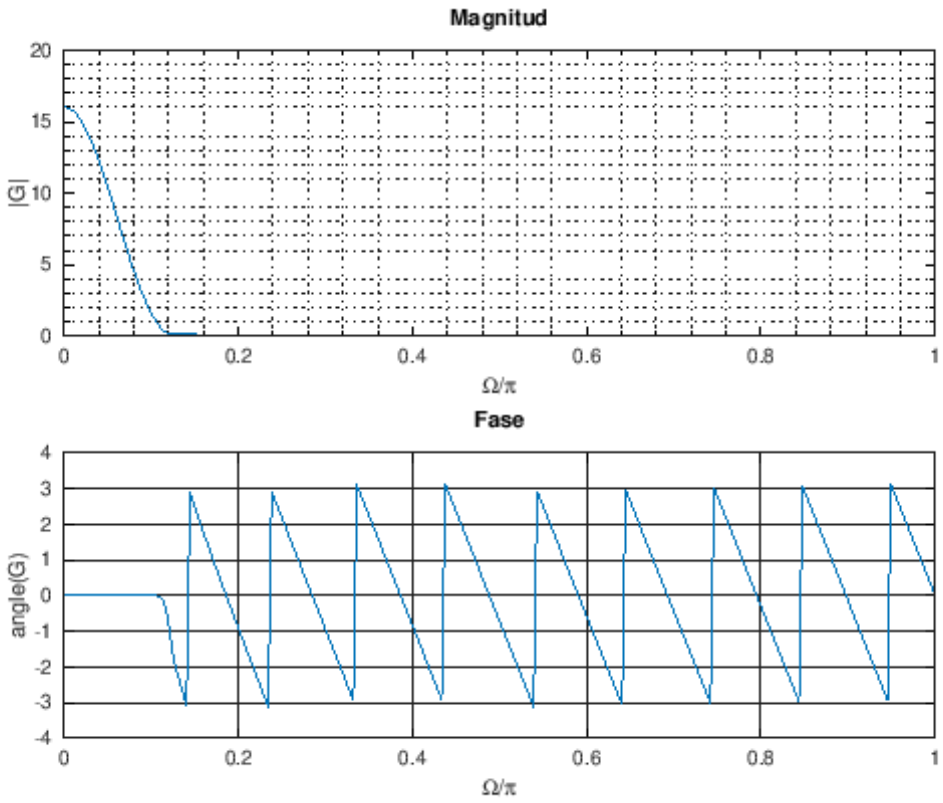


[5]:
```
%==========================================
% Calculo de la Respuesta en Frecuencia
%==========================================
Omega = [0:1/2^8:1]*pi;
N = 1000;
n = [0:N];
index = 1;
figure();

for omega=Omega
    xn = exp(j*omega*n);
    yn = conv(xn,gn);
    if (omega>1)
        plot(real(yn),imag(yn),'o');
    end
    hold on
    H_Mag1(index) = abs(yn(N/2));
    H_Fase1(index) = angle(yn(N/2)*conj(xn(N/2-n_delay_filter)));
    index = index+1;
end
```

```
hold off
```



[6]:
```
subplot(2,1,1)
h=plot(Omega/pi,H_Mag1);
title('Magnitud');
ylabel('|G|')
xlabel('\Omega/\pi');
grid minor
subplot(2,1,2)
h=plot(Omega/pi,H_Fase1);
title('Fase');
ylabel('angle(G)')
xlabel('\Omega/\pi');
grid
```

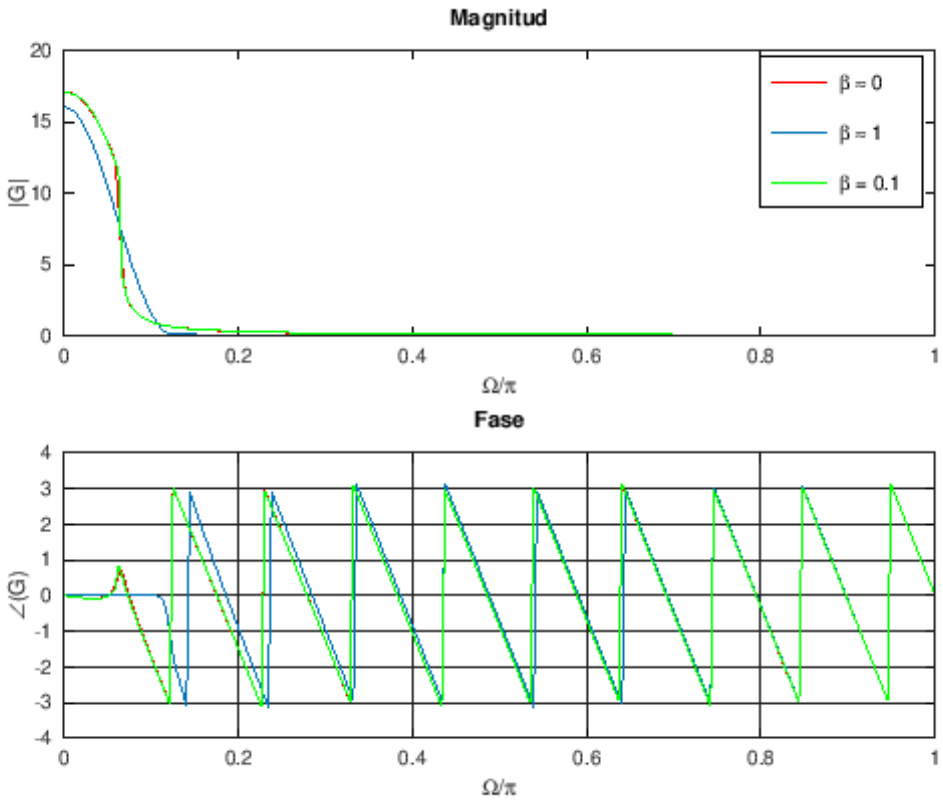


2.3.3 Comparación de la Respuesta en Frecuencia con $\beta \approx 1$ y $\beta \approx 0$

Aquí puede verse el exceso de ancho de banda en el plot en azul, correspondiente al gráfico anterior con $\beta \approx 1$ y el menor ancho de banda en el plot en rojo correspondiente al plot del $\beta = 0.1$,y en verde se puede ver $\beta \approx 0$. Nótese que entre el plot en rojo y el plot en verde no hay mucha diferencia.

[7]:
```
%Definimos parámetros del filtro
M = 16;  %Factor de sobremuestreo
beta = 0.1001; %Factor de roll-off
L = 30;  % 2*L*M+1 es el largo del filtro sobremuestreado
t = [-L:1/M:L]*T;

gn = sinc(t/T).*cos(pi*beta*t/T)./(1-4*beta^2*t.^2/T^2);
Omega = [0:1/2^8:1]*pi;
N = 1000;
n = [0:N];
index = 1;
for omega=Omega
    xn = exp(j*omega*n);
```

```
    yn = conv(xn,gn);
    H_Mag2(index) = abs(yn(N/2));
    H_Fase2(index) = angle(yn(N/2)*conj(xn(N/2-n_delay_filter)));
    index = index+1;
end

beta = 0.0000000000000001001; %Factor de roll-off
gn = sinc(t/T).*cos(pi*beta*t/T)./(1-4*beta^2*t.^2/T^2);

index = 1;
for omega=Omega
    xn = exp(j*omega*n);
    yn = conv(xn,gn);
    H_Mag3(index) = abs(yn(N/2));
    H_Fase3(index) = angle(yn(N/2)*conj(xn(N/2-n_delay_filter)));
    index = index+1;
end

subplot(2,1,1)
h=plot(Omega/pi,H_Mag2,color='r');
hold on
h=plot(Omega/pi,H_Mag1);
h=plot(Omega/pi,H_Mag3,color='g');
title('Magnitud');
ylabel('|G|')
xlabel('\Omega/\pi');
legend('\beta \approx 0','\beta \approx 1','\beta = 0.1');
hold off

subplot(2,1,2)
h=plot(Omega/pi,H_Fase2,color='r');
hold on
h=plot(Omega/pi,H_Fase1);
h=plot(Omega/pi,H_Fase3,color='g');
title('Fase');
ylabel('\angle(G)')
xlabel('\Omega/\pi');
grid
hold off
```



2.4 Simulación de señal de salida y Consideraciones de Comunicaciones

Determinar por simulación la señal de salida y generar el diagrama de ojo . Repetir para $\beta = 0.1$ y $\beta = 0.5$

Sensibilidad a la fase de muestreo (se ve en el diagrama de ojo q para beta 1 es menos sensible que para beta 0.1) (esa sensibilidad a la fase de muestreo significa que pequeñas modificaciones en ese instante de muestreo generarían mayor o menor ISI (interferencia entre simbolos).

Un beta lo mas chico posible es lo q se busca en comunicaciones ya que mientras mas chico es el beta menor es el exceso de ancho de banda (comparado con un el rectangulo q es la TF del sinc (beta=0) , y el coste se paga en que se necesitan mas coeficientes para representar el filtro ya que para beta mas chico las colas decaen mas lentamente)

2.4.1 Generación de símbolos BPSK

Sea:

$$x[n] = \sum_{k=0}^{L-1} a[k]\delta[n - kM] \text{ con } a[k] \in \{\pm 1\}$$

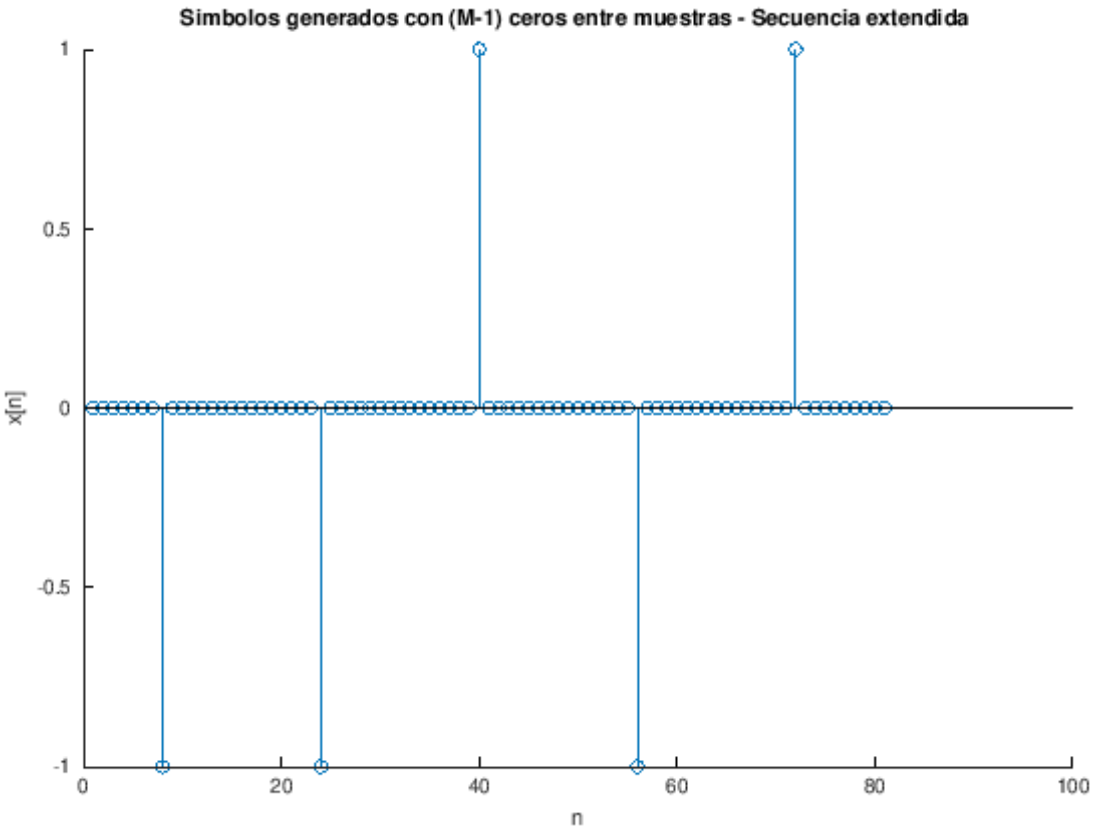
un **proceso aleatorio blanco** con símbolos equiprobables

Vamos a generar 1000 símbolos con (M-1) ceros entre muestras utilizando RNG interno de octave.

[8]:
```
%==========================================
% Generacion Simbolos
%==========================================
n_symbols = 1000;
PAM_K=2 ; % 2 -> BPSK ;   4-> PAM4

ak = 2*randi([0,PAM_K-1],1,n_symbols)-(PAM_K-1);%PAMK

xn = zeros(1,n_symbols*M);

xn(1:M:end) = ak;

n_tabs_xn=length(xn)
```

```
n_tabs_xn =  16000
```

[9]:
```
h = stem(xn(10:10+M*5));
title('Simbolos generados con (M-1) ceros entre muestras - Secuencia extendida␣
 ↪');
ylabel('x[n]')
xlabel('n');
```



2.4.2 Señal Transmitida

La señal transmitida es la convolución de los símbolos generados con la respuesta del filtro

$$s[n] = \sum_{k=-\infty}^{\infty} x[k]g[n-k] = x[n] * g[n]$$

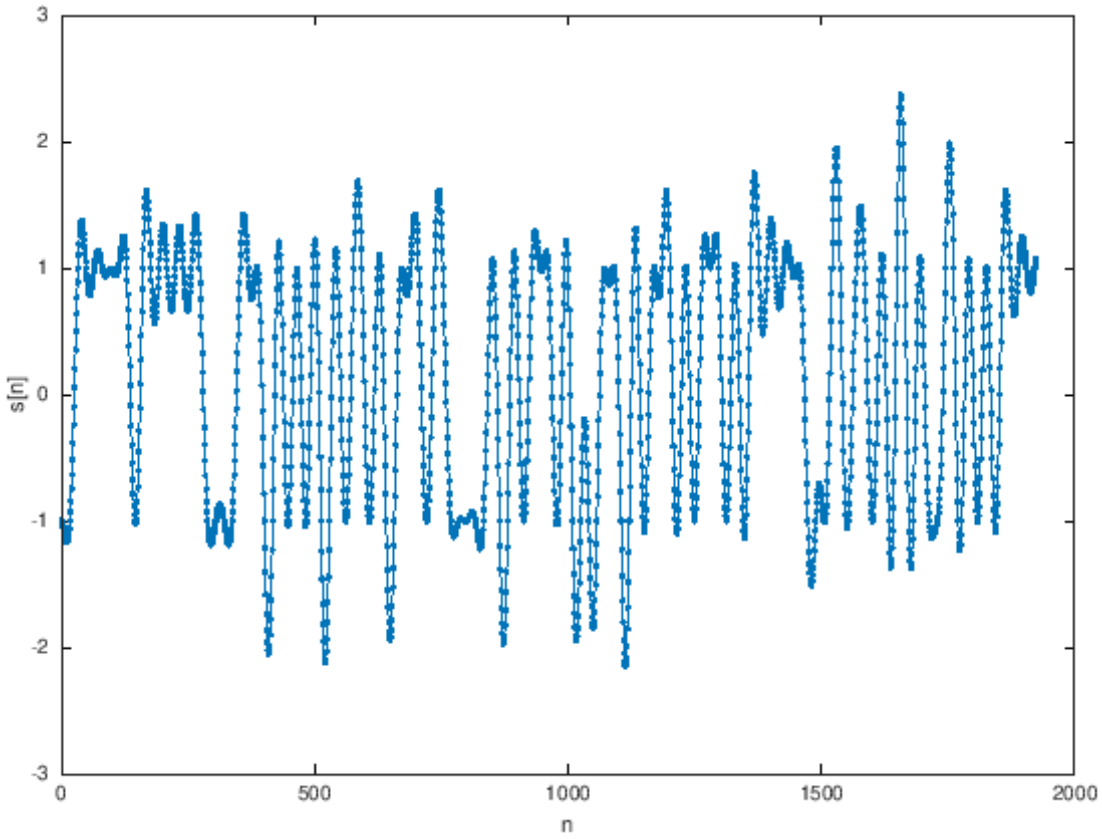
[10]:
```
%=========================================
% Señal Transmitida
%=========================================
n_tabs_gn=length(gn)
```

```
n_tabs_gn =  961
```

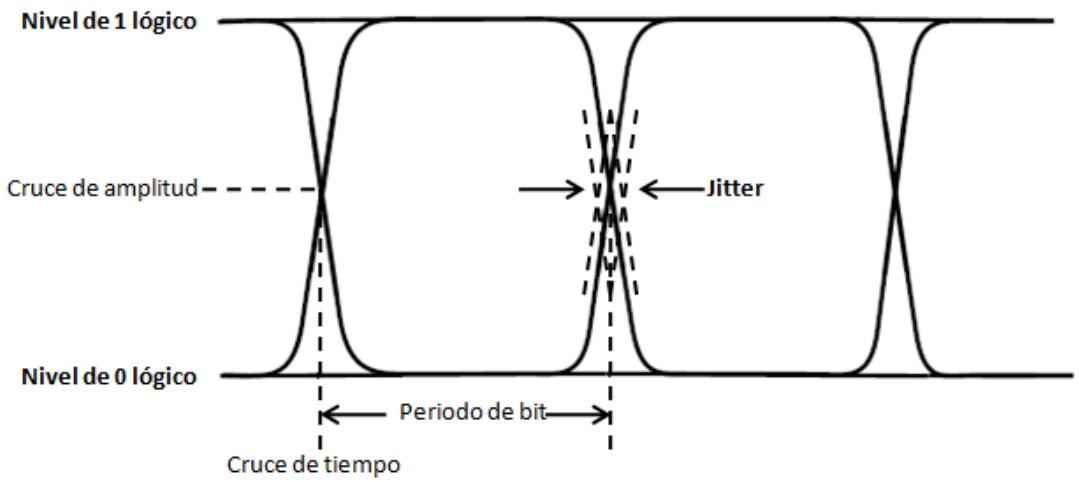
[11]:
```
sn = conv(xn,gn);

n_tabs_sn = length(sn)
```

```
n_tabs_sn =  16960
```

[12]:
```
h = plot(sn((2*L*M+1):(2*L*M+1)*3),'.-');
ylabel('s[n]')
xlabel('n');
```



2.4.3 Diagrama de ojo



Referencia para entender el diagrama de ojo y poder extraer inforemación valiosa de él :

https://rintintin.colorado.edu/~gifford/5830-AWL/Anritsu%20Eye%20Diagram.pdf

Diagrama de Ojo para $\beta \approx 1$ Desde el punto de vista de inmunidad a la fase de muestreo éste es el caso deseable, ya que pequeños desplazamientos en el instante de muestreo tienen la menor probabilidad de generar Interferencia entre símbolos. La desventaja es un mayor exceso de ancho de banda

[13]:
```
%==========================================
% Generacion de Diagrama Ojo
%==========================================

%Definimos parámetros del filtro
M = 16;  %Factor de sobremuestreo
beta = 0.99951; %Factor de roll-off
L = 30;  % 2*L*M+1 es el largo del filtro sobremuestreado
t = [-L:1/M:L]*T;

gn = sinc(t/T).*cos(pi*beta*t/T)./(1-4*beta^2*t.^2/T^2);

sn = conv(xn,gn);
d = M/2+1; %Delay para centrar el ojo
for m = 2*L+1:n_symbols-(2*L+1)
    sn_p = sn(m*M+d:m*M+d+M);
    plot([-M/2:1:M/2],sn_p,color='b')
    hold on
end
title('Diagrama de ojo con \beta \approx 1')
grid
```

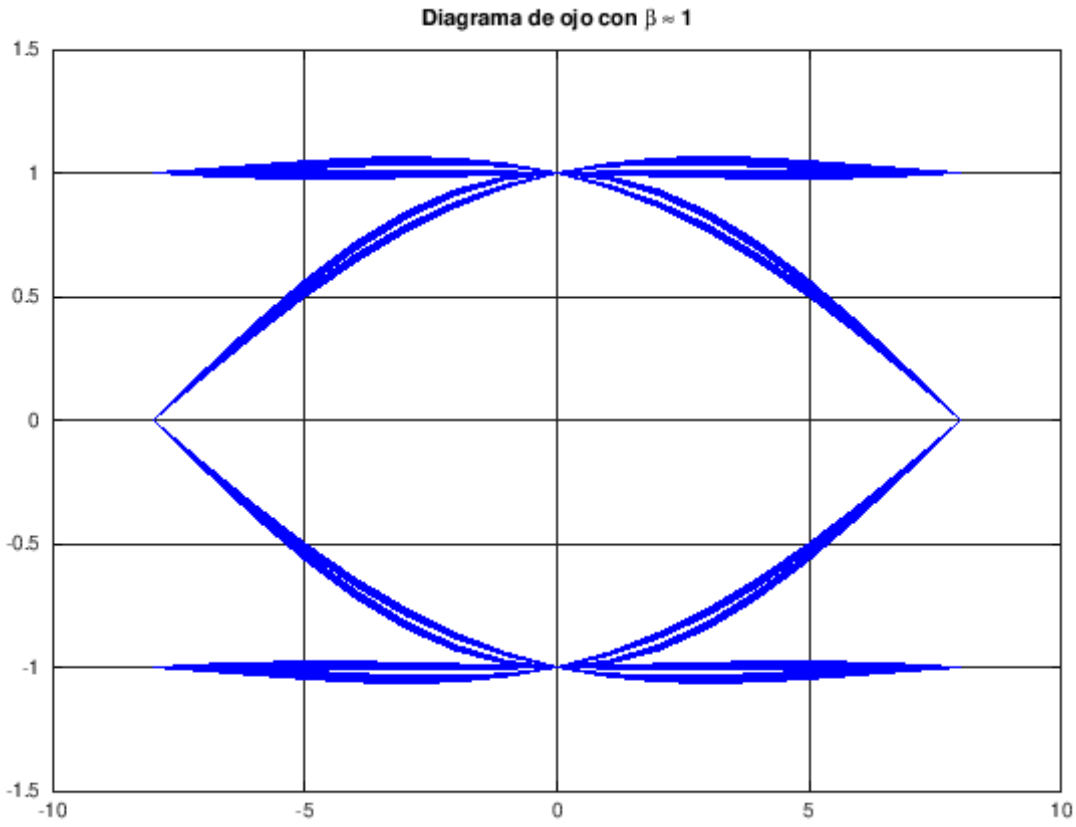


Diagrama de Ojo para $\beta = 0.1$ En este caso cualquier pequeña desviación de la fase de muestreo podría causar interferencia entre símbolos. Tenemos la ventaja de que el ancho de banda es mínimo pero se paga con una mayor posibilidad de jitter.

[14]:
```
%Definimos parámetros del filtro
M = 16;  %Factor de sobremuestreo
beta = 0.10001; %Factor de roll-off
L = 30;  % 2*L*M+1 es el largo del filtro sobremuestreado
t = [-L:1/M:L]*T;

gn = sinc(t/T).*cos(pi*beta*t/T)./(1-4*beta^2*t.^2/T^2);

sn = conv(xn,gn);

d = M/2 + 1; %Delay para centrar el ojo
for m = 2*L+1:n_symbols-(2*L+1)
    sn_p = sn(m*M+d:m*M+d+M);
    plot([-M/2:1:M/2],sn_p,color='b')
    hold on
end
```

```
title('Diagrama de ojo con \beta =0.1')
grid
```

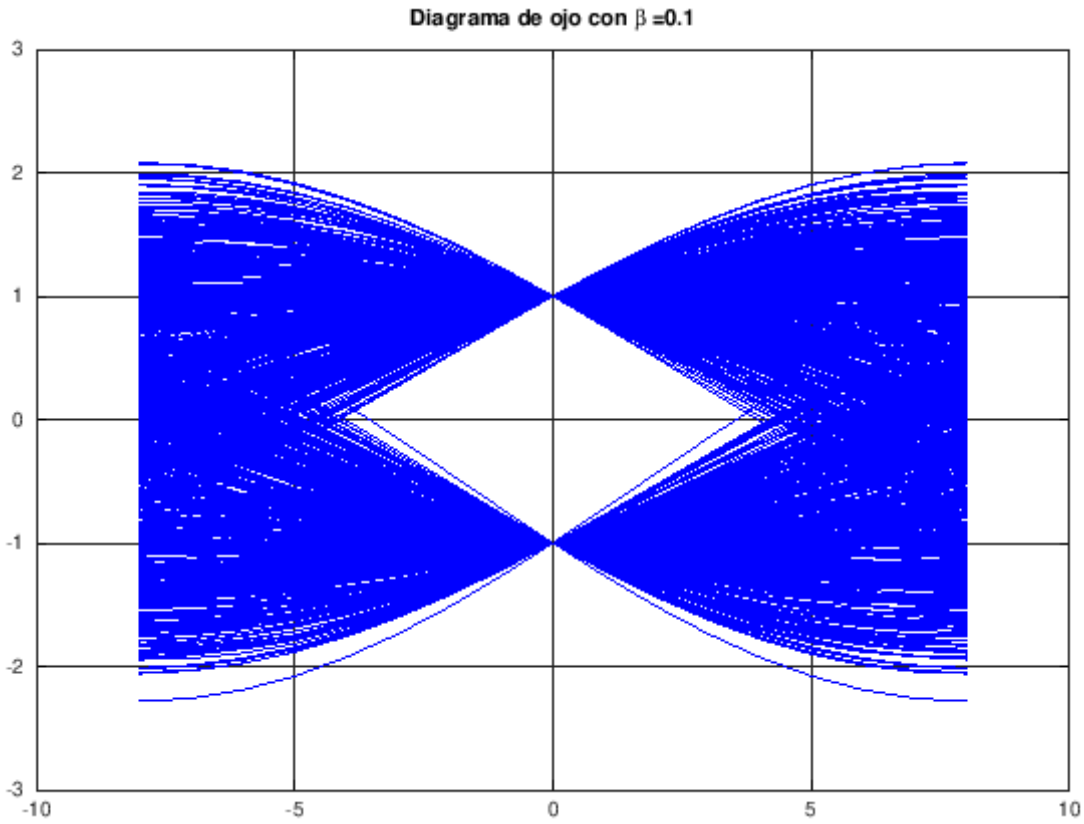
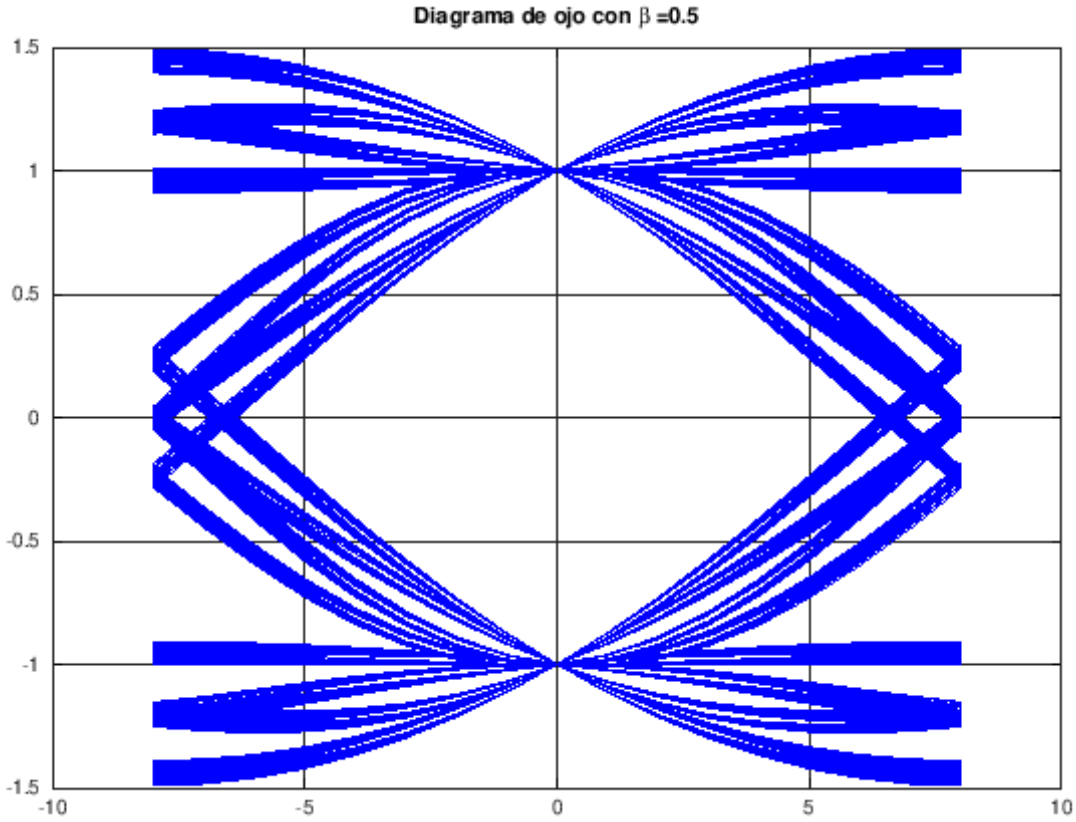


Diagrama de Ojo para $\beta = 0.5$ En este caso se nota un aumento en la apertura horizontal del ojo, lo que tiene un costo asociado al aumento del ancho de banda del filtro.

[15]:
```
%Definimos parámetros del filtro
M = 16;  %Factor de sobremuestreo
beta = 0.50001; %Factor de roll-off
L = 30;  % 2*L*M+1 es el largo del filtro sobremuestreado
t = [-L:1/M:L]*T;

gn = sinc(t/T).*cos(pi*beta*t/T)./(1-4*beta^2*t.^2/T^2);

sn = conv(xn,gn);

d = M/2 + 1; %Delay para centrar el ojo
for m = 2*L+1:n_symbols-(2*L+1)
    sn_p = sn(m*M+d:m*M+d+M);
    plot([-M/2:1:M/2],sn_p,color='b')
```

```
    hold on
end
title('Diagrama de ojo con \beta =0.5')
grid
```

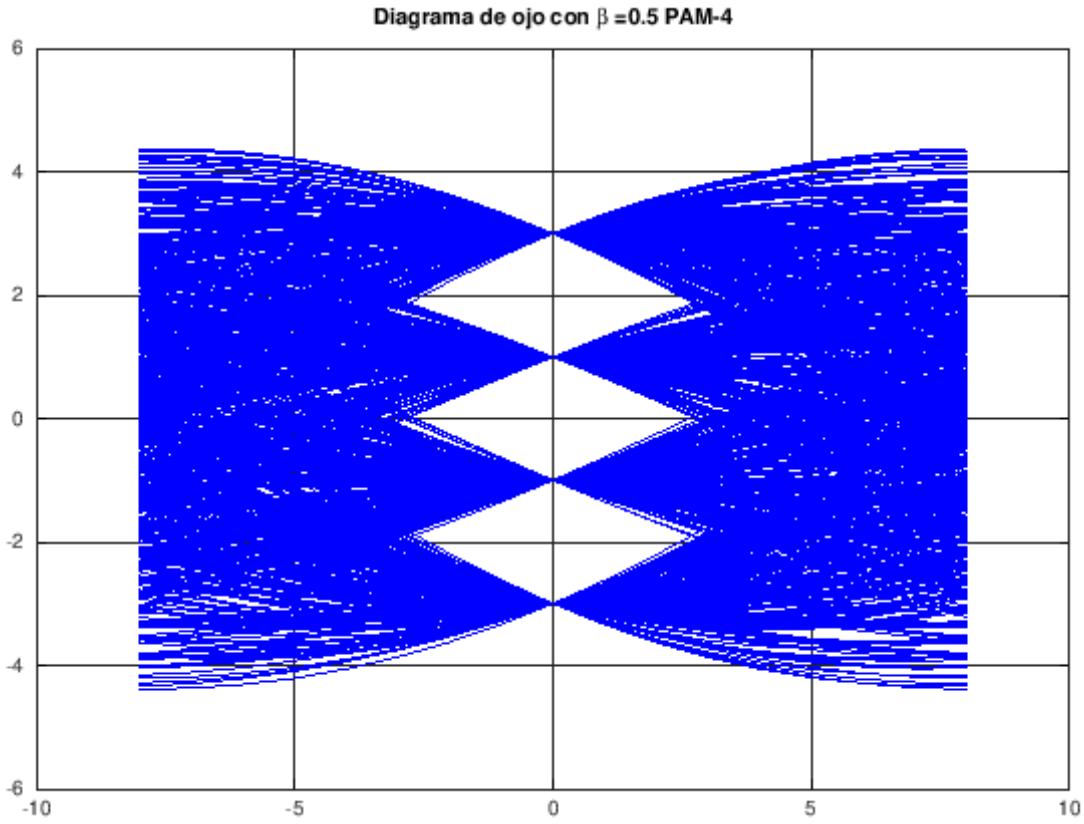


2.4.4 Modificación a PAM-4 con $\beta = 0.5$

Aquí se realiza una modificación del último caso de $\beta = 0.5$ para una señal PAM-4

[16]:
```
%=========================================
% Generacion Simbolos PAM 4
%=========================================
n_symbols = 1000;
PAM_K=4 ; % 2 -> BPSK ;   4-> PAM4

ak = 2*randi([0,PAM_K-1],1,n_symbols)-(PAM_K-1);%PAMK

xn = zeros(1,n_symbols*M);

xn(1:M:end) = ak;
```

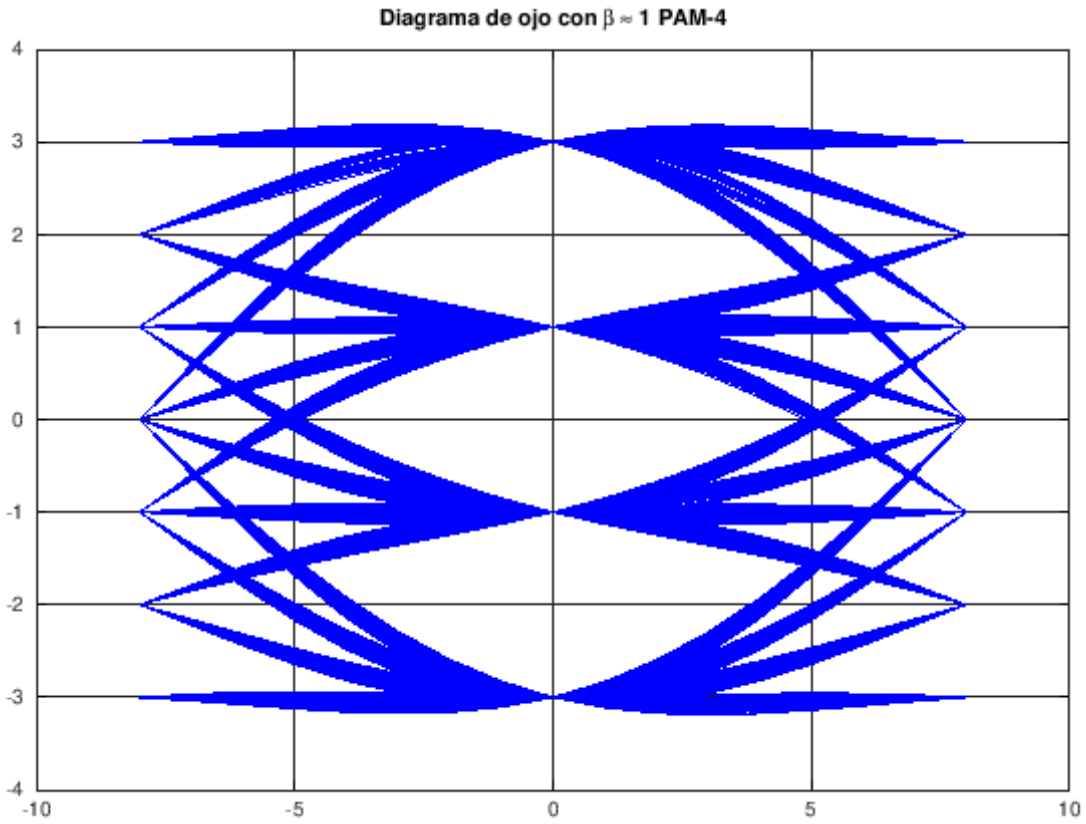
```
%Definimos parámetros del filtro
M = 16;  %Factor de sobremuestreo
beta = 0.50001; %Factor de roll-off
L = 30;  % 2*L*M+1 es el largo del filtro sobremuestreado
t = [-L:1/M:L]*T;

gn = sinc(t/T).*cos(pi*beta*t/T)./(1-4*beta^2*t.^2/T^2);

sn = conv(xn,gn);
for m = 2*L+1:n_symbols-(2*L+1)
    sn_p = sn(m*M+d:m*M+d+M);
    plot([-M/2:1:M/2],sn_p,color='b')
    hold on
end
title('Diagrama de ojo con \beta =0.5 PAM-4')
grid
```



2.4.5 Modificación a PAM-4 con $\beta \approx 1$

Aquí se realiza una modificación del último caso de $\beta \approx 1$ para una señal PAM-4

[17]:
```
%Definimos parámetros del filtro
M = 16;  %Factor de sobremuestreo
beta = 0.9991; %Factor de roll-off
L = 30;  % 2*L*M+1 es el largo del filtro sobremuestreado
t = [-L:1/M:L]*T;

gn = sinc(t/T).*cos(pi*beta*t/T)./(1-4*beta^2*t.^2/T^2);

sn = conv(xn,gn);
for m = 2*L+1:n_symbols-(2*L+1)
    sn_p = sn(m*M+d:m*M+d+M);
    plot([-M/2:1:M/2],sn_p,color='b')
    hold on
end
title('Diagrama de ojo con \beta \approx 1 PAM-4')
grid
```

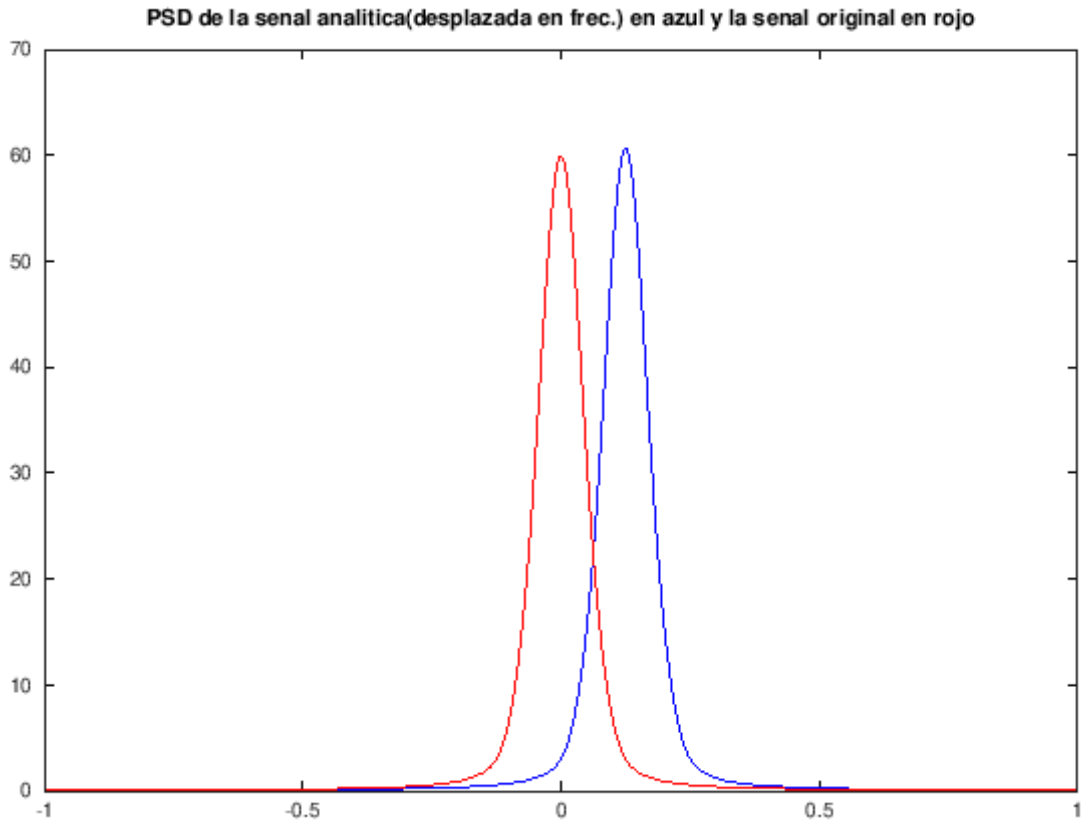


2.5 Densidad espectral de potencia

Si bien ésto no está solicitado en éste práctico, vamos a introducir el tema y presentar las funciones que se usarán en el resto de los laboratorios para poder trabajar con PSD (Power Spectral Density.

Aquí se ve en azul la **señal analítica**,correspondiente a un desplazamiento de la original en rojo. Esto se ve en mayor detalle en el laboratorio N°2.

[18]:
```
%==========================================
% Power Spectral Density
%==========================================
pkg load signal
n=[1:length(sn)];
signal_1 = exp(j*2*pi/M*n).*sn ; % Desplazamiento de 2*pi/M para generar una␣
 ↪señal analitica
signal_2 = sn ;


[Hpsd_1,freq]=pwelch(signal_1,rectwin(32),[],2^10,'twosided','centerdc');
[Hpsd_2]=pwelch(signal_2,rectwin(32),[],2^10,'twosided','centerdc');

plot(freq*2,Hpsd_1,color='b'); % Plot la PSD

hold on
plot(freq*2,Hpsd_2,color='r');
title('PSD de la senal analitica(desplazada en frec.) en azul y la senal␣
 ↪original en rojo')
```



3 Laboratorio N°2

VER PAGINA 745 MESSERSCHMIT COMMUNICATIONS - TIMING RECOVERY

Otro filtro muy utilizado en comunicaciones digitales es el transformador de Hilbert:

$$f[n] = \frac{2\sin^2(\pi n/2)}{\pi n}, \quad n \neq 0; \quad f[0] = 0$$

$$F\left(e^{j\Omega}\right) = \begin{cases} j & \text{si } -\pi < \Omega < 0 \\ -j & \text{si } 0 < \Omega < \pi \end{cases}$$

Se pide:

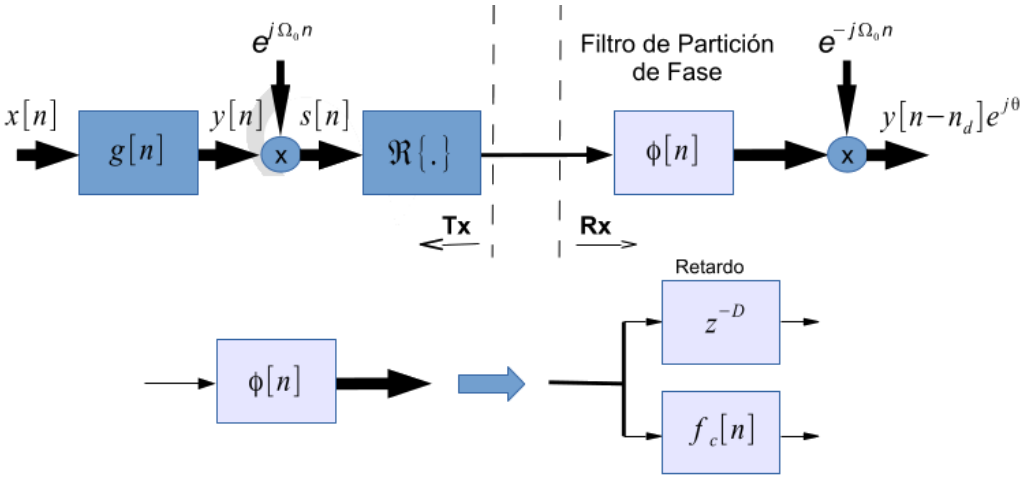
A. Utilizar la auto-función $e^{j\Omega n}$ para obtener por simulación la respuesta en frecuencia $|F(e^{j\Omega})|$ y $\angle F(e^{j\Omega})$

B. Sea $x[n] = \sum_{k=0}^{L-1} a[k]\delta[n - kM]$ con $a[k] \in \{\pm 1 \pm j\}$ y $g[n]$ el filtro de caída cosenoidal del Lab. N°1, seleccionar un valor de Ω_0 para que la señal $s[n] = e^{j\Omega_0 n}(x[n] * g[n])$ sea analítica ($S(e^{j\Omega}) = 0$ para $-\pi < \Omega < 0$)

C. Verificar usando estimación de densidad espectral de potencia que

$$\Im\{s[n]\} = \Re\{s[n]\} * f[n]$$

D. Implementar en un simulador del sistema de comunicaciones digitales elemental de la Figura siguiente :



E. Comprobar que :

Siendo $p[n] = \sum_k a[k]g[n - kM]$:

$$E\left\{|p[n]|^2\right\} = \sum_k |g[n - kM]|^2$$

F. Verificar que $E\left\{|p[n]|^2\right\}$ es una secuencia real periódica con período M y con serie de Fourier dada por:

$$E\left\{|p[n]|^2\right\} = \sum_{q=<M>} b_q e^{jq\frac{2\pi}{M}n}$$

Con:

$$b_q = \frac{1}{2\pi M}\int_{2\pi} G\left(e^{j\Omega}\right) G^*\left(e^{j\left(\Omega - q\frac{2\pi}{m}\right)}\right) d\Omega$$

G. Comprobar que si $\beta < 1$:

$$\begin{aligned} E\left\{|p[n]|^2\right\} &= b_0 + 2\Re\left\{b_1 e^{j\frac{2\pi}{M}n}\right\} \\ &= b_0 + 2\,|b_1|\cos\left(\frac{2\pi}{M}n + \angle b_1\right) \end{aligned}$$

Es decir que **existe un tono cosenoidal periódico con período M** . Lo cual constituye el fundamento del método de la línea espectral muy usado en lazos de recuperación de sincronismo.

H. Utilice simulación para verificar la existencia del tono de pulsación $\Omega = 2\pi/M$ en el espectro de $|p[n]|^2$

I. Considere un segundo canal independiente con señal :

$$\tilde{p}[n] = \sum_k \tilde{a}[k] g\left[n - kM - n_d\right]$$

Con n_d siendo una cierta demora y $\tilde{a}[k]$ siendo independiente de $a[k]$

Verificar que si $\beta < 1$ y $n_d = M/2$:

$$\begin{aligned} E\left\{|\tilde{p}[n]|^2\right\} &= b_0 - 2\Re\left\{b_1 e^{j\frac{2x}{M}n}\right\} \\ &= b_0 - 2\,|b_1|\cos\left(\frac{2\pi}{M}n + \angle b_1\right) \end{aligned}$$

Siendo b_0 y b_1 los calculados en el apartado anterior cuando $n_d = 0$

En base al resultado anterior , mostrar que el tono de sincronismo contenido en :

$$u[n] = p[n] + \tilde{p}[n]$$

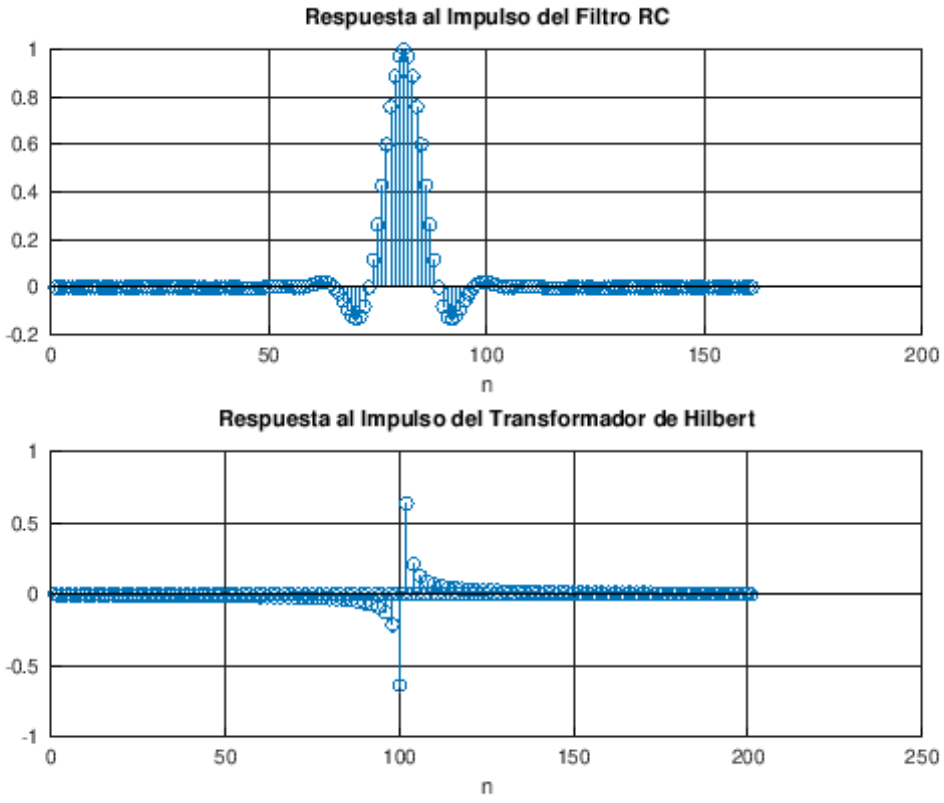
Se cancela, por lo que no puede extraerse señal de sincronismo de $E\left\{|u[n]|^2\right\}$

3.1 Respuesta al impulso del transformador de Hilbert

Aquí inicializamos el código y obtenemos la respuesta al impulso del transformador de Hilbert y la comparamos con la del filtro de caída cosenoidal

[19]:
```
%========================================================================
% Procesamiento de Señales en Tiempo Discreto
% Prof.: Dr. Mario Hueda
% Practico Lab. 2
%========================================================================

pkg load signal;
%=========================================
% Generacion de la Respuesta al Impulso
%=========================================
fB = 32e9;         % Velocidad de simbolos (baud rate)
T = 1/fB; % Tiempo entre simbolos
M = 8;  %Factor de sobremuestreo
fs = fB*M;         % Sample rate

beta = 0.5001; %Factor de roll-off
```

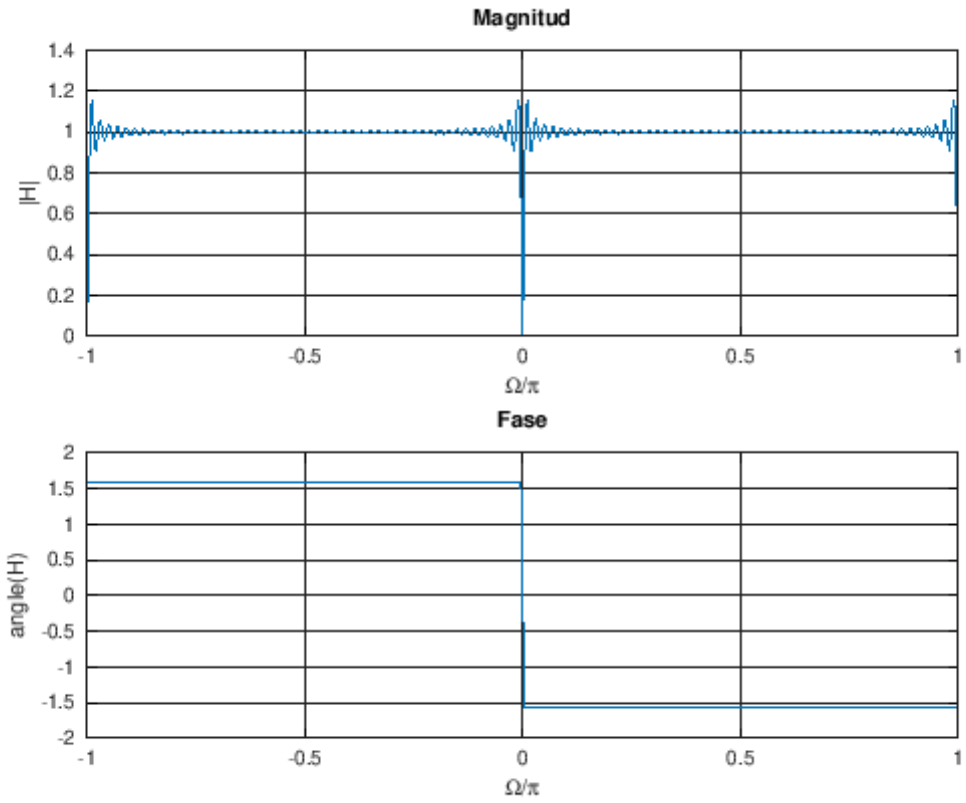
```
L = 10;  % 2*L*M+1 es el largo del filtro sobremuestreado
t = [-L:1/M:L]*T;
n_delay_RC_filter = L*M; %Retardo del filtro RC
gn = sinc(t/T).*cos(pi*beta*t/T)./(1-4*beta^2*t.^2/T^2);

Lf=100;
n=[-Lf:Lf];
n_delay_Hilbert_filter = Lf; %Retardo del filtro de Hilbert
fn = 2*sin(pi*n/2).^2./(pi*n);
fn(Lf+1)=0;

figure(1)
subplot(2,1,1)
h = stem(gn);
title('Respuesta al Impulso del Filtro RC');
xlabel('n');
grid on
subplot(2,1,2)
h = stem(fn);
title('Respuesta al Impulso del Transformador de Hilbert');
xlabel('n');
grid on
```



3.2 A. Respuesta en frecuencia del transformador de Hilbert

Aquí realizamos lo solitado en el primer apartado:

[20]:
```
%=========================================
% Calculo de la Respuesta en Frecuencia
%=========================================
Omega = [-1:1/2^8:1]*pi;
N = 1000;
n = [0:N];
index = 1;
for omega=Omega
    xn = exp(j*omega*n);
    yn = conv(xn,fn);
    H_Mag(index) = abs(yn(N/2));
    H_Fase(index) = angle(yn(N/2)*conj(xn(N/2-n_delay_Hilbert_filter)));
    index = index+1;
endfor

figure(2)
subplot(2,1,1)
h=plot(Omega/pi,H_Mag);
title('Magnitud');
ylabel('|H|')
xlabel('\Omega/\pi');
grid
subplot(2,1,2)
h=plot(Omega/pi,H_Fase);
title('Fase');
ylabel('angle(H)')
xlabel('\Omega/\pi');
grid
```



3.3 B. Generación de símbolos y desplazamiento para obtener señal analítica

3.3.1 Símbolos 4-QAM

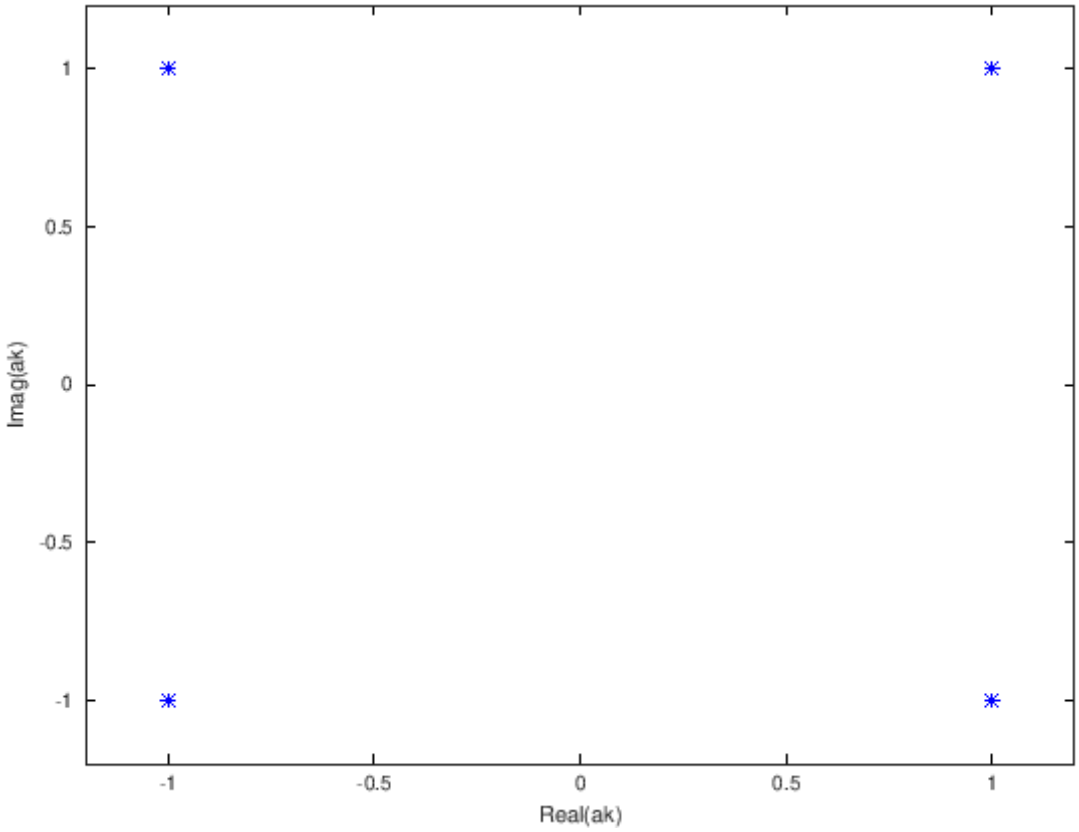
Generamos la secuencia de símbolos :

$$x[n] = \sum_{k=0}^{L-1} a[k]\delta[n - kM] \, \text{con} \, a[k] \in \{\pm 1 \pm j\}$$

Notar que para una mísma velocidad de símbolos(baud-rate) , utilizando 4-QAM enviamos 2 bits de información por cada símbolo con lo cual duplicamos la velocidad de bits (bit-rate o bps)

[21]:
```
%==========================================
% Generar simbolos QPSK

%==========================================
n_symbols = 2^14;
PAM_K=2;
```

```
ak = (2*randi([0,PAM_K-1],1,n_symbols)-(PAM_K-1))␣
 ↪+j*(2*randi([0,PAM_K-1],1,n_symbols)-(PAM_K-1));

xn = zeros(1,n_symbols*M);
xn(1:M:end) = ak;

figure(3)
h = plot(ak,'b*');
xlabel('Real(ak)')
ylabel('Imag(ak)');
axis([-1.2 1.2 -1.2 1.2])
```

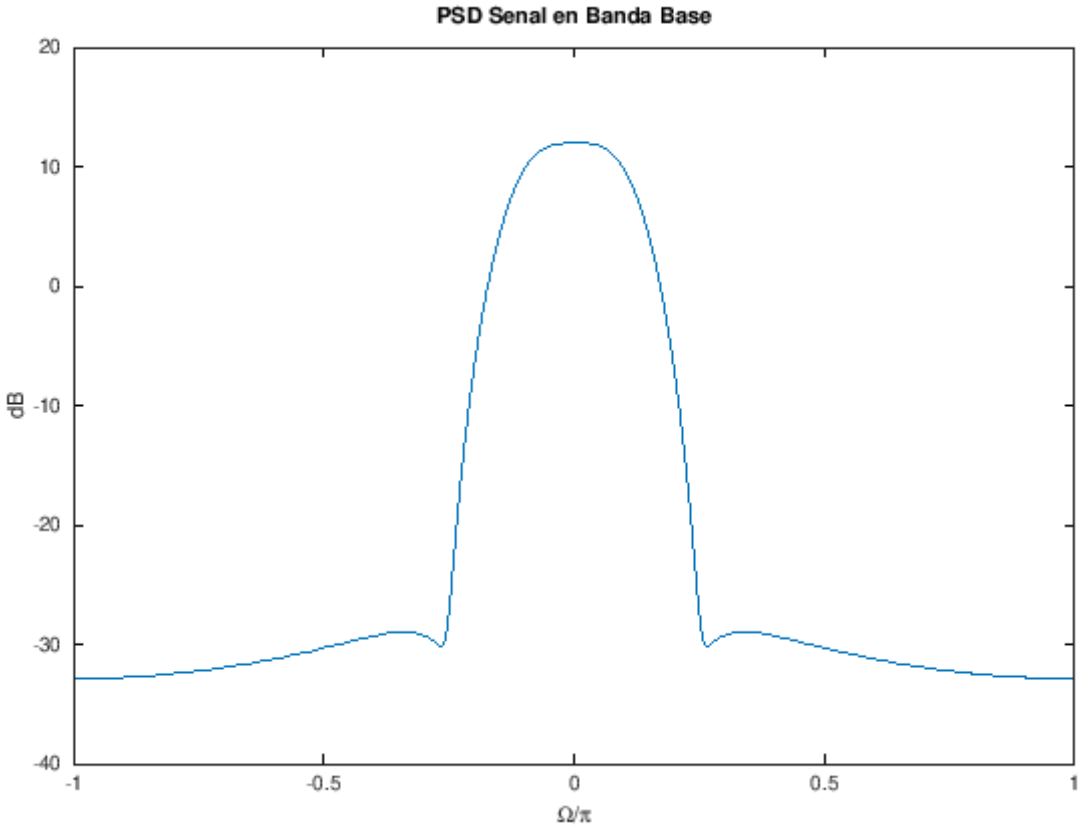


3.3.2 Señal en banda-base

Aquí convolucionamos los símbolos con el filtro de caída cosenoidal $g[n]$ quedando:

$$y[n] = \sum_{k=0}^{L-1} a[k]g[n - kM]$$

[22]:
```
%=========================================
% Señal Banda-Base
%=========================================

yn = conv(xn,gn);
figure(4)

[Hpsd F]=pwelch(yn, 32, [], 2^10,'twosided','centerdc');
h=plot(F*2,10*log10(Hpsd));
title('PSD Senal en Banda Base')
ylabel('dB')
xlabel('\Omega/\pi');
```



3.3.3 Señal analítica (desplazada)

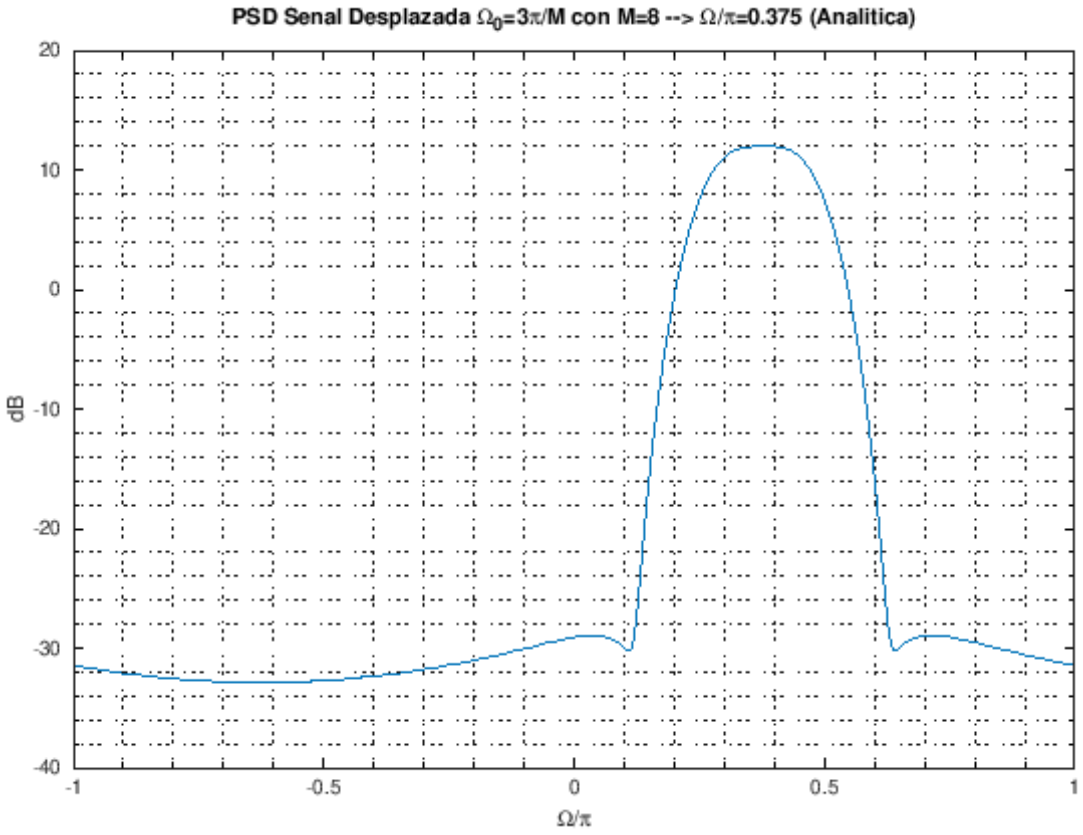
Aquí modulamos la señal $y[n]$ con una exponencial compleja con frecuenca Ω_0 para generar un desplazamiento en frecuencia tal de obtener una secuencia $s[n]$ analítica.

$$s[n] = e^{j\Omega_0 n}(x[n] * g[n])$$

$S(e^{j\Omega}) = 0$ para $-\pi < \Omega < 0$

En el siguiente gráfico puede verse como para un $\Omega_0 = 3\pi/M$ con $M = 8$ ya se cumple la condición de que la señal $s[n]$ sea analítica.

[23]:
```
%=========================================
% Señal Modulada (Analitica)
%=========================================

n=[1:length(yn)];
Omega_c=3*pi/M; %Portadora arbitraria para generar seÃ±al analitica (ojo:␣
 ↪depende de M)
carrier=exp(j*Omega_c*n);
sn = yn.*carrier;

[Hpsd F]=pwelch(sn, 32, [], 2^10,'twosided','centerdc');
h=plot(F*2,10*log10(Hpsd));
title('PSD Senal Desplazada \Omega_0=3\pi/M con M=8 --> \Omega/\pi=0.375␣
 ↪(Analitica)')
grid minor
ylabel('dB')
xlabel('\Omega/\pi');
```



3.4 C. Recuperación de parte imaginaria a partir de parte real usando transformador de Hilbert

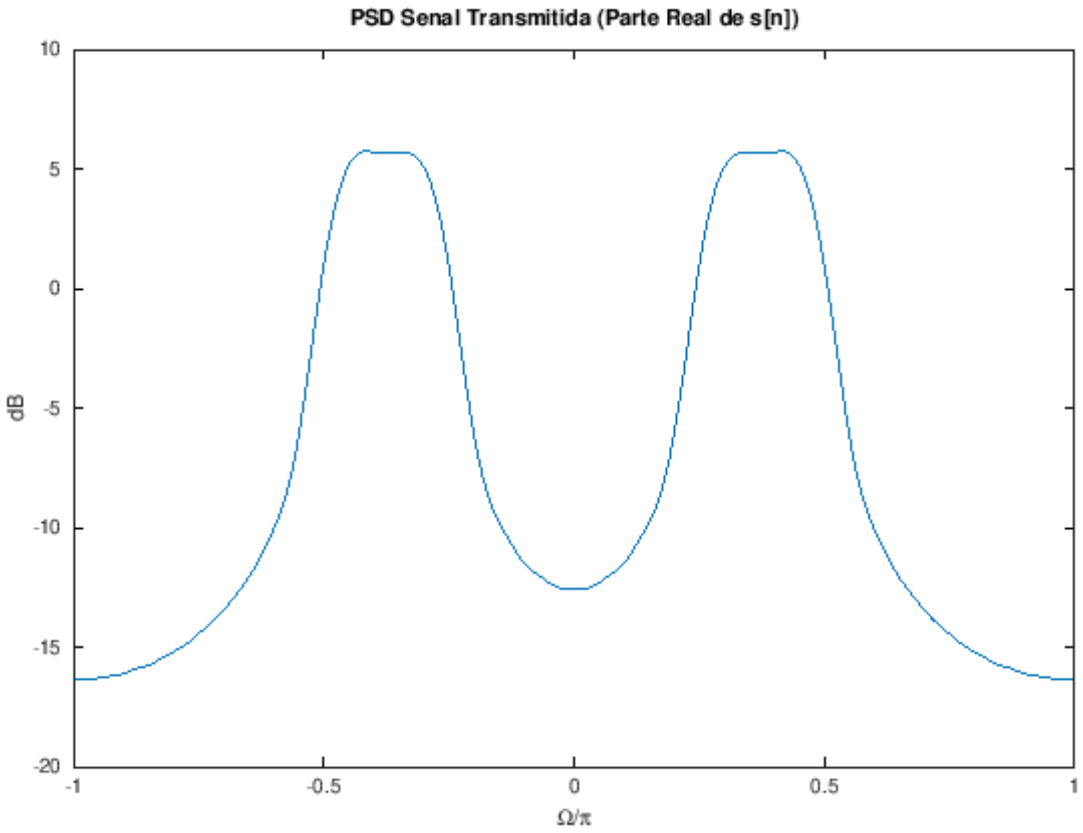
Se nos pide verificar que:

$$\Im\{s[n]\} = \Re\{s[n]\} * f[n]$$

3.4.1 Parte real de $s[n]$ - Señal Transmitida

Por el canal Tx se transmite sólo $\Re\{s[n]\}$

[24]:
```
%=========================================
% Señal Transmitida (Parte Real de la SeÃ±al Analitica)
%=========================================

sn_r = real(sn);
[Hpsd F]=pwelch(sn_r, rectwin(32), [], 2^10,'twosided','centerdc');
h=plot(F*2,10*log10(Hpsd));
title('PSD Senal Transmitida (Parte Real de s[n])')
%grid minor
ylabel('dB')
xlabel('\Omega/\pi');
```



3.4.2 Parte imaginaria de $s[n]$ recuperada de parte real

Aquí realizamos la convolución de la parte real con el transformador de Hilbert

$$\Re\{s[n]\} * f[n]$$

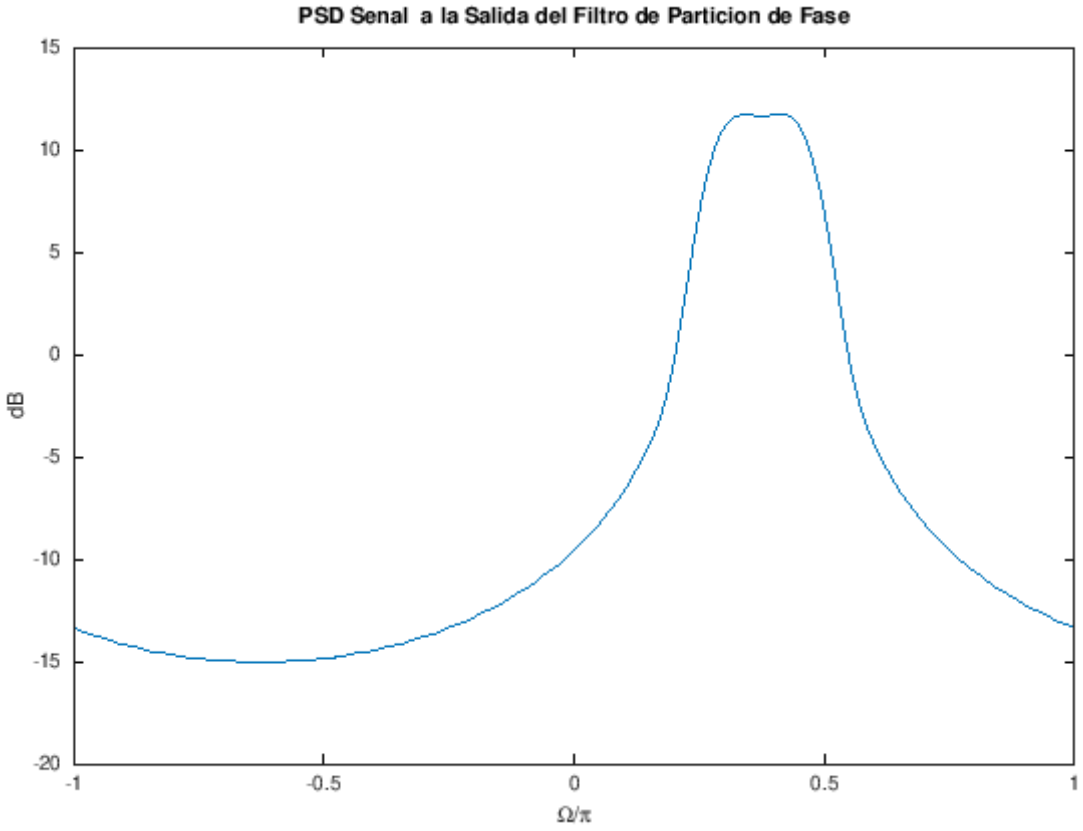
Para obtener la parte imaginaria y luego hacemos el plot de la suma de la parte real mas la parte imaginaria recuperada.

$$\hat{s}[n] = \Re\{s[n]\} + j(\Re\{s[n]\} * f[n])$$

[25]:
```
%=========================================
% Filtro de Particion de Fase
%=========================================

sn_i = conv(sn_r,fn);
```

[26]:
```
sn_hat = sn_r+j*sn_i(Lf+1+0:end-Lf+0);

[Hpsd F]=pwelch(sn_hat, rectwin(32), [], 2^10,'twosided','centerdc');
```

```
h=plot(F*2,10*log10(Hpsd));

title('PSD Senal  a la Salida del Filtro de Particion de Fase')
ylabel('dB')
xlabel('\Omega/\pi');
```



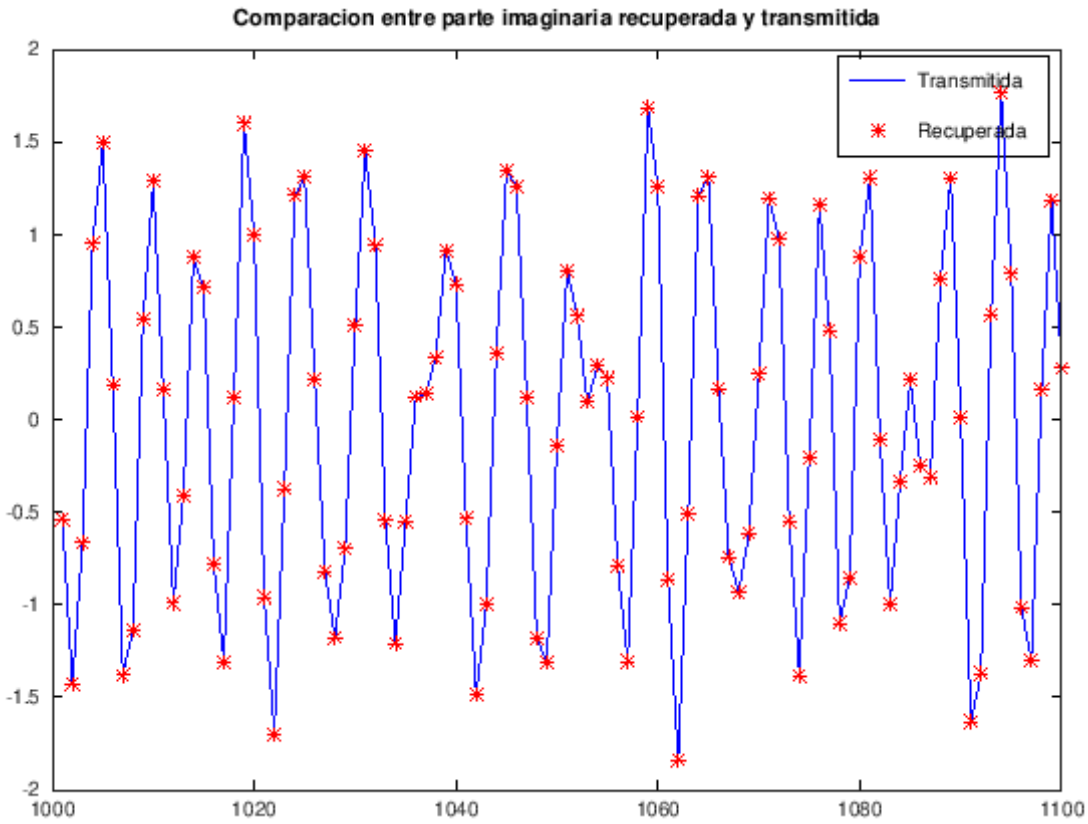
3.4.3 Comparación entre parte imaginaria recuperada y transmitida

Aquí comprobamos si realmente hemos logrado recuperar la parte imaginaria transmitida de la parte real recibida.

Para el ploteo de la señal recuperada desplazamos a la izquierda la misma para que quede en fase con la original transmitida. Este desplazamiento es proporcional a la longitud del filtro de Hilbert que usemos.

[27]:
```
n=[1:100]+1000; %100 puntos de ventana de tiempo arbitraria
h=plot(n,imag(sn(n)),'b',n,sn_i(n+Lf),'r*'); % Lf se suma para desplazar a la␣
 ↪izquierda la señal recuperada
%h=plot(n,imag(sn(n)),'b',n,sn_i(n),'r*'); % Probar este plot para ver la␣
 ↪necesidad del desplazamiento en Lf
```

```
legend('Transmitida', 'Recuperada');
title('Comparacion entre parte imaginaria recuperada y transmitida')
```



3.5 D. Resto de etapas de un sistema de comunicaciones elemental

Con lo visto hasta el punto anterior, hemos observado las etapas de procesamiento inicial con el filtro de caida cosenoidal, la modulación para lograr una señal analítica u luego transmitir su parte real y hemos visto como en el receptor recuperamos la parte imaginaria de la parte real transmitida.

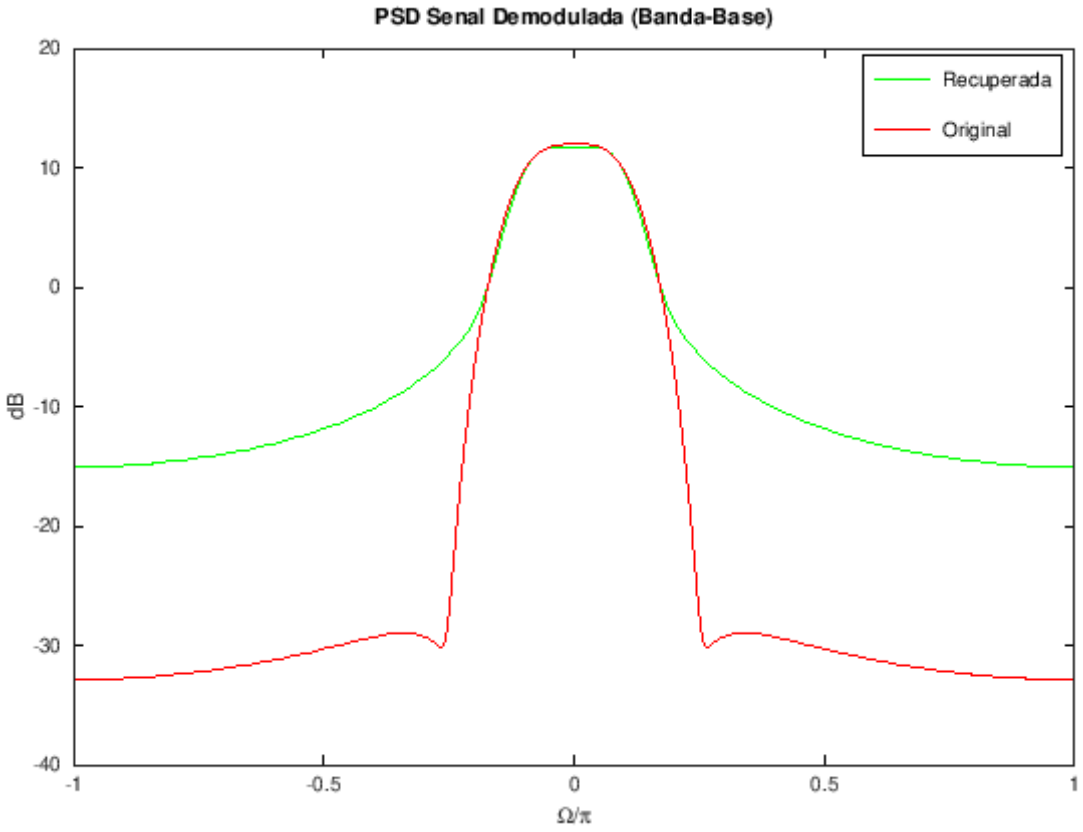
Nos falta demodular la señal de nuevo a banda-base para lo cual la multiplicamos por el conjugado de la exponencial compleja que usamos para convertirla en analítica.

$$\hat{y}[n] = \hat{s}[n]e^{-j\Omega_0 n}$$

3.5.1 Señal demodulada

En verde podemos ver el espectro de la señal recibida demodulada mientras que en rojo podemos ver la señal transmitida en banda base

[28]:
```
%==========================================
% Señal Demodulada (Banda-Base)
%==========================================

n=[1:length(sn_hat)];
carrier=exp(j*Omega_c*n);
yn_hat = sn_hat.*conj(carrier); %portadora conjugada (demodulacion)


[Hpsd F]=pwelch(yn_hat, rectwin(32), [], 2^10,'twosided','centerdc');
h=plot(F*2,10*log10(Hpsd),'g');
hold on
[Hpsd F]=pwelch(yn, 32, [], 2^10,'twosided','centerdc');
h=plot(F*2,10*log10(Hpsd),'r');
hold off
title('PSD Senal Demodulada (Banda-Base)')
ylabel('dB')
xlabel('\Omega/\pi');
legend('Recuperada','Original');
```



3.6 E. Verificaciones de Esperanza Matemática

Se nos pide verificar que:

Siendo $p[n] = \sum_k a[k]g[n - kM]$ y $E\left\{|a[k]|^2\right\} = 1$:

$$E\left\{|p[n]|^2\right\} = \sum_k |g[n - kM]|^2$$

3.6.1 Esperanza matemática

La esperanza matemática en inglés llamada "Expected Value" , o valor-esperado se puede escribir como:

$$E\{X\} = \sum_{k=1}^{N} x_i p_i$$

Siendo x_i el valor de la variable en cada instante y p_i la probabilidad para cada correspondiente valor de la variable.

En el caso de vectores de coeficientes a_k equiprobables como los que venimos usando en los laboratorios, los $p_i = 1/K$ siendo K la longitud del vector que contiene a los $a[k] \in \{\pm 1\}$ por lo que podemos escribir:

$$E\{a_k\} = \frac{1}{K} \sum_{k=1}^{K} a_k$$

y para el caso complejo con $a[k] \in \{\pm 1 \pm j\}$:

$$E\{a_k a_k^*\} = E\left\{|a_k|^2\right\} = \frac{1}{2K} \sum_{k=1}^{K} a_k a_k^* = 1$$

Comprobación de espranza para $a[k] \in \{\pm 1 \pm j\}$: Aquí comprobamos que para $a[k] \in \{\pm 1 \pm j\}$:

$$E\{a_k a_k^*\} = E\left\{|a_k|^2\right\} = \frac{1}{2K} \sum_{k=1}^{K} a_k a_k^* = 1$$

[29]:
```
K = 2^14;

ak = (2*randi([0,1],1,K)-(1)) +j*(2*randi([0,1],1,K)-(1));
```

[30]:
```
ak_mod=ak.*conj(ak);

E_ak_mod=(1/(2*K))*sum(ak_mod)
```

```
E_ak_mod =  1
```

3.6.2 Valor esperado equivalente al módulo del coseno realzado

Para calcular el módulo de $p[n]$ lo multimplicamos por su conjugado:

$$p[n] = \sum_k a[k]g[n - kM]$$

$$p^*[n] = \sum_r a^*[r]g^*[n - rM]$$

$$E\left\{p[n]p^*[n]\right\} = E\left\{|p[n]|^2\right\}$$

$$E\left\{|p[n]|^2\right\} = E\left\{\sum_k \sum_r a[k]a^*[r]g[n - kM]g^*[n - rM]\right\}$$

Debido a lo visto en el apartado anterior y la definición de esperanza matemática que sólo se aplica a los términos aleatórios (**Ver página 745 Messerschmitt**) , es equivalente realizar:

$$E\left\{|p[n]|^2\right\} = \sum_k \sum_r E\left\{a[k]a^*[r]\right\} g[n - kM]g^*[n - rM]$$

$$E\left\{|p[n]|^2\right\} = \sum_k \sum_r \delta[k - r]g[n - kM]g^*[n - rM]$$

Lo que aporta valores no nulos para $k = r$, entonces escribimos:

$$E\left\{|p[n]|^2\right\} = \sum_k g[n - kM]g^*[n - kM]$$

$$E\left\{|p[n]|^2\right\} = \sum_k |g[n - kM]|^2$$

3.7 F. Secuencia valor-esperado real y periódica

Para verificar que $E\left\{|p[n]|^2\right\}$ es periódica con período M, escribimos:

$$E\left\{|p[n + M]|^2\right\} = \sum_k |g[n - (k - 1)M]|^2$$

Haciendo $r = k - 1$

$$E\left\{|p[n+M]|^2\right\} = \sum_k |g[n-rM]|^2$$

Y como sumar para todos los valores de r es equivalente a sumar para todos los valores de k queda **verificada la periodicidad**.

$$E\left\{|p[n+M]|^2\right\} = \sum_k |g[n-rM]|^2 = E\left\{|p[n]|^2\right\}$$

Además como estamos tratando con el módulo de un número complejo al cuadrado, claramente la secuencia **es real** y positiva.

3.7.1 Serie de Fourier de la secuencia valor-esperado

Por lo visto en el apartado anterior podemos escribir la serie de Fourier de una secuencia periódica con período M como :

$$E\left\{|p[n]|^2\right\} = \sum_{q=<M>} b_q e^{jq\frac{2\pi}{M}n}$$

El siguiente desarrollo nos permitirá obtener los coeficientes de la serie de fourier como se ha visto en clase.

Primero definimos :

$$x[n] = g[n]g^*[n] = |g[n]|^2$$

y por simplicidad renombramos a la secuencia esperanza-matemática como $\tilde{s}[n]$:

$$\tilde{s}[n] = E\left\{|p[n]|^2\right\} = \sum_k |g[n-kM]|^2 = \sum_k x[n-kM]$$

Siendo $g[n]$ la respuesta al impulso del filtro transmisor, en éste caso el coseno realzado visto en los apartados anteriores.

Podemos escribir la **definición de los coeficientes de la serie de Fourier** de la secuencia periódca con período M $\tilde{s}[n]$ como:

$$b_q = \frac{1}{M} \sum_{n=<M>} \tilde{s}[n] e^{-jq\frac{2\pi}{M}n}$$

Y como

$$\tilde{s}[n] = \sum_k x[n-kM]$$

Reescribimos los b_q como :

$$b_q = \frac{1}{M} \sum_{n=<M>} \sum_k x[n - kM] e^{-jq\frac{2\pi}{M}n}$$

Si multiplicamos la expresión anterior por $e^{jq\frac{2\pi}{M}kM} = 1$ nos queda:

$$b_q = \frac{1}{M} \sum_{n=<M>} \sum_k x[n - kM] e^{-jq\frac{2\pi}{M}n} e^{jq\frac{2\pi}{M}kM}$$

Y podemos reescribir lo anterior de la siguiente manera:

$$b_q = \frac{1}{M} \sum_{n=<M>} \sum_k x[n - kM] e^{-jq\frac{2\pi}{M}(n-kM)}$$

Y si definimos $y[n]$ como

$$y[n] = x[n] e^{-jq\frac{2\pi}{M}n}$$

Puedo reescribir b_q como

$$b_q = \frac{1}{M} \sum_{n=<M>} \sum_k y[n - kM]$$

Que con mas detalle se escribe:

$$b_q = \frac{1}{M} \sum_{n=0}^{M-1} \sum_{k=-\infty}^{\infty} y[n - kM]$$

Si consideramos por ejemplo para $n = 0$ obtendremos la suma de los términos en k como $y[0]$, $y[\pm M]$,$y[\pm 2M]$… etc.

Si consideramos para $n = 1$ obtendremos la suma de los términos en k como $y[1]$, $y[\pm M+1]$,$y[\pm 2M+1]$… etc.

Si por último consideramos para $n = M - 1$ la suma de los términos en k como $y[M-1]$, $y[\pm M + M - 1]$,$y[\pm 2M + M - 1]$… etc.

Lo que nos permite ver que la sumatoria anterior es equivalente a sumar todos los valores de $y[k]$

$$b_q = \frac{1}{M} \sum_{k=-\infty}^{\infty} y[k]$$

Por lo que reescribiendo en función de $x[n]$ nos queda

$$b_q = \frac{1}{M} \sum_{k=-\infty}^{\infty} x[k] e^{-jq\frac{2\pi}{M}k}$$

La expresión anterior corresponde a la Transformada de Fourier de $x[n] = g[n]g^*[n] = |g[n]|^2$ valuada en $\Omega = q\frac{2\pi}{M}$

$$b_q = \frac{1}{M} X\left(e^{j\Omega}\right)\Big|_{\Omega=\frac{q2\pi}{M}}$$

3.8 G. Comprobación de existencia de tono senoidal periódico en la secuencia esperanza-matemática

Debido a las propiedades de la transformada de fourier, si llamamos $x[n] = g[n]g^*[n]$, teniendo en cuenta de que un producto en el tiempo equivale a una convolución en frecuencia, podemos escribir la transformada de fourier de esa secuencia como :

$$X\left(e^{j\Omega}\right) = \frac{1}{2\pi} G(e^{j\Omega}) * G^*(e^{j\Omega})$$

$$X\left(e^{j\Omega}\right) = \frac{1}{2\pi} \int_{2\pi} G\left(e^{j\Theta}\right) G^*\left(e^{j(\Theta-\Omega)}\right) d\Theta$$

Recordando la conclusión del apartado anterior tenemos:

$$b_q = \frac{1}{M} X\left(e^{j\Omega}\right)\Big|_{\Omega=\frac{q2\pi}{M}}$$

Por lo que escribimos:

$$b_q = \frac{1}{M2\pi} \int_{2\pi} G\left(e^{j\Theta}\right) G^*\left(e^{j(\Theta-\Omega)}\right) d\Theta\Big|_{\Omega=\frac{q2\pi}{M}}$$
$$b_q = \frac{1}{M2\pi} \int_{2\pi} G\left(e^{j\Omega}\right) G^*\left(e^{j\left(\Omega-\frac{q2\pi}{M}\right)}\right) d\Omega$$

Calculamos los coefientes no nulos de la serie de Fourier que, para el caso de interés con un factor de roll-off $\beta < 1$ se dan para $|q| \leq 1$

$$\begin{aligned}
b_0 &= \frac{1}{2\pi M} \int_{2\pi} G\left(e^{j\Omega}\right) G^*\left(e^{j\Omega}\right) d\Omega \\
&= \frac{1}{2\pi M} \int_{2\pi} \left|G\left(e^{j\Omega}\right)\right|^2 d\Omega \\
b_1 &= \frac{1}{2\pi M} \int_{2\pi} G\left(e^{j\Omega}\right) G\left(e^{j\left(\Omega-\frac{2\pi}{M}\right)}\right) d\Omega \\
b_{-1} &= \frac{1}{2\pi M} \int_{2\pi} G\left(e^{j\Omega}\right) G\left(e^{j\left(\Omega+\frac{2\pi}{M}\right)}\right) d\Omega
\end{aligned}$$

Si graficaramos el espectro de $G\left(e^{j\Omega}\right)$ se observaría la **dependencia del tono de timing con el factor de roll-off** , ya que para valores de β mayores, el solapamiento entre el espectro $G\left(e^{j\Omega}\right)$ y sus réplicas en $q\frac{2\pi}{M}$ es mayor, por lo tanto el tono de timing es mayor.

Reescribimos la serie de Fourier con sus tres términos distintos de cero como:

$$E\left\{|p[n]|^2\right\} = \sum_{q=-1}^{1} b_q e^{jq\frac{2\pi}{M}n}$$

Teniendo en cuenta que $E\left\{|p[n]|^2\right\}$ es periódica y real, se cumple que $b_{-1} = b_1^*$, lo que nos permite escribir :

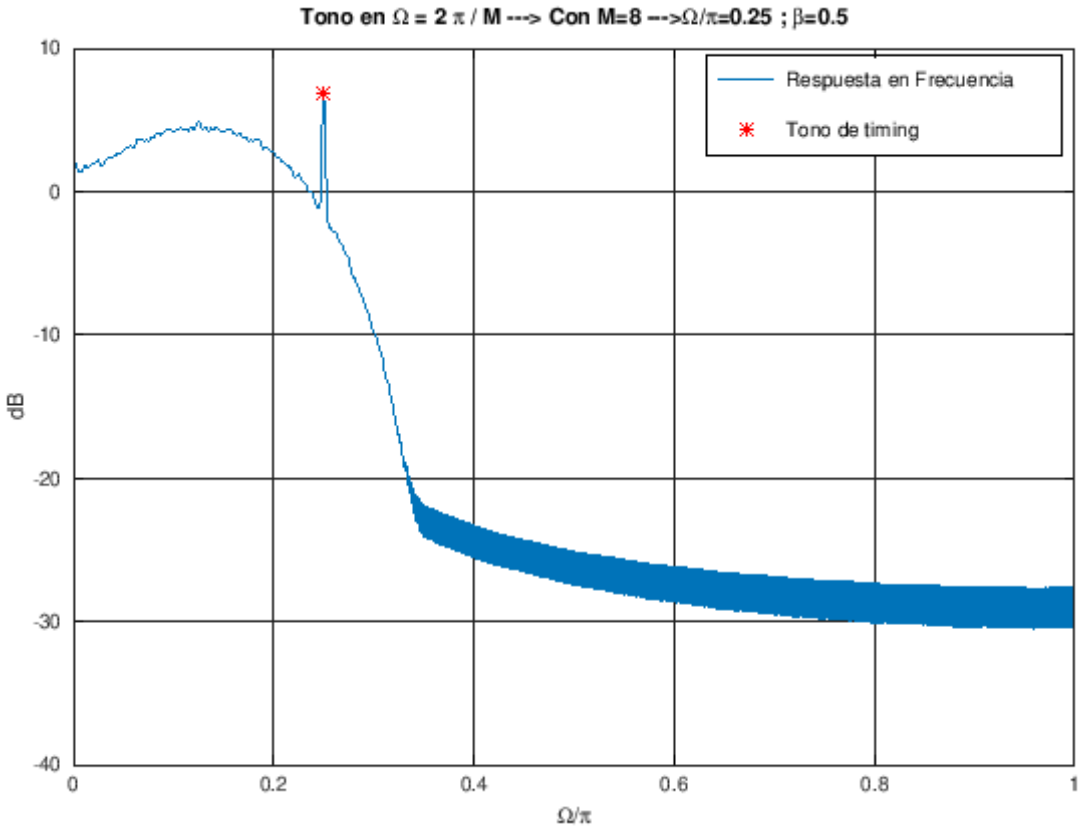
$$\begin{aligned} E\left\{|p[n]|^2\right\} &= b_0 + b_1 e^{\left(j\frac{2\pi}{M}n\right)} + b_1^* e^{\left(-j\frac{2\pi}{M}n\right)} \\ &= b_0 + 2\Re\left(b_1 e^{\left(j\frac{2\pi}{M}n\right)}\right) \\ &= b_0 + 2\left|b_1\right|\cos\left(\frac{2\pi}{M}n + \angle b_1\right) \end{aligned}$$

3.9 H. Simulación tono senoidal en secuencia esperanza-matemática

Aquí simularemos el espectro de la secuencia $E\left\{|p[n]|^2\right\}$ para distintos valores de β y M.

[31]:
```
M = 8;  %Factor de sobremuestreo

ak = (2*randi([0,1],1,n_symbols)-1) +j*(2*randi([0,1],1,n_symbols)-1);
xn = zeros(1,n_symbols*M);
xn(1:M:end) = ak;


beta = .50001; %Factor de roll-off
%beta=0.9999999995;
L = 40;  % 2*L*M+1 es el largo del filtro sobremuestreado
t = [-L:1/M:L]*T;

gn = sinc(t/T).*cos(pi*beta*t/T)./(1-4*beta^2*t.^2/T^2);
pn = conv(xn,gn);


%
yn=pn;
Z=abs(yn).^2;

[pxx w]=pwelch(Z,1024,[],1024*2);
[G,loc]=max(10*log10(sqrt(pxx)));
```

[32]:
```
plot(2*w,10*log10(sqrt(pxx)),2*w(loc),G,'r*');
title('Tono en \Omega = 2 \pi / M ---> Con M=8 --->\Omega/\pi=0.25 ; \beta=0.5'␣
 ↪)
ylabel('dB')
xlabel('\Omega/\pi');
legend('Respuesta en Frecuencia','Tono de timing');
grid
```

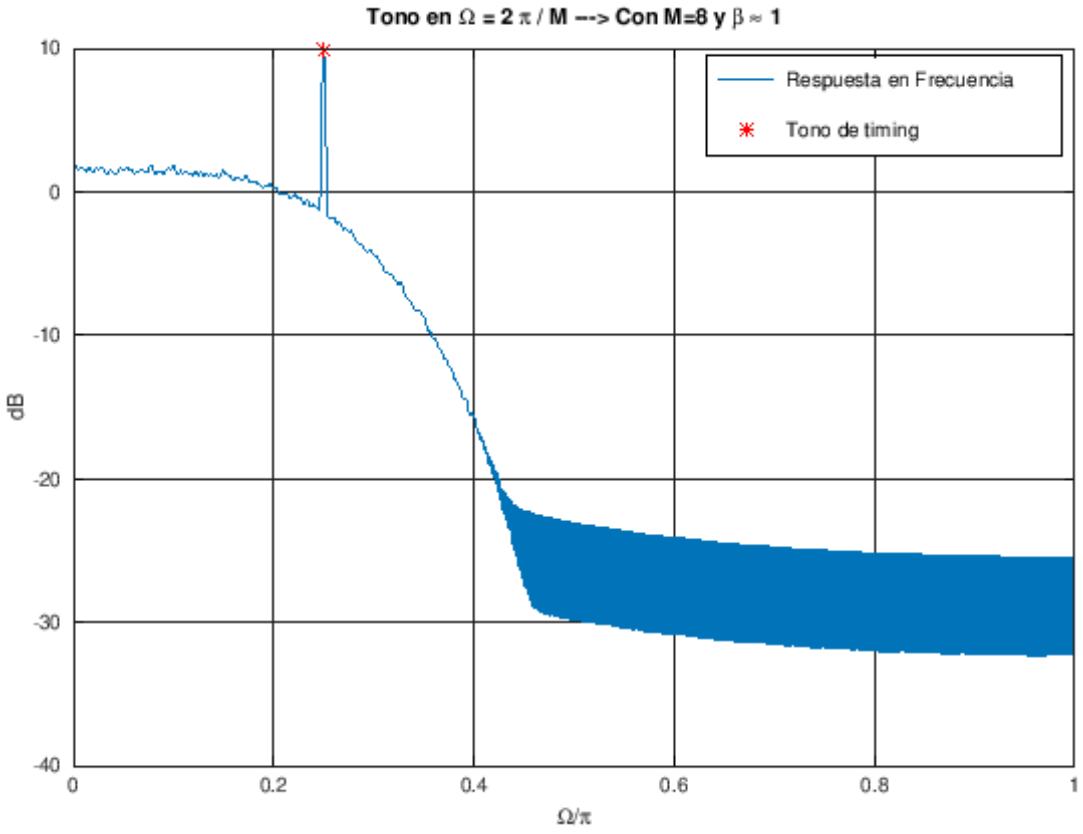


3.9.1 Simulación dependencia de tono de timing con factor de roll-off

Aquí aumentamos el roll-off y observamos que el tono de timing aumenta en amplitud.

[33]:
```
% AQUI AUMENTAMOS EL ROLL-OFF PARA COMPROBAR QUE EL TONO DE TIMING ES MAYOR A␣
 ↪MEDIDA QUE
%AUMENTA EXCESO DE ANCHO DE BANDA
M = 8;  %Factor de sobremuestreo

ak = (2*randi([0,1],1,n_symbols)-1) +j*(2*randi([0,1],1,n_symbols)-1);
xn = zeros(1,n_symbols*M);
xn(1:M:end) = ak;
```

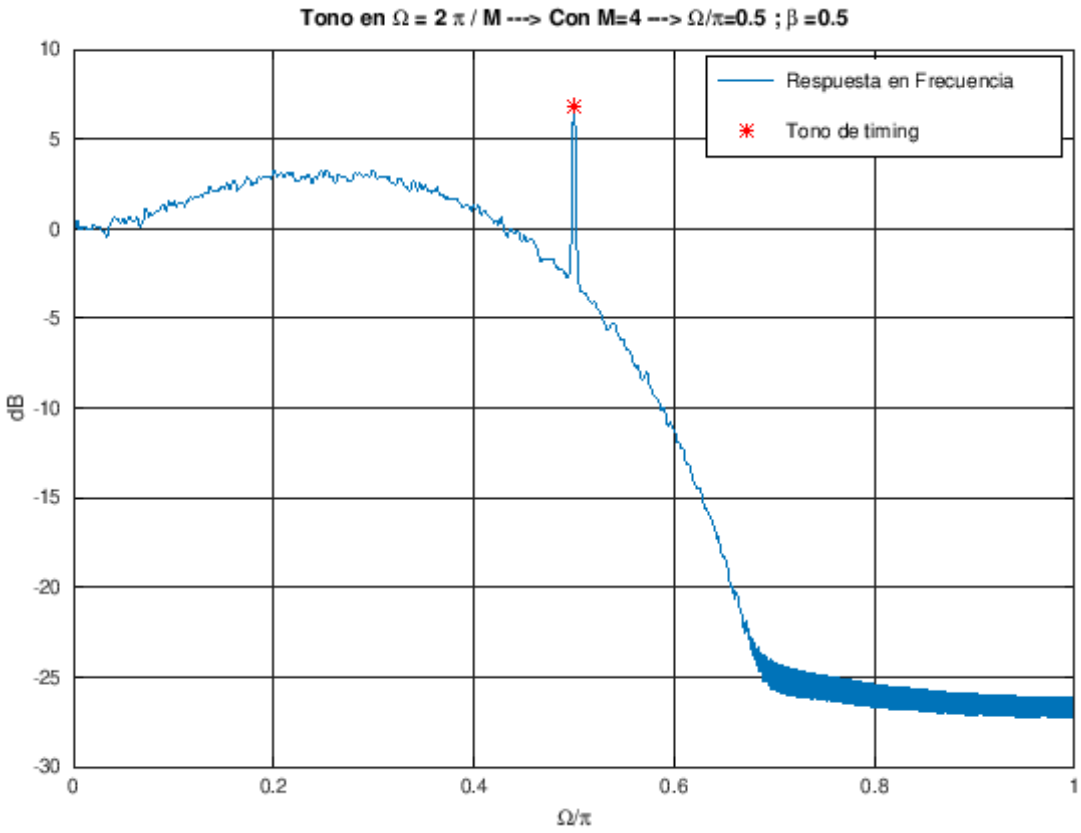
```
%beta = .50001; %Factor de roll-off
beta=0.9999999995;
L = 40;  % 2*L*M+1 es el largo del filtro sobremuestreado
t = [-L:1/M:L]*T;

gn = sinc(t/T).*cos(pi*beta*t/T)./(1-4*beta^2*t.^2/T^2);
pn = conv(xn,gn);


%
yn=pn;
Z=abs(yn).^2;

[pxx w]=pwelch(Z,1024,[],1024*2);
[G,loc]=max(10*log10(sqrt(pxx)));
plot(2*w,10*log10(sqrt(pxx)),2*w(loc),G,'r*');
title('Tono en \Omega = 2 \pi / M ---> Con M=8 y \beta \approx 1')
ylabel('dB')
xlabel('\Omega/\pi');
legend('Respuesta en Frecuencia','Tono de timing');
grid
```



3.9.2 Simulación de desplazamiento de tono de timing al disminuir factor de sobremuestreo M

Disminuímos el factor de sobremuestreo M y observamos un desplazamiento a la derecha del tono de timing.

[34]:
```
M = 4;  %Factor de sobremuestreo

ak = (2*randi([0,1],1,n_symbols)-1) +j*(2*randi([0,1],1,n_symbols)-1);
xn = zeros(1,n_symbols*M);
xn(1:M:end) = ak;


beta = .50001; %Factor de roll-off
%beta=0.9999999995;
L = 40;  % 2*L*M+1 es el largo del filtro sobremuestreado
t = [-L:1/M:L]*T;

gn = sinc(t/T).*cos(pi*beta*t/T)./(1-4*beta^2*t.^2/T^2);
pn = conv(xn,gn);


%
yn=pn;
Z=abs(yn).^2;

[pxx w]=pwelch(Z,1024,[],1024*2);
[G,loc]=max(10*log10(sqrt(pxx)));
plot(2*w,10*log10(sqrt(pxx)),2*w(loc),G,'r*');
title('Tono en \Omega = 2 \pi / M ---> Con M=4 ---> \Omega/\pi=0.5 ; \beta =0.
 ↪5')
ylabel('dB')
xlabel('\Omega/\pi');
legend('Respuesta en Frecuencia','Tono de timing');
grid
```

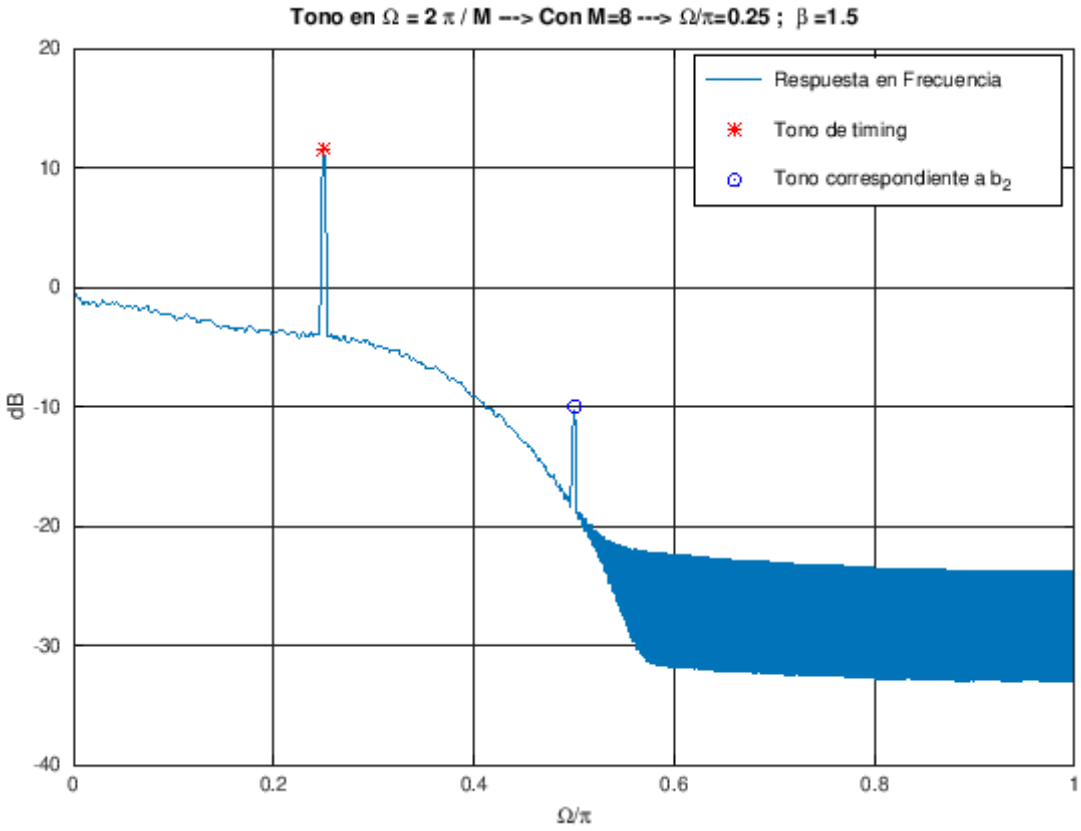


3.9.3 Simulación de aparición de coeficientes (tonos) extra cuando $\beta > 1$

Aquí veremos en simulación como cuando usamos valores de $\beta > 1$ aparece el tono correspondiente al coeficiente de la serie de Fourier b_2 , que antes no aparecía

[35]:
```
M = 8;  %Factor de sobremuestreo

ak = (2*randi([0,1],1,n_symbols)-1) +j*(2*randi([0,1],1,n_symbols)-1);
xn = zeros(1,n_symbols*M);
xn(1:M:end) = ak;


beta = 1.50001; %Factor de roll-off
%beta=0.9999999995;
L = 40;  % 2*L*M+1 es el largo del filtro sobremuestreado
t = [-L:1/M:L]*T;

gn = sinc(t/T).*cos(pi*beta*t/T)./(1-4*beta^2*t.^2/T^2);
pn = conv(xn,gn);
```

```
%
yn=pn;
Z=abs(yn).^2;

[pxx w]=pwelch(Z,1024,[],1024*2);
[G,loc]=max(10*log10(sqrt(pxx)));
plot(2*w,10*log10(sqrt(pxx)),2*w(loc),G,'r*',2*w(2*loc),-10,'bo');
title('Tono en \Omega = 2 \pi / M ---> Con M=8 ---> \Omega/\pi=0.25 ;  \beta =1.
 ↪5')
ylabel('dB')
xlabel('\Omega/\pi');
legend('Respuesta en Frecuencia','Tono de timing','Tono correspondiente a b_2');
grid
```



3.10 I. Comprobación de cancelación de tono senoidal

Consideramos ahora una secuencia independiente :

$$\tilde{p}[n] = \sum_k \tilde{a}[k] g\left[n - kM - n_d\right]$$

Con n_d siendo una cierta demora y $\tilde{a}[k]$ siendo independiente de $a[k]$

Usando la propiedad de desplazamiento de la TF:

$$\mathbb{F}\left\{g\left[n - n_d\right]\right\} \rightarrow e^{-j\Omega n_d} G\left(e^{j\Omega}\right)$$

Si hacemos $n_d = M/2$ y calculamos los coeficientes para la nueva secuencia tenemos:

$$b'_k = \left.\frac{e^{-j\Omega n_d}}{M} X\left(e^{j\Omega}\right)\right|_{\Omega=\frac{k2\pi}{M}}$$

$$b'_k = \left.\frac{e^{-j\Omega \frac{M}{2}}}{M} X\left(e^{j\Omega}\right)\right|_{\Omega=\frac{k2\pi}{M}}$$

Reemplazando $\Omega = \frac{k2\pi}{M}$:

$$b'_k = \left.\frac{e^{-j\frac{k2\pi}{M}\frac{M}{2}}}{M} X\left(e^{j\Omega}\right)\right|_{\Omega=\frac{k2\pi}{M}}$$

$$b'_k = \left.\frac{e^{-jk\pi}}{M} X\left(e^{j\Omega}\right)\right|_{\Omega=\frac{k2\pi}{M}}$$

$$b'_k = e^{-jk\pi} b_k$$

Particularizando para los valores no nulos tenemos:

$$\begin{aligned}
b'_1 &= e^{-j\pi} b_1 = -b_1 \\
b'_{-1} &= e^{j\pi} b_{-1} = -b_{-1} \\
E\left\{|\tilde{p}[n]|^2\right\} &= b_0 - b_1 e^{\left(j\frac{2\pi}{M}n\right)} - b_1^* e^{\left(-j\frac{2\pi}{M}n\right)} \\
&= b_0 - \left(b_1 e^{\left(j\frac{2\pi}{M}n\right)} + b_1^* e^{\left(-j\frac{2\pi}{M}n\right)}\right) \\
&= b_0 - 2\Re\left(b_1 e^{\left(j\frac{2\pi}{M}n\right)}\right)
\end{aligned}$$

Con lo que tenemos que si generamos $u[n] = p[n] + \tilde{p}[n]$, nos queda en base a los resultados anteriores:

$$E\left\{|p[n]|^2\right\} + E\left\{|\tilde{p}[n]|^2\right\} = 2b_0$$

Con lo que comprobamos que el tono senoidal queda cancelado, con lo que concluimos en que no puede extraerse señal de sincronismo de $E\left\{|u[n]|^2\right\}$

3.10.1 Simulación Cancelación del tono senoidal

Aquí generamos una nueva secuencia independiente, que tiene la demora $n_d = M/2$ respecto de la anterior, y comprobamos que al sumarlas se cancela el tono senoidal.

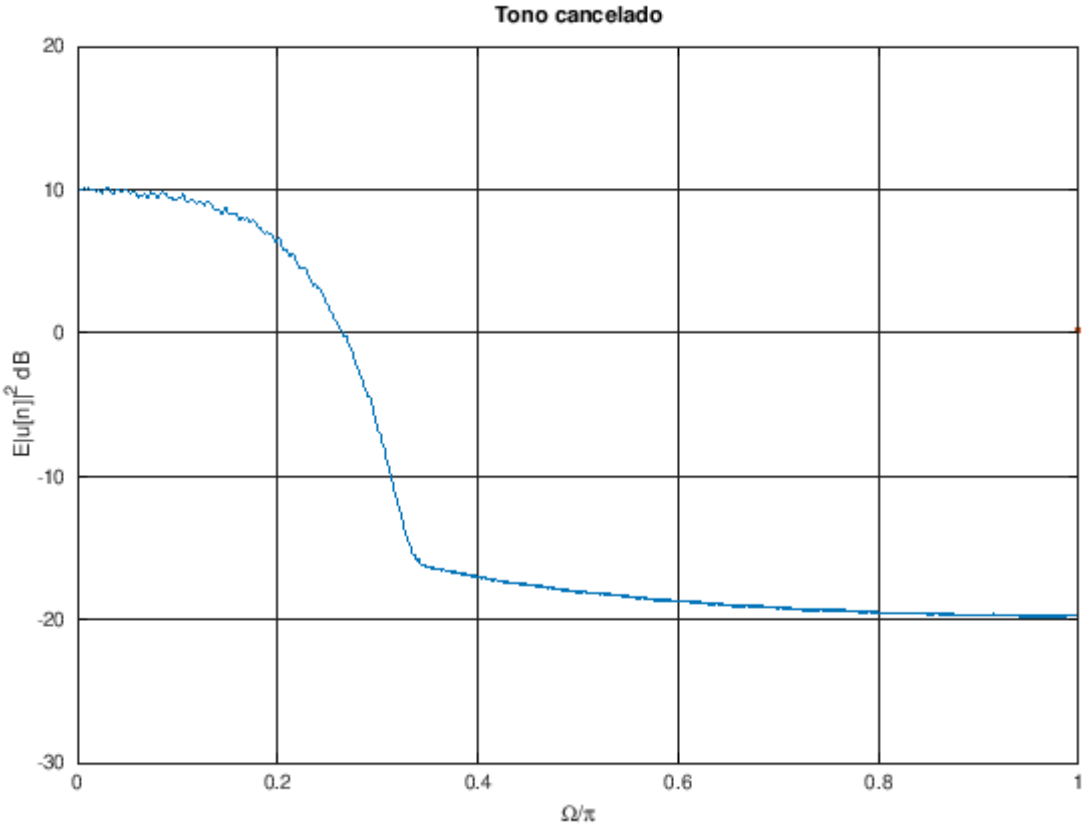
Es decir que si por alguna razón dos señales se mezclaran de ésta manera :

$$u[n] = p[n] + \tilde{p}[n]$$

No puede extraerse señal de sincronismo de $E\left\{|u[n]|^2\right\}$

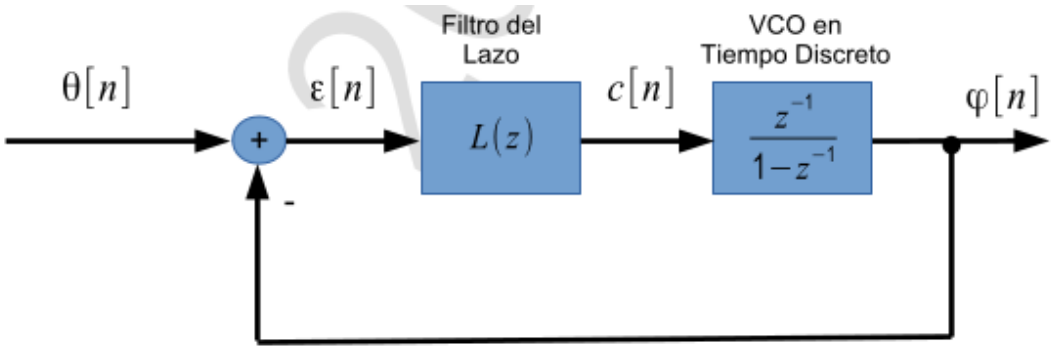
[36]:
```
M = 8;  %Factor de sobremuestreo
beta = .50001; %Factor de roll-off
%beta=0.9999999995;
fB = 32e9;        % Velocidad de simbolos (baud rate)
T = 1/fB; % Tiempo entre simbolos

L = 40;  % 2*L*M+1 es el largo del filtro sobremuestreado
t = [-L:1/M:L]*T;
gn = sinc(t/T).*cos(pi*beta*t/T)./(1-4*beta^2*t.^2/T^2);

ak = (2*randi([0,1],1,n_symbols)-1) +j*(2*randi([0,1],1,n_symbols)-1);
xn = zeros(1,n_symbols*M);
xn(1:M:end) = ak;




pn = conv(xn,gn);

bk = (2*randi([0,1],1,n_symbols)-1) +j*(2*randi([0,1],1,n_symbols)-1);
xxn = zeros(1,n_symbols*M);
xxn(1:M:end) = bk; % Independiente de xn

qn =conv(xxn,gn);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Genero la señal q tiene la demora en n_d=M/2%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n_d=M/2; % Si se modifica éste valor a algo distinto de M/2 se nota que el tono␣
 ↪no se cancela

pn_tilde=1.0*[zeros(1,n_d) qn(1:end-n_d)];

un=pn+pn_tilde ; %Sumo los dos canales
```

```
Z=abs(un).^2;

[pxx w]=pwelch(Z,1024,[],1024*2);
plot(2*w,10*log10(sqrt(pxx)),2*w(loc));
title('Tono cancelado')
ylabel('E{|u[n]|^{2}} dB')
xlabel('\Omega/\pi');

grid
```



4 Laboratorio N°3

En la siguiente figura se presenta un lazo de enganche de fase (PLL) en tiempo discreto:



Las consignas del laboratorio se listan a continación:

A. El error de régimen permanente se define por:

$$\epsilon_{ss} = \lim_{n\to\infty} \epsilon[n]$$

Utilizando el teorema del valor final :

$$x[\infty] = \lim_{z\to 1}(z-1)X(z)$$

Encontrar una expresión para ϵ_{ss} e función de $L(z)$ y $\Theta(z)$, suponiendo que $\epsilon[n] = 0$ para $n < 0$

B. Para

$$\theta[n] = \Omega_0 nu[n]$$

Demostrar que $\epsilon_{ss} = 0$ si $L(z)$ tiene un polo en $z = 1$

C. Para $L(z) = K_p$ (filtro proporcional). Verifique que la función de lazo cerrado es :

$$H(e^{j\Omega}) = \frac{K_p}{e^{j\Omega} + K_p - 1}$$

Con frecuencia de corte (-3dB) dada por $\Omega_c \approx K_p$

D. Analizar la respuesta del lazo para los siguientes filtros:

$$L_1(z) = z^{-D}K_p$$
$$L_2(z) = z^{-D}\left(K_p + K_i\frac{z^{-1}}{1-z^{-1}}\right)$$

donde D es la latencia del lazo.

E. Evalúe la establidad del lazo en función de la latencia D utilizando el diagrama de polos y ceros (e.g., función zplane de Matlab)

F. Seleccione valores de ganancias apropiados (e.g., asegurar estabilidad del lazo) y verifique la magnitud de la respuesta en frecuencia por medio de simulaciones Usar señales de entrada

$$\theta[n] = Acos(\Omega n)$$

4.1 Breve Introdución

Llevó un tiempo considerable adaptar éste código a Octave, pero sirvió para entender mejor cada bloque del código compartido por el docente.

El tema de PLL es inmenso, y ésto es sólo un grano de arena en un desierto, pero igualmente hacerlo funcionar bloque por bloque, primero en MATLAB y luego en su traslado a Octave resultó estimulante.

Para repasar el tema de PLL primero se hizo una breve lectura del libro de Comunicaciones recomendado por el docente (A partir de la **Página 700 del Messerschmitt** hay explicaciones muy importantes respecto a éste tema).

Luego de obtener las transformadas Z de las señales en la figura al inicio de éste apartado, podemos encontrar la función de transferencia a lazo cerrado haciendo :

4.1.1 Obtención de la función de transferencia

$$E(z) = \Theta(z) - \Phi(z)$$

$$G(z) = L(z)\frac{z^{-1}}{1 - z^{-1}}$$

Siendo $G(z)$ la respuesta a **lazo abierto**, dada por la cascada del VCO discreto y el filtro pasa-bajos $L(z)$

Usando las dos expresiones anteriores tenemos:

$$E(z)G(z) = \Phi(z)$$
$$(\Theta(z) - \Phi(z))G(z) = \Phi(z)$$

$$\Theta(z)G(z) - \Phi(z)G(z) = \Phi(z)$$

$$\Theta(z)G(z) = \Phi(z)(1 + G(z))$$

De donde finalmente obtenemos :

$$H(z) = \frac{\Phi(z)}{\Theta(z)} = \frac{G(z)}{1 + G(z)}$$

En la expresión anterior se ve claramente que si $G(z) = -1$ el sistema se vuelve inestable. Por lo que es a partir de el análisis de ésta condición que se evaluan el **margen-de-ganancia** (Ganancia por la que hay q multiplicar $|G(e^{j\Omega})|$ para llegar a 1 cuando e $\angle G(e^{j\Omega}) = -180°$) y el **margen-de-fase** (La fase que hay que sumarle a $\angle G(e^{j\Omega})$ para llegar a -180° cuando $|G(e^{j\Omega})| = 1$)

Operando usando las expresiones anteriores de $H(z)$ y $G(z)$ llegamos a la siguiente expresión:

$$H(z) = \frac{\Phi(z)}{\Theta(z)} = \frac{L(z)z^{-1}}{1 - z^{-1} + L(z)z^{-1}}$$

4.2 A. Error en régimen permanente

Teniendo en cuenta la definición:

$$\epsilon_{ss} = \lim_{n\to\infty} \epsilon[n]$$

y utilizando el teorema del valor final :

$$\epsilon_{ss} = \epsilon[\infty] = \lim_{z\to 1}(z-1)E(z)$$

Se nos pide encontrar una expresión para ϵ_{ss} en función de $L(z)$ y $\Theta(z)$, suponiendo que $\epsilon[n] = 0$ para $n < 0$. Ésta última suposición no es más que la condición necesaria para que sea válido el teorema.

Comenzamos recordando la expresión:

$$E(z)G(z) = \Phi(z)$$

Siendo $G(z) = L(z)\frac{z^{-1}}{1-z^{-1}}$ escribimos:

$$E(z)\left(L(z)\frac{z^{-1}}{1-z^{-1}}\right) = \Phi(z)$$

Recordando que :

$$E(z) = \Theta(z) - \Phi(z)$$

Despejamos $\Phi(z)$

$$\Phi(z) = \Theta(z) - E(z)$$

Y tenemos:

$$E(z)\left(L(z)\frac{z^{-1}}{1-z^{-1}}\right) = \Theta(z) - E(z)$$

De donde operamos:

$$E(z)\left(1 + L(z)\frac{z^{-1}}{1-z^{-1}}\right) = \Theta(z)$$

Y finalmente despejamos

$$E(z) = \frac{\Theta(z)}{\left(1 + L(z)\frac{z^{-1}}{1-z^{-1}}\right)}$$

Si aplicamos el teorema del valor final usando la expresión anterior tenemos:

$$\epsilon_{ss} = \epsilon[\infty] = \lim_{z\to 1}(z-1)E(z)$$

$$\epsilon_{ss} = \lim_{z\to 1}(z-1)\frac{\Theta(z)}{\left(1 + L(z)\frac{z^{-1}}{1-z^{-1}}\right)}$$

Operando y dejando la expresión completa en función de z tenemos

$$\epsilon_{ss} = \lim_{z\to 1}(z-1)\frac{\Theta(z)}{\left(1 + L(z)\frac{1}{z-1}\right)}$$

Y multiplicando numerador y denominador por $(z-1)$ nos queda:

$$\epsilon_{ss} = \lim_{z\to 1}\frac{(z-1)^2\Theta(z)}{(z-1) + L(z)}$$

4.3 B. Error en régimen permanente cuando la entrada es una rampa

4.3.1 Con el polo en z=1 en el filtro

Se nos pide demostrar que $\epsilon_{ss} = 0$ si $\theta[n] = \Omega_0 nu[n]$ y si $L(z)$ tiene un polo en $z = 1$
Si aplicamos transformada Z a $\theta[n]$ obtenemos mirando una tabla:

$$\Theta(z) = \frac{\Omega_0 z}{(z-1)^2}$$

Y si $L(z)$ tiene un polo en $z = 1$ significa que podemos expresarlo por ejemplo como:

$$L(z) = \frac{K_i}{z-1}$$

Uniendo las expresiones anteriores a la del error obtenida en el apartado anterior tenemos:

$$\epsilon_{ss} = \lim_{z\to 1}\frac{(z-1)^2\Theta(z)}{(z-1) + L(z)}$$

$$\epsilon_{ss} = \lim_{z\to 1}\frac{(z-1)^2\frac{\Omega_0 z}{(z-1)^2}}{(z-1) + \frac{K_i}{z-1}}$$

Simplificando obtenemos:

$$\epsilon_{ss} = \lim_{z\to 1} \frac{\Omega_0 z}{(z-1) + \frac{K_i}{z-1}}$$

$$\epsilon_{ss} = \lim_{z\to 1} \frac{\Omega_0 z(z-1)}{(z-1)^2 + K_i} = \frac{0}{K_i} = 0$$

Con lo que hemos logrado la demostración solicitada.

4.3.2 Error para misma entrada usando filtro proporcional

En éste caso comprobamos para la misma entrada que pasa si $L(z) = K_p$

$$\epsilon_{ss} = \lim_{z\to 1} \frac{(z-1)^2 \frac{\Omega_0 z}{(z-1)^2}}{(z-1) + K_p}$$

$$\epsilon_{ss} = \lim_{z\to 1} \frac{\Omega_0 z}{(z-1) + K_p}$$

$$\epsilon_{ss} = \frac{\Omega_0}{K_p}$$

En éste caso vemos que sin existir el polo en z=1 si existe error en estado estacionario, a diferencia del apartado anterior.

4.4 C. Verificación de la respuesta en frecuencia para filtro proporcional

En la breve introducción obtuvimos la función de transferencia

$$H(z) = \frac{\Phi(z)}{\Theta(z)} = \frac{L(z)z^{-1}}{1 - z^{-1} + L(z)z^{-1}}$$

Multiplicando por z numerador y denominador obtenemos

$$H(z) = \frac{L(z)}{z + L(z) - 1}$$

De la cual si hacemos $z = e^{j\Omega}$ (transformada de Fourier) y $L(z) = K_p$ (Filtro proporcional) nos queda

$$H(e^{j\Omega}) = \frac{K_p}{e^{j\Omega} + K_p - 1}$$

Se nos pide verificar que la frecuencia de corte (-3dB) está dada por $\Omega_c \approx K_p$

Si calculamos el módulo al cuadrado de la expresión anterior valuada en Ω_c tenemos:

$$|H(e^{j\Omega_c})|^2 = \left\| \frac{K_p}{e^{j\Omega_c} + K_p - 1} \right\|^2 = \left\| \frac{K_p}{\cos(\Omega_c) + j\sin(\Omega_c) + K_p - 1} \right\|^2 \approx \left\| \frac{K_p}{K_p + j\Omega_c} \right\|^2, \quad |\Omega_c| \ll 1$$

Debido a que para $|\Omega_c| \ll 1$:

$$\cos(\Omega_c) \approx 1$$

$$\sin(\Omega_c) \approx \Omega_c$$

Partiendo desde la expresión obtenida

$$|H(e^{j\Omega_c})|^2 = \left\| \frac{K_p}{K_p + j\Omega_c} \right\|^2$$

Hacemos $\Omega_c = K_p$

$$|H(e^{j\Omega_c})|^2 = \left\| \frac{K_p}{K_p + jK_p} \right\|^2$$

Que simplificando queda:

$$|H(e^{j\Omega_c})|^2 = \left\| \frac{1}{1 + j} \right\|^2$$

Lo que es equivalente a escribir:

$$|H(e^{j\Omega_c})|^2 = \left(\frac{1}{\sqrt{2}} \right)^2 = \frac{1}{2}$$

Lo que en decibeles nos queda (luego de multiplicar por 10 y aplicar logaritmo en ambos lados):

$$20log\left(|H(e^{j\Omega_c})|\right) \approx -3dB$$

Por lo que queda demostrado que la frecuencia de corte de $-3dB$ es $\Omega_c \approx K_p$

4.4.1 Filtro equivalente en tiempo continuo

Hemos visto que si comprimimos en un factor T (período de muestreo) el eje de frecuencias discreto en la representación de la transformada de Fourier del filtro de tiempo discreto, nos da su equivalente en tiempo continuo:

$$H_c(j\omega) = H(e^{j\omega T})$$

Con $\Omega_c = 2\pi f_c T = \omega_c T \approx K_p$

4.5 D. Respuesta en frecuencia para distintos filtros

Se nos pide analizar la respuesta del lazo para los siguientes filtros:

$$L_1(z) = z^{-D} K_p$$
$$L_2(z) = z^{-D} \left(K_p + K_i \frac{z^{-1}}{1 - z^{-1}} \right)$$

D es la latencia del lazo en ciclos de reloj . Ésta latencia es un modelado de las demoras de procesamiento posibles en una implementación real del sistema, que son posibles causas de inestabilidad del sistema como se verá en las simulaciones.

El primero es un filtro proporcional y el segundo tiene además una componente integral.

Partiendo de la función de tranferecia obtenida en la introdución:

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

Para $L_1(z)$ reemplazamos

$$L_1(z) = z^{-D} K_p$$

$$H_1(z) = \frac{z^{-D} K_p z^{-1}}{1 - z^{-1} + z^{-D} K_p z^{-1}}$$

Lo que operando nos deja:

$$H_1(z) = \frac{K_p z^{-(D+1)}}{1 - z^{-1} + K_p z^{-(D+1)}}$$

Para $L_2(z)$ reemplazamos

$$L_2(z) = z^{-D} \left(K_p + K_i \frac{z^{-1}}{1 - z^{-1}} \right)$$

$$H_2(z) = \frac{z^{-D} \left(K_p + K_i \frac{z^{-1}}{1 - z^{-1}} \right) z^{-1}}{1 - z^{-1} + z^{-D} \left(K_p + K_i \frac{z^{-1}}{1 - z^{-1}} \right) z^{-1}}$$

Lo que multiplicando numerador y denominador por $(1 - z^{-1})$ nos deja con

$$H_2(z) = \frac{z^{-(D+1)} \left(K_p (1 - z^{-1}) + K_i z^{-1} \right)}{(1 - z^{-1})^2 + z^{-(D+1)} \left(K_p (1 - z^{-1}) + K_i z^{-1} \right)}$$

Lo que operando nos deja luego de expandir y distribuir:

$$H_2(z) = \frac{z^{-(D+1)}K_p + (K_i - K_p)\, z^{-(D+2)}}{1 - 2z^{-1} + z^{-2} + K_p z^{-(D+1)} + (K_i - K_p)\, z^{-(D+2)}}$$

Son éstas expresiones de $H_1(z)$ y $H_2(z)$ las que usaremos en matlab para analizar estabilidad, analizando si el círculo unitario queda incluído en la región de convergencia.

4.5.1 Simulaciones en Octave de la respuesta del lazo

En ésta simulación graficaremos la respuesta en frecuencia del PLL para los casos $|H_1(j\omega)|$ y $|H_2(j\omega)|$, en tiempo-continuo .

El código propuesto por el docente comienza definiendo el ancho de banda B_l en Hz. Define una frecuencia de muestreo f_s y inicialmente usa un valor de latencia $D = 0$ para analizar el sistema en condiciones ideales.

La ganancia proporcional K_p está normalizada de la siguiente manera:

$$K_p = \frac{2\pi B_l}{f_s}$$

Es en la expresión anterior que vemos que elegiremos la frecuencia de muestreo lo suficientemente alta como para que K_p sea mucho menor que uno caso en el cual es válida su equivalencia con la frecuencia de corte como se vio en apartados anteriores.

Y se define inicialmente la ganancia integral K_i de la siguiente manera:

$$K_i = \frac{K_p}{10000}$$

Vale la pena notar de que en la expresión de $H_2(z)$ del apartado anterior se ve que la ganancia $K_i - K_p$ es la que produce modificaciones en la función de transferencia, por lo que tiene sentido definir K_i en función de K_p ya que de ésta manera estamos definiendo un factor dada por la mencionada diferencia entre ambos.

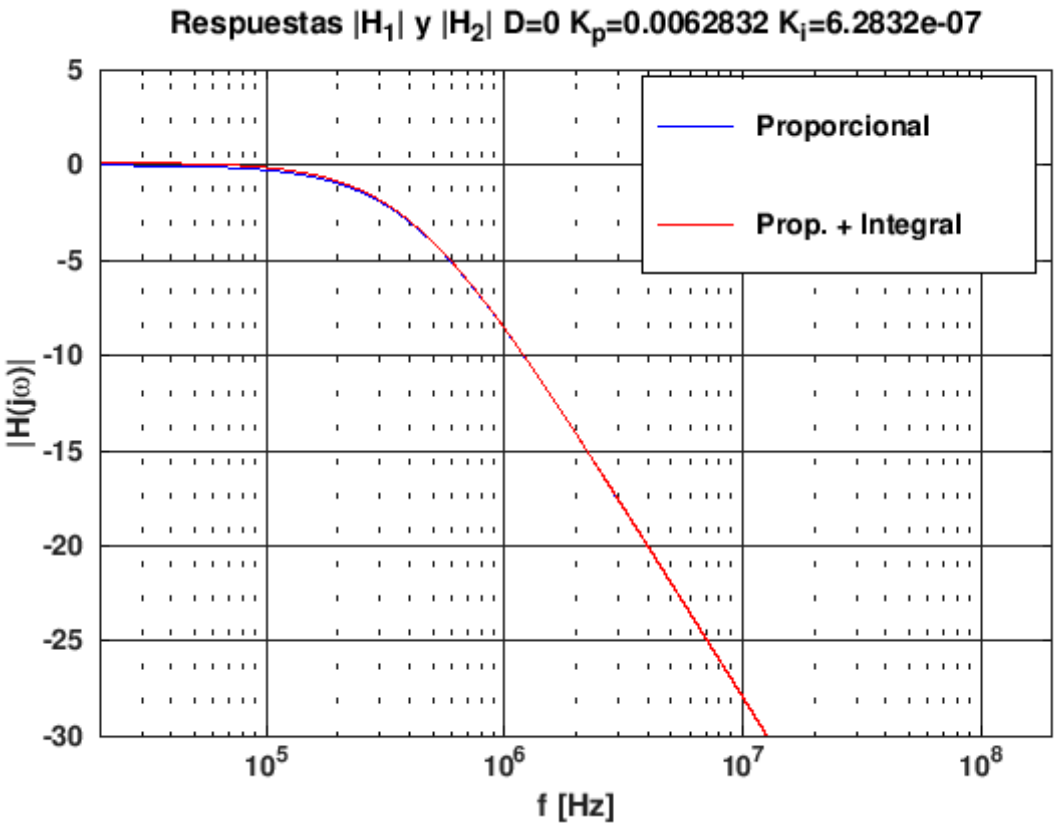
4.5.2 Función para simulación del lazo variando los parámetros

Al código provisto por el docente se lo ha modificado para su uso en esta jupyter-notebook y para hacerlo compatible con Octave. Además el código presentado a continuación genera una función que podemos llamar más adelante para simular variando los parámetros.

[37]:
```
Bl=.40e6;                % Ancho de banda, en Hz
fs=400e6;                % Frecuencia de muestreo, in Hz
D=0;                     % Latencia en ciclos de reloj de muestreo
Kp=Bl/fs*2*pi;    % Ganancia proporcional Normalizada para el ancho de banda y␣
 ↪la frecuencia de muestreo
Ki=Kp*1e-4*1;   % Ganancia integral a Modificar para analizar modificaciones en␣
 ↪la respuesta
color_h1='b';
```

```
color_h2='r';
% Definimos como función para volver a usar

function respuesta_lazo(Bl,fs,D,Kp,Ki,color_h1,color_h2)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%Definiciones Funciones de Transferencia%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

omega=[1e-5:1e-5:1]*pi;
z=exp(j*omega);


L1=z.^(-D)*Kp; % Filtro proporcional y latencia

L2=z.^(-D).*(Kp+Ki*z.^(-1)./(1-z.^(-1))); % Filtro proporcional Integral y␣
 ↪Latencia

VCO=z.^(-1)./(1-z.^(-1)); % Transformada Z de la función de transferencia del␣
 ↪VCO

G1=L1.*VCO; % Respuesta Lazo Abierto para L1(P)
G2=L2.*VCO; % Respuesta Lazo Abierto para L2 (P+I)

H1=G1./(1+G1); % Respuesta a Lazo Cerrado para L1
H2=G2./(1+G2); % Respuesta a Lazo Cerrado para L2

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%Plot de Respuesta en Frecuencia para H1 y H2%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
f_hertz=omega*fs/(2*pi); % Comprimo el Eje en T_s y expreso en hertz(divido por␣
 ↪2pi)
h=plot(f_hertz,20*log10(abs(H1)),color_h1,f_hertz,20*log10(abs(H2)),color_h2);

% ajustes del plot


set(h,'Linewidth',3);
set(h,'Markersize',16);
set(gca,'XScale','log','YScale','lin','FontWeight','bold','FontSize',14);
set(gca,'Linewidth',2);
legend(' Proporcional',' Prop. + Integral')
xlabel('f [Hz]');
ylabel('|H(j\omega)|');
str=sprintf('Respuestas |H_1| y |H_2| D=%s K_p=%s␣
 ↪K_i=%s',num2str(D),num2str(Kp),num2str(Ki)) ;

title(str)
```

```
grid on
axis([2*10^4 fs/2 -30 5])
endfunction

respuesta_lazo(Bl,fs,D,Kp,Ki,color_h1,color_h2)
```



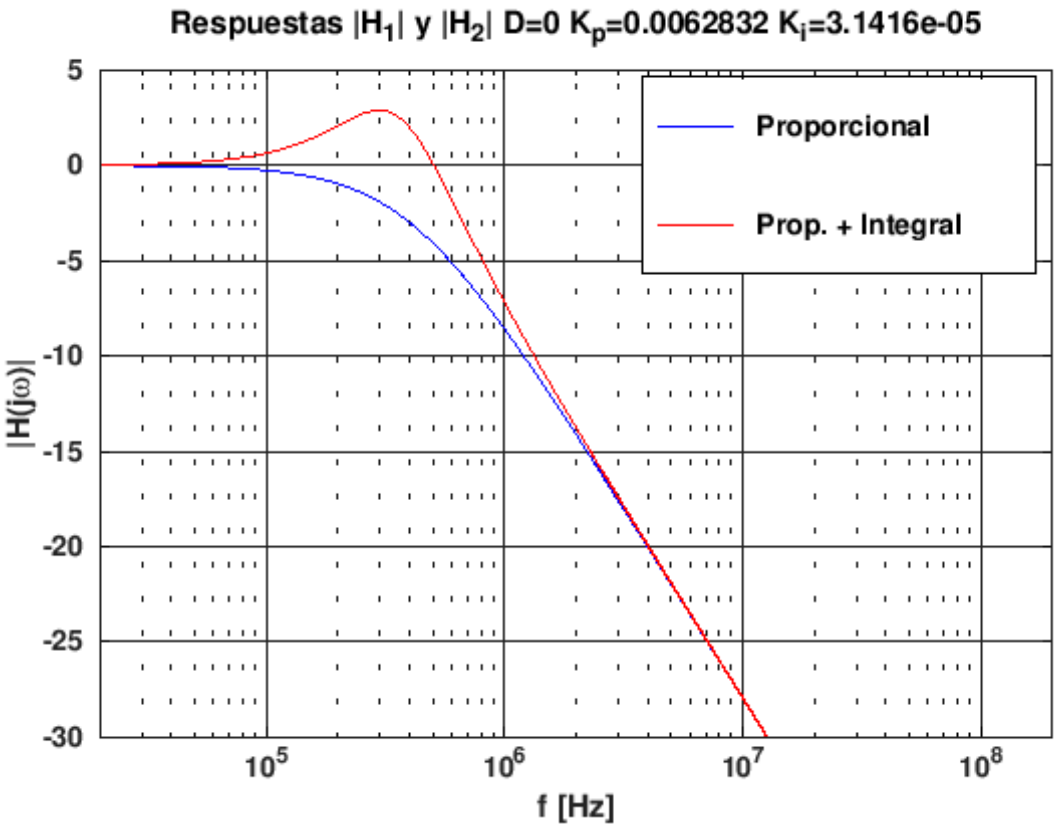
4.5.3 Simulaciones de respuesta del lazo para variaciones en los parámetros

Aquí aprovecharemos que hemos definido la parte anterior del código como una función y la llamaremos para distintos valores de entrada, analizando así los cambios en el ancho de banda al modificar la latencia y las ganancias.

Primer caso en el que no coinciden $|H_1(j\omega)|$ y $|H_2(j\omega)|$ Es igual al caso anterior con la diferencia de que :

$$K_i = \frac{K_p}{200}$$

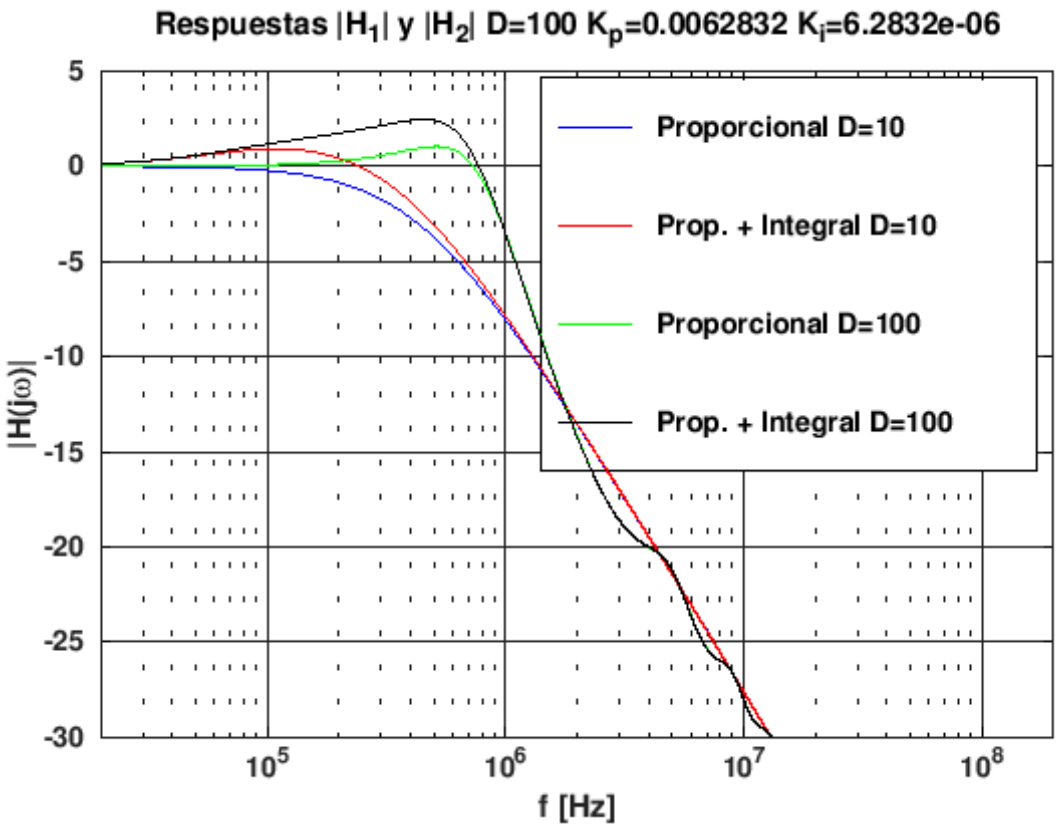
[38]:
```
Bl=.40e6;                % Ancho de banda, en Hz
fs=400e6;                % Frecuencia de muestreo, in Hz
D=0;                     % Latencia en ciclos de reloj de muestreo
Kp=Bl/fs*2*pi;     % Ganancia proporcional Normalizada para el ancho de banda y␣
 ↪la frecuencia de muestreo
Ki=Kp/200;   % Ganancia integral a Modificar para analizar modificaciones en la␣
 ↪respuesta
color_h1='b';
color_h2='r';

respuesta_lazo(Bl,fs,D,Kp,Ki,color_h1,color_h2)
```



Agregado de latencias al caso anterior Usando $K_i = K_p/1000$ comparamos para una latencia de $D = 10$ (Azul H_1 y Rojo H_2)y $D = 100$ (verde H_1 y negro H_2)

Como veremos con mayor detalle en los diagramas de polos y ceros, al aumentar la latencia aparecen nuevos polos que cada vez se acercan más al círculo unitario, pudiendo llegar a hacer inestable al sistema. En éstos gráficos esa tendencia a la inestabilidad se nota en los picos que se generan en la respuesta.

[39]:
```
Bl=.40e6;                % Ancho de banda, en Hz
fs=400e6;                % Frecuencia de muestreo, in Hz
D=10;                     % Latencia en ciclos de reloj de muestreo
Kp=Bl/fs*2*pi;    % Ganancia proporcional Normalizada para el ancho de banda y␣
 ↪la frecuencia de muestreo
Ki=Kp/1000;   % Ganancia integral a Modificar para analizar modificaciones en␣
 ↪la respuesta
color_h1='b';
color_h2='r';

respuesta_lazo(Bl,fs,D,Kp,Ki,color_h1,color_h2)
hold on
D=100;
color_h1='g';
color_h2='k';
respuesta_lazo(Bl,fs,D,Kp,Ki,color_h1,color_h2)
legend(' Proporcional D=10',' Prop. + Integral D=10',' Proporcional D=100','␣
 ↪Prop. + Integral D=100')
hold off
```



4.6 E. Simulación diagrama de polos y ceros

En éste bloque del programa, hemos cambiado completamente el código provisto por el docente, por una versión que nos permite ver las funciones de transferencia y aplicarles herramientas incluídas en el paquete de control como rlocus (para graficar el lugar de las raíces para ganancias variables, lo que nos permite ver el efecto de aumentar la ganancia de un vistazo) y también step (que nos permite ver la respuesta al escalón de los sistemas, lo que es muy útil a la hora de analizar el error en estado estacionario).

4.6.1 Función de Transferencia Simbólica con función MATLAB/OCTAVE TF

Como para éste caso ya hemos obtenido las expresiones para la respuesta del lazo $H_1(z)$ y $H_2(z)$, vamos a expresarlas en función de potencias positivas (para que se vean mejor en Octave) y luego usaremos la función TF para generar las funciones de transferencia para ambos casos.

Es decir:

$$H_1(z) = \frac{K_p z^{-(D+1)}}{1 - z^{-1} + K_p z^{-(D+1)}}$$

Que experesada en potencias positivas de z sería:

$$H_1(z) = \frac{K_p}{z^{(D+1)} - z^D + K_p}$$

Para el segundo filtro tenemos:

$$H_2(z) = \frac{z^{-(D+1)} K_p + (K_i - K_p)\, z^{-(D+2)}}{1 - 2z^{-1} + z^{-2} + K_p z^{-(D+1)} + (K_i - K_p)\, z^{-(D+2)}}$$

Que expresado en potencias positivas de z queda:

$$H_2(z) = \frac{zK_p + (K_i - K_p)}{z^{(D+2)} - 2z^{(D+1)} + z^D + K_p z + (K_i - K_p)}$$

4.6.2 Primera simulación sin modelado de latencia

Aquí generamos la función de transferencia para una latencia $D = 0$

[40]:
```
%% Ésta función genera la función de tranferencia simbólicamente.

function [H1,H2,L1,L2,G1,G2] =  generar_ft(Bl,fs,D,Kp,Ki)

    z=tf('z',1/fs); % Para tiempo discreto es fundamental aclarar la frecuencia␣
↪de muestreo

    L1=z^(-D)*Kp; % Filtro proporcional y latencia
```

```
    L2=z^(-D)*(Kp+Ki*z^(-1)/(1-z^(-1))); % Filtro proporcional Integral y␣
 ↪Latencia

    VCO=z^(-1)/(1-z^(-1)); % Transformada Z de la función de transferencia del␣
 ↪VCO

    G1=L1*VCO; % Respuesta Lazo Abierto para L1(P)
    G2=L2*VCO; % Respuesta Lazo Abierto para L2 (P+I)

   % H1=G1/(1+G1); % Respuesta a Lazo Cerrado para L1
    H1=Kp/(z^(D+1)-z^D+Kp); % La escribo de ésta manera para poder verla mejor
    %H2=G2/(1+G2); % Respuesta a Lazo Cerrado para L2
    H2=(z*Kp+Ki-Kp)/(z^(D+2)-2*z^(D+1)+z^D+Kp*z+Ki-Kp);% La escribo de ésta␣
 ↪manera para poder verla mejor
endfunction
```

[41]:
```
Bl=.40e6;               % Ancho de banda, en Hz
fs=400e6;               % Frecuencia de muestreo, in Hz
D=0;                    % Latencia en ciclos de reloj de muestreo
Kp=Bl/fs*2*pi;    % Ganancia proporcional Normalizada para el ancho de banda y␣
 ↪la frecuencia de muestreo
Ki=Kp*1e-4*1.0;   % Ganancia integral a Modificar para analizar modificaciones␣
 ↪en la respuesta

[H1,H2,L1,L2,G1,G2]=generar_ft(Bl,10000*fs,D,Kp,Ki);
H1
```

```
Transfer function 'H1' from input 'u1' to output …

       0.006283
 y1:  ----------
      z - 0.9937

Sampling time: 2.5e-13 s
Discrete-time model.
```

[42]:
```
H2
```

```
Transfer function 'H2' from input 'u1' to output …

      0.006283 z - 0.006283
 y1:  ---------------------
      z^2 - 1.994 z + 0.9937
```

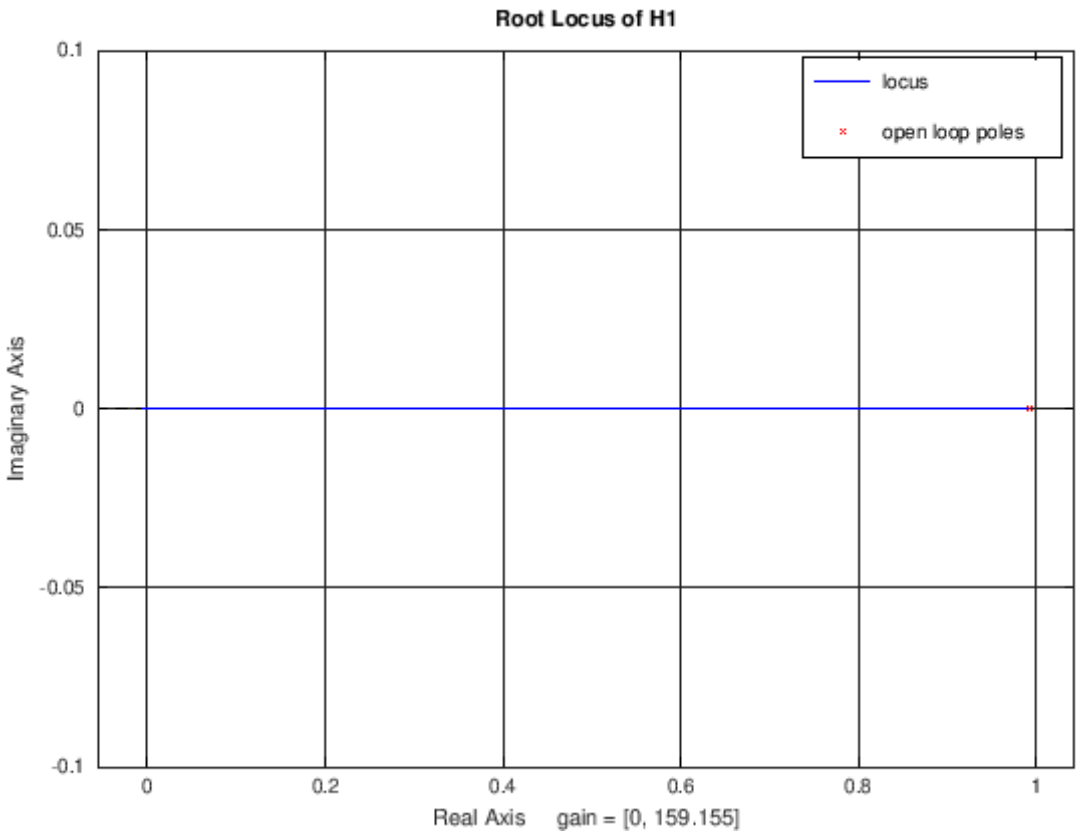
```
Sampling time: 2.5e-13 s
Discrete-time model.
```

Diagramas de polos y ceros sin latencia Aquí para el ejemplo inicial sin latencia vemos los diagramas de ceros y polos, y con rlocus analizamos el efecto de modificar la ganancia en función de K_p para $H_1(z)$ y $(K_i - Kp)$ para $H_2(z)$.

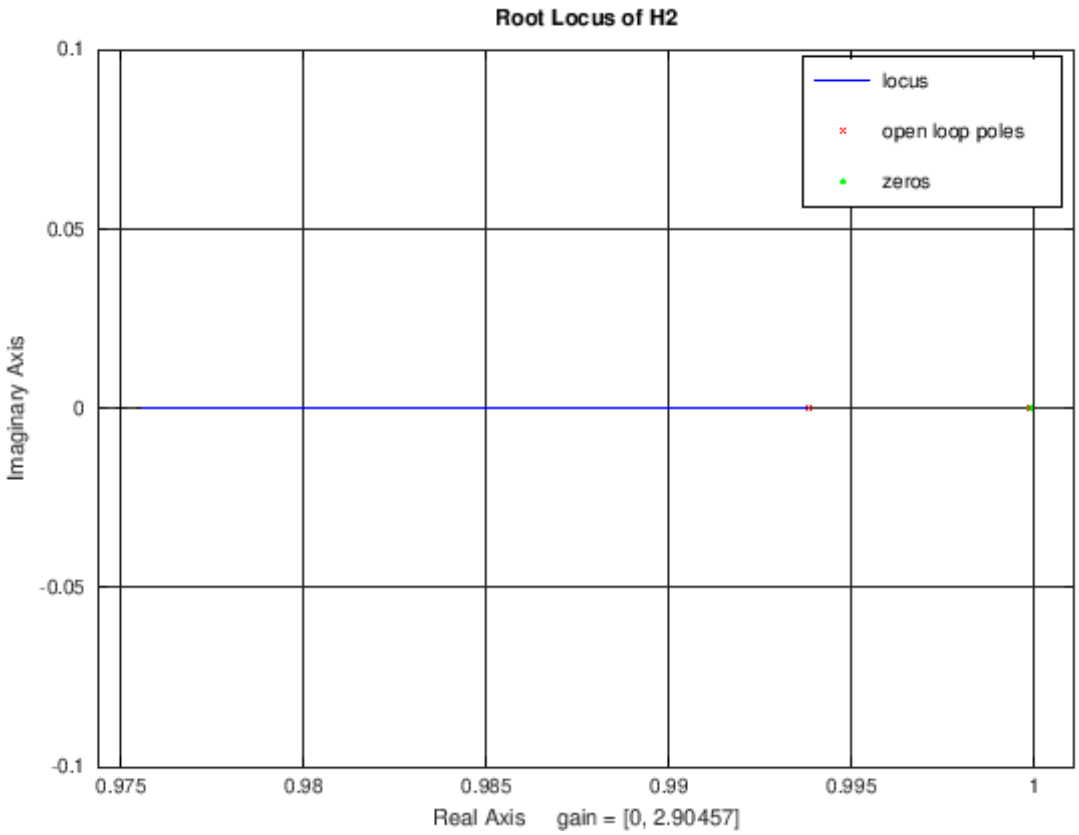
Tener en cuenta de que dice polos a lazo abierto (open loop poles) en la leyenda, ésto es incorrecto para nuestro caso ya que estamos viendo los polos de la respuesta a lazo cerrado en relidad, ya que al generar la función de transferecia lo hicimos usando las expresiones finales de $H_1(z)$ y $H_2(z)$. (la leyenda dice eso porque, como se puede ver leyendo la documentación, es de uso común en sistemas de control con retroalimentación en los que se analiza comúnmente los polos a lazo abierto)

pzmap es la función usada para el resto de diagramas de polos y ceros, que es equivalente pero no muestra el efecto de aumentar la ganancia.

[43]:
```
rlocus(H1);% Aquí veremos sólo el polo en z=1
```



[44]:
```
rlocus(H2); % Aquí vemos el polo y el cero en H2 para D=0
```



Respuesta al escalón sin latencia Como se aprende en Sistemas de Control, una forma de observar el error en estado estacionario de un sistema es aplicarle un escalón y ver su respuesta para $t \to \infty$

4.6.3 Simulación agregando latencia $D = 10$ con ganancia tal que sea estable

Veremos que aumenta la cantidad de polos , pero al estar normalizada la ganancia K_p , se mantiene estable

[45]:
```
D=10;

[H1,H2,L1,L2,G1,G2]=generar_ft(Bl,10000*fs,D,Kp,Ki);
```

[46]:
```
H1
```

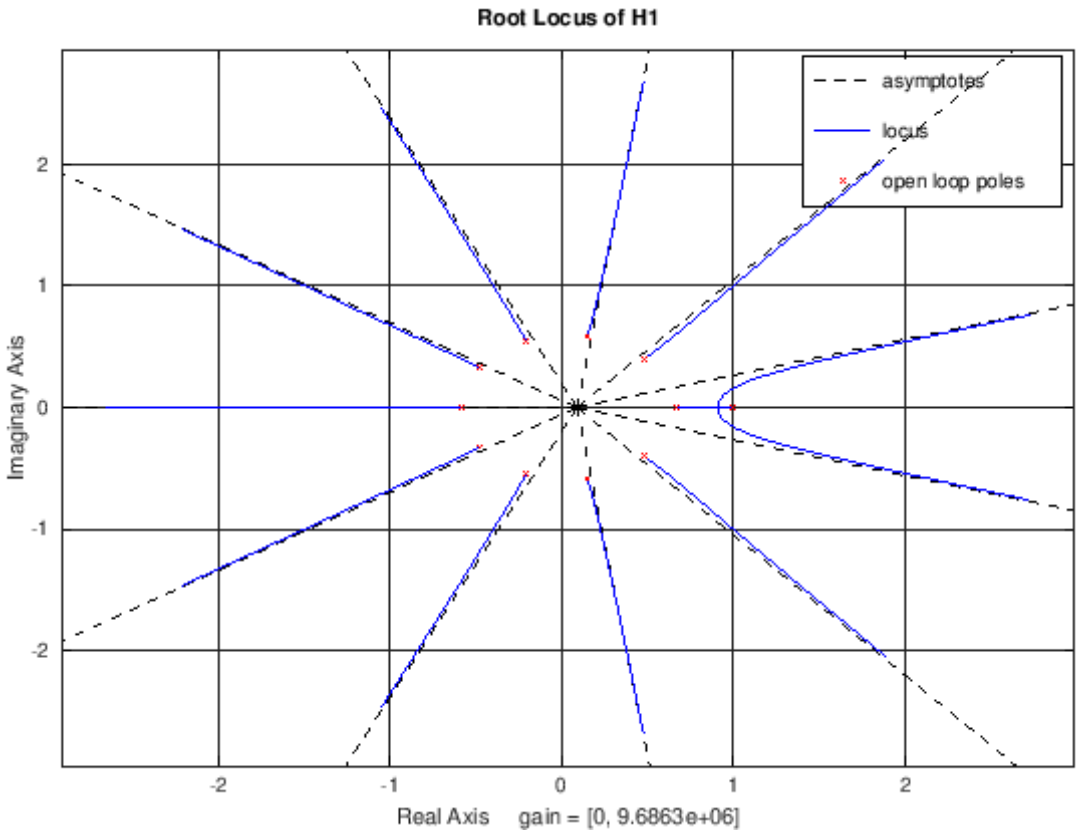
```
Transfer function 'H1' from input 'u1' to output …

             0.006283
 y1:  ----------------------
```

```
      z^11 - z^10 + 0.006283

Sampling time: 2.5e-13 s
Discrete-time model.
```

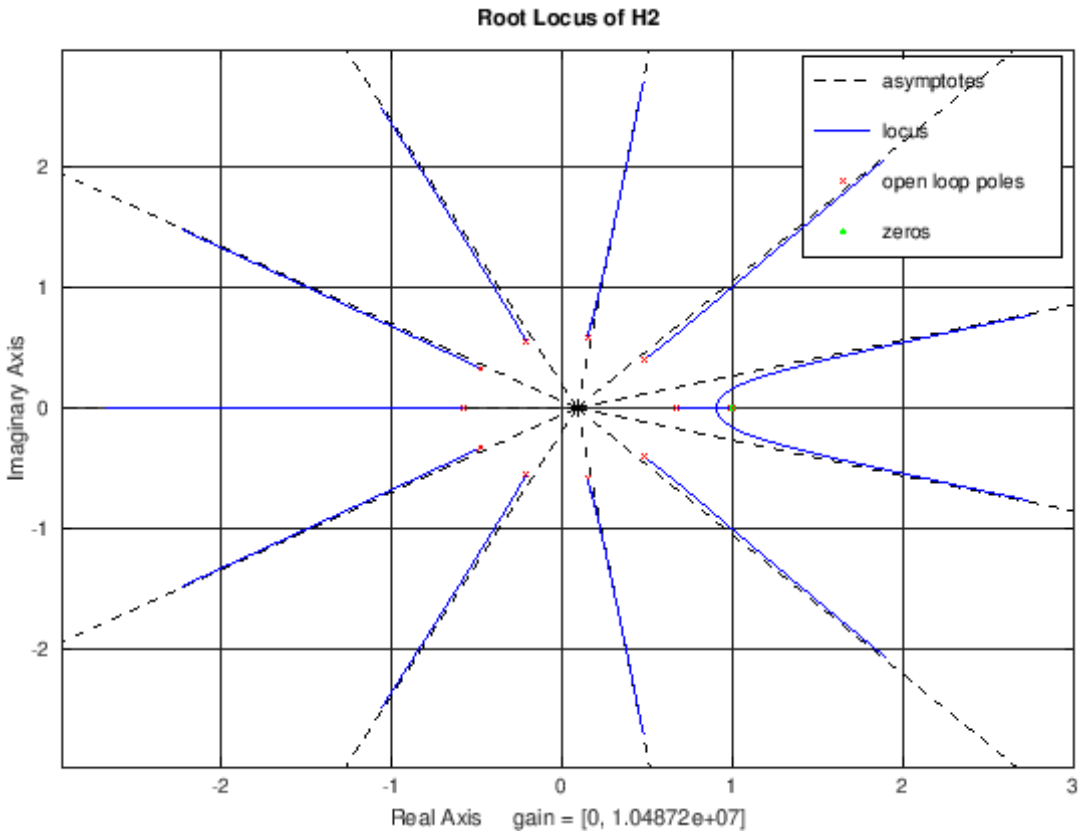
[47]:
```
H2
```

```
Transfer function 'H2' from input 'u1' to output …

                0.006283 z - 0.006283
 y1:  ----------------------------------------
      z^12 - 2 z^11 + z^10 + 0.006283 z - 0.006283

Sampling time: 2.5e-13 s
Discrete-time model.
```

[48]:
```
rlocus(H1);
```



[49]:
```
rlocus(H2)
```



4.6.4 Simulación agregando latencia $D = 10$ con ganancia tal que sea inestable

Al usar **rlocus** vemos ya el comportamiento asimptótico de el efecto de modificar la ganancia.

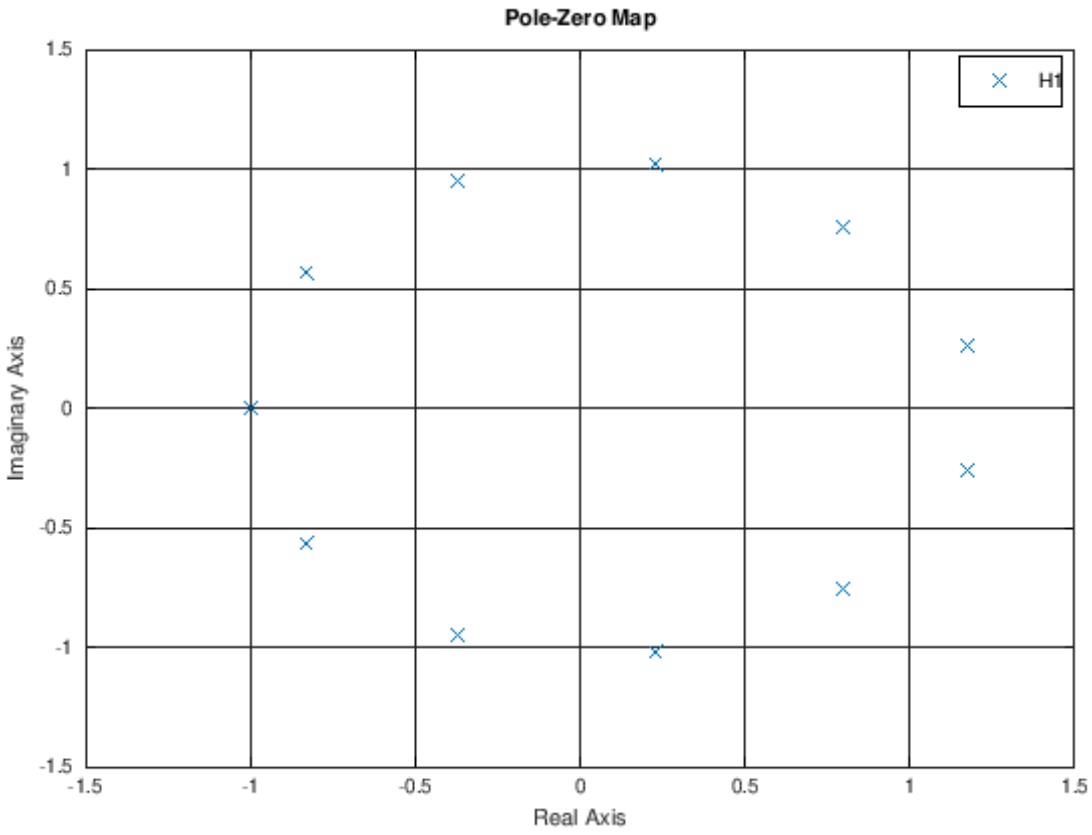
En MATLAB se puede usar **rltool** que diréctamente nos permite mediante una interfáz gráfica modificar la ganancia y ver el comportamiento de estabilidad. Para comprobar que los polos quedarían fuera del círculo unitario para un valor de ganancia no normalizado hacemos:

$k_p = 2$

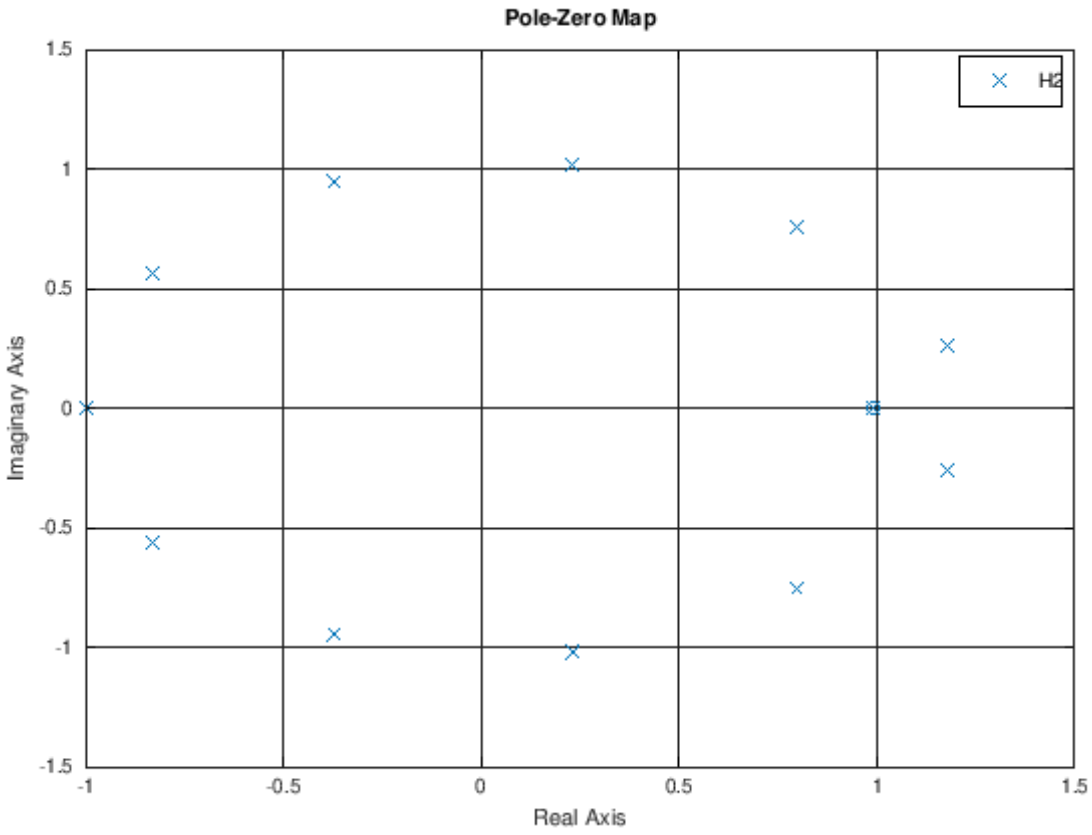
Veremos cómo ya comienzan a aparecer polos fuera del círculo unitario.

[50]:
```
D=10;
Kp=2;
Ki=Kp/100;
[H1,H2,L1,L2,G1,G2]=generar_ft(Bl,10000*fs,D,Kp,Ki);
```

[51]:
```
pzmap(H1);
```


Pole-Zero Map
H1
Imaginary Axis
Real Axis
1.5
1
0.5
0
-0.5
-1
-1.5
-1.5
-1
-0.5
0
0.5
1
1.5

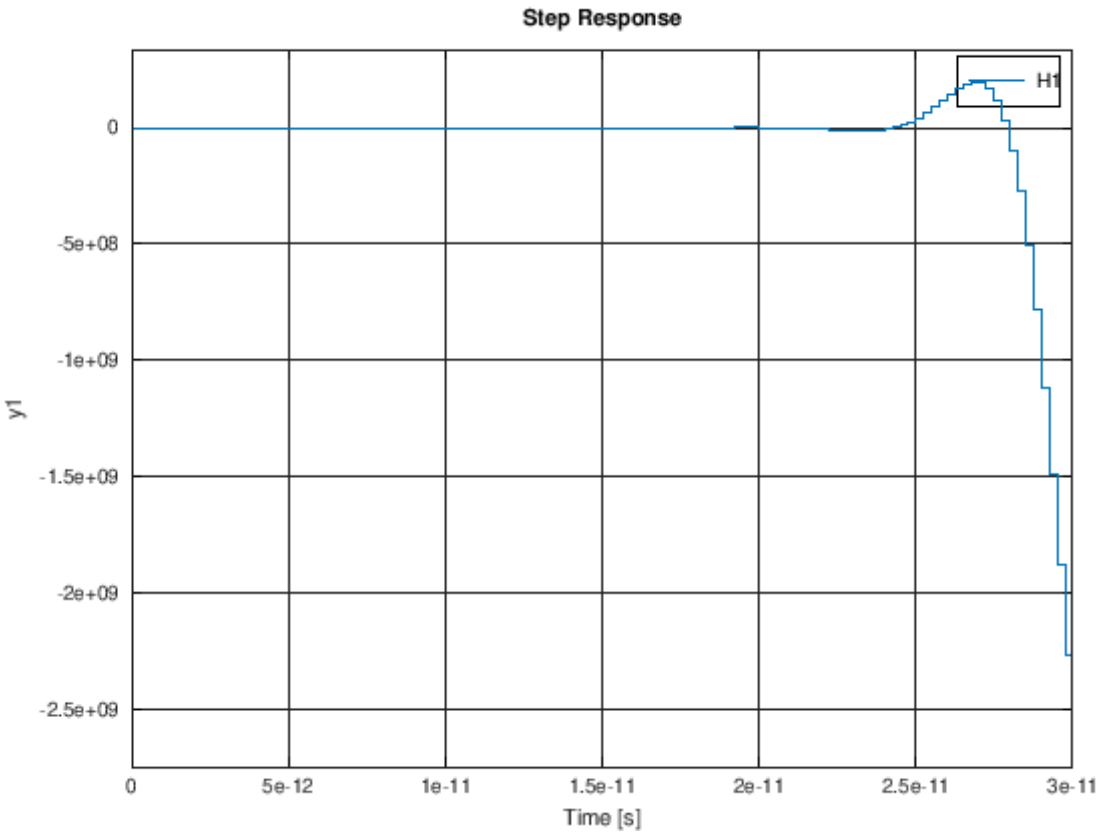
[52]:
```
pzmap(H2)
```



4.6.5 Respuesta al escalón para observar inestabilidad

Vemos que para el caso inestable el error en estado estacionario aumenta.

[53]:
```
step(H1)
```

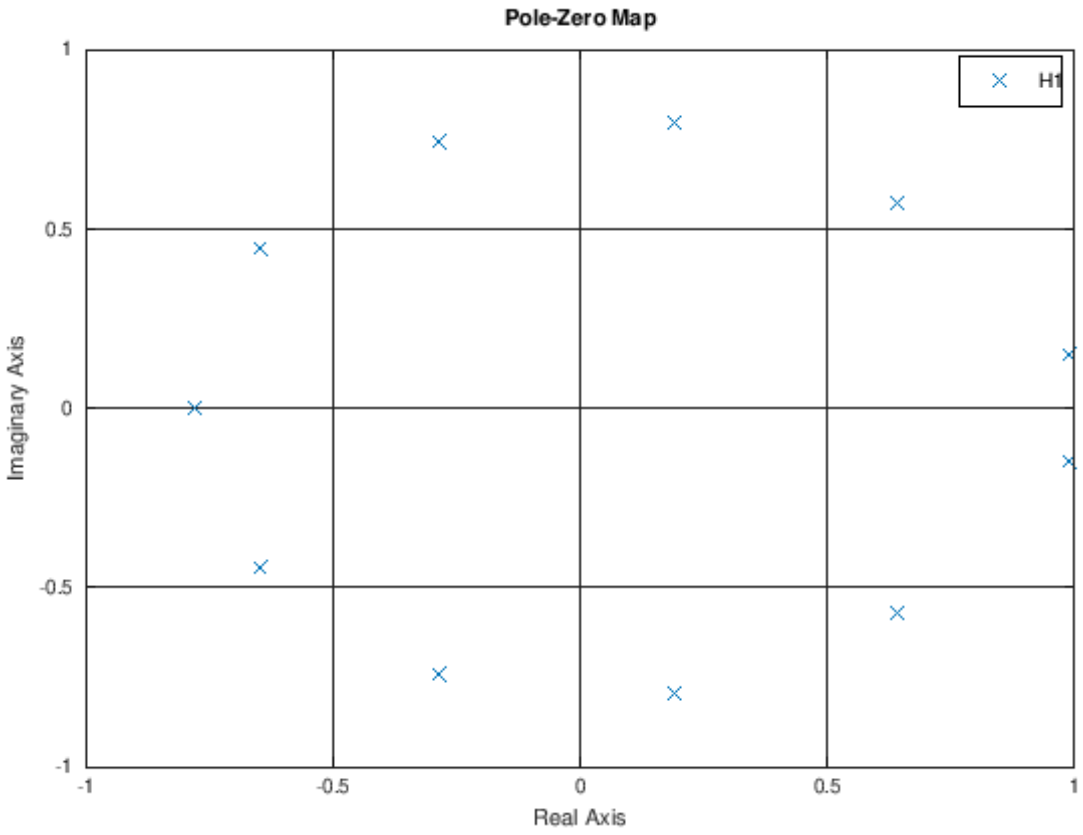


4.6.6 Ejemplo límite de estabilidad

Luego de hacer ciertas pruebas para la latencia seleccionada, y ver en la **página 712** del Messerschmit el **Ejemplo 14-10** probamos con un valor de K_p máximo en el que aún hay estabilidad. (El ejemplo del libro no tenía latencia por eso aquí K_p,para mantener la estabilidad, tiene que ser unas 10 veces menor aproximadamente)

[54]:
```
D=10;
Kp=0.15;
Ki=Kp/100;
[H1,H2,L1,L2,G1,G2]=generar_ft(Bl,10000*fs,D,Kp,Ki);
```

[55]:
```
pzmap(H1);
```

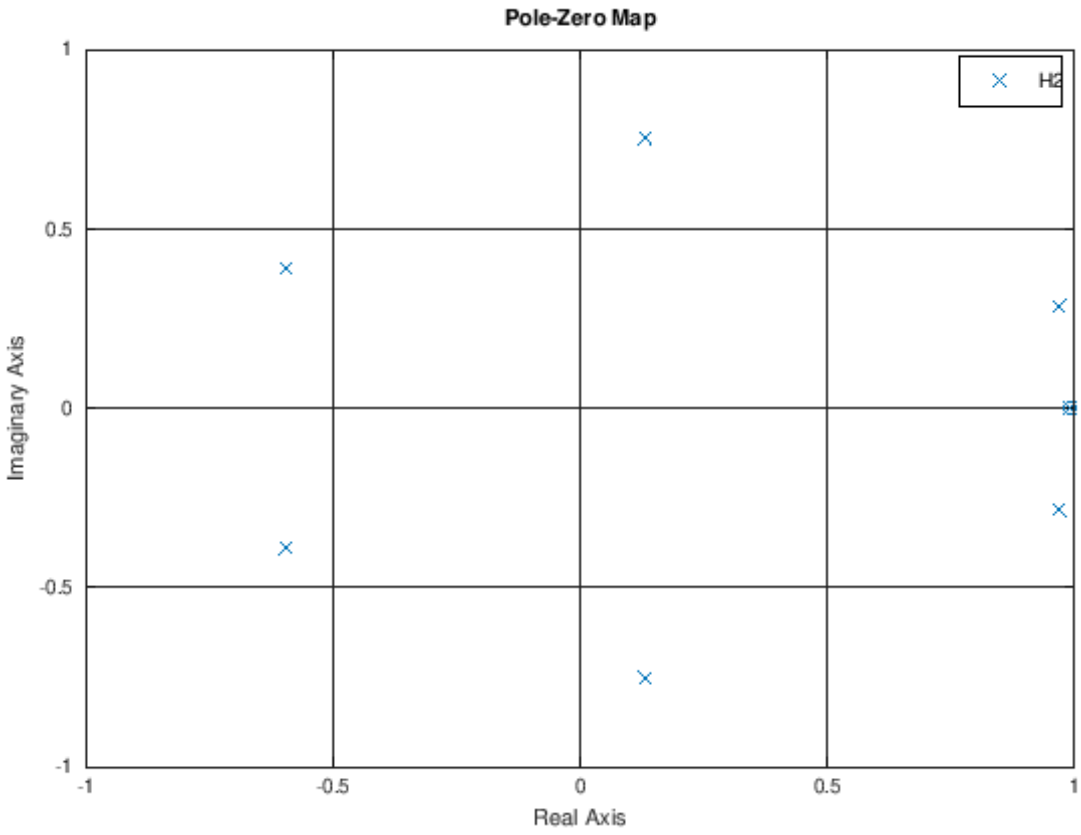


4.6.7 Ejemplo bajando latencia con K_p máximo manteniendo estabilidad

[56]:
```
D=5;
Kp=0.3;
Ki=Kp/100;
[H1,H2,L1,L2,G1,G2]=generar_ft(Bl,fs,D,Kp,Ki);
```

[57]:
```
H2
pzmap(H2);
```

```
Transfer function 'H2' from input 'u1' to output …

                0.3 z - 0.297
 y1:  ---------------------------------
      z^7 - 2 z^6 + z^5 + 0.3 z - 0.297

Sampling time: 2.5e-09 s
Discrete-time model.
```



4.6.8 Diagramas de bode para ver margen de ganancia y margen de fase

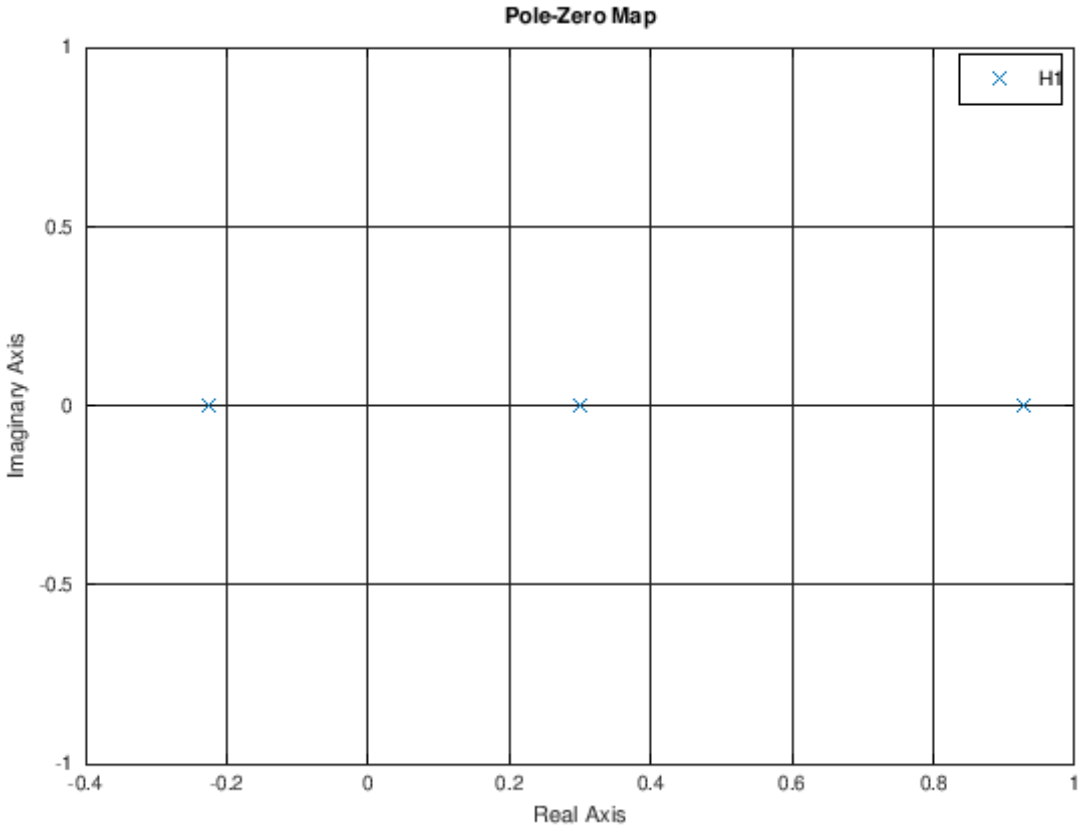
Lo que hicimos en el primer apartado de simulación para ver la respuesta en frecuencia con el código compartido por el docente, lo podemos hacer ahora mucho mas simple con la función de transferencia que hemos generado.

Tener en cuenta que aquí las frecuencias están en rad/s por lo que son equivalentes a las anteriores multiplicadas por 2π.

Para observar valores válidos de margen de fase y margen de ganancia hemos modificado los parámetros del lazo usado anteriormente bajando su ancho de banda y frecuencia de muestreo.

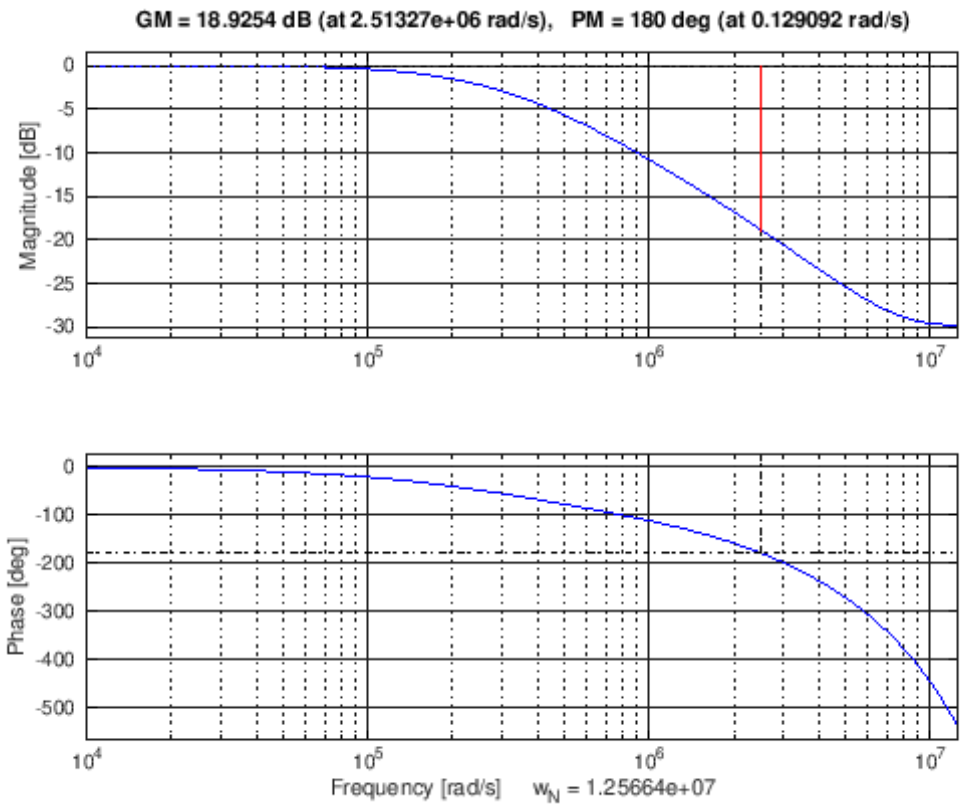
[58]:
```
Bl=.04e6;              % Ancho de banda, en Hz
fs=4e6;               % Frecuencia de muestreo, in Hz
D=2;                   % Latencia en ciclos de reloj de muestreo
Kp=Bl/fs*2*pi;    % Ganancia proporcional Normalizada para el ancho de banda y␣
↪la frecuencia de muestreo
Ki=Kp*1e-2*1.0;   % Ganancia integral a Modificar para analizar modificaciones␣
↪en la respuesta
```

```
[H1,H2,L1,L2,G1,G2]=generar_ft(Bl,fs,D,Kp,Ki);

pzmap(H1);
```



[59]:
```
[gamma, phi, w_gamma, w_phi] = margin (H1)

margin(H1)
```

```
gamma =  8.8363
phi =  180.00
w_gamma =    2.5133e+06
w_phi =  0.12909
```



4.7 F. Simulación de salida PLL para diferentes entradas usando LSIM

En ésta sección se ha modificado el código provisto por el docente para encontrar las salidas del sistema. Se ha usado la función **lsim** de MATLAB/OCTAVE para aplicar cierta entrada a la función de transferencia de tiempo discreto.

4.7.1 Entrada tono cosenoidal sin ruido

[60]:
```
%Genero función de tranferencia

Bl=.40e6;               % Ancho de banda, en Hz
fs=400e6;               % Frecuencia de muestreo, in Hz
D=0;                    % Latencia en ciclos de reloj de muestreo
Kp=Bl/fs*2*pi;    % Ganancia proporcional Normalizada para el ancho de banda y␣
 ↪la frecuencia de muestreo
Ki=Kp*1e-4*1.0;    % Ganancia integral a Modificar para analizar modificaciones␣
 ↪en la respuesta
```

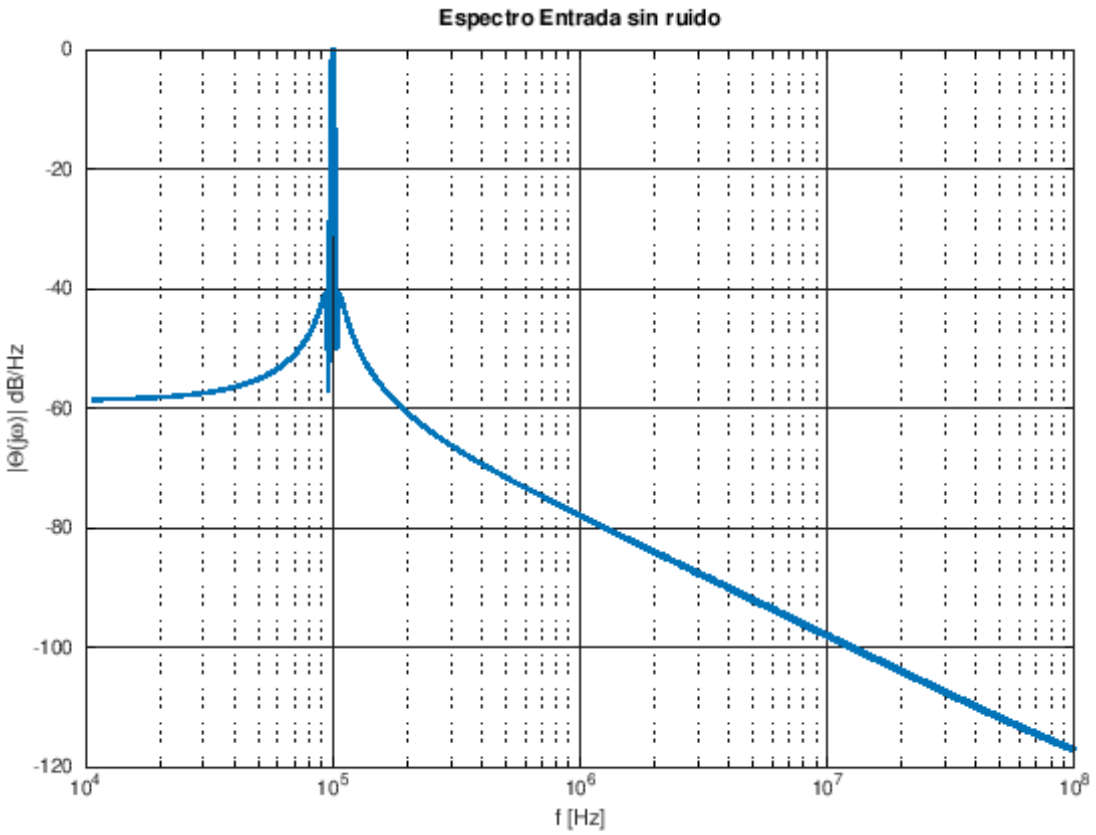
```
[H1,H2,L1,L2,G1,G2]=generar_ft(Bl,fs,D,Kp,Ki);
```

[61]:
```
H1
```

```
Transfer function 'H1' from input 'u1' to output …

       0.006283
 y1:  ----------
      z - 0.9937

Sampling time: 2.5e-09 s
Discrete-time model.
```

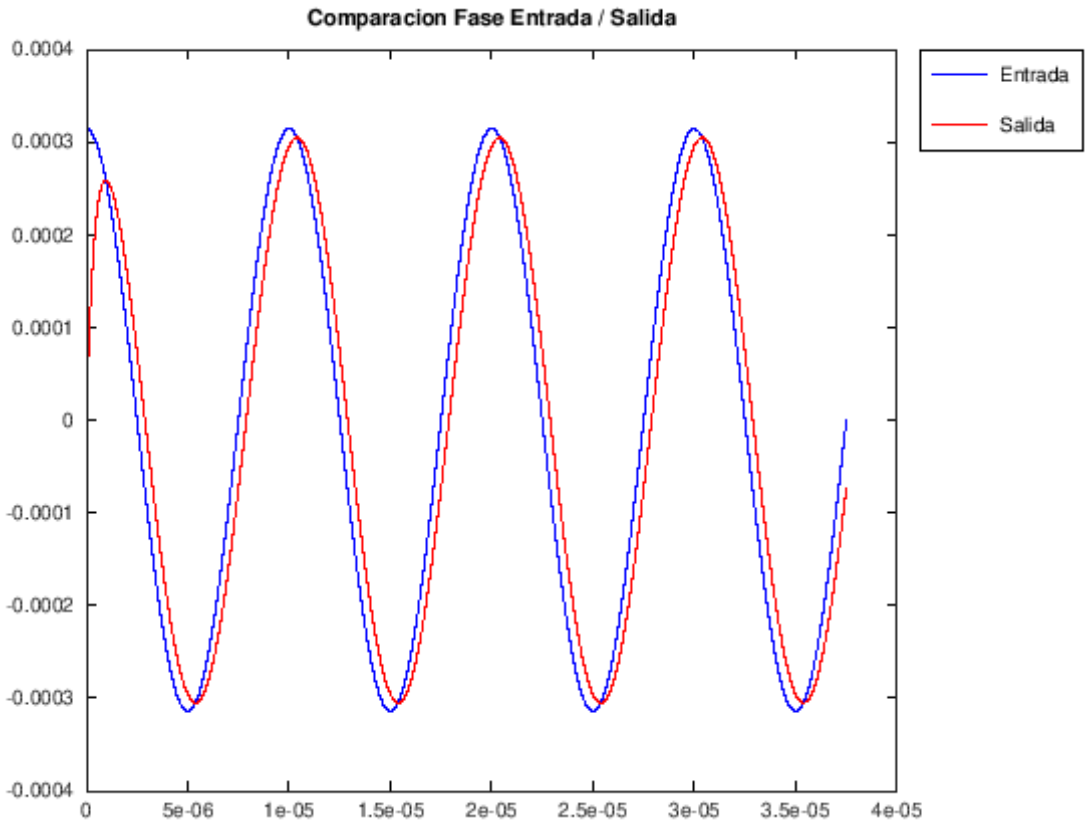
[62]:
```
N=1e6; % Cantidad de iteraciones para respuesta temporal
n=1:(N);  % Indice discreto para respuesta temporal
t=0:(1/fs):((N-1)/fs); % Indice de tiempo muestreado para respuesta temporal
f0=1e5;               %Frecuencia de Tono de Jitter
A=.00005;  % Amplitud de Tono de Jitter

tita_n_jitter=2*pi*A*cos(2*pi*f0*t);
```

[63]:
```
[H,F]=pwelch(tita_n_jitter, hamming(2^18), [], 2^18, fs);
m=find((F>1e4)&(F<1e8));
h=semilogx(F(m),10*log10(abs(H(m))/max(H)));
set(h,'Linewidth',6);
set(h,'Markersize',16);
title('Espectro Entrada sin ruido');
xlabel('f [Hz]');
ylabel('|\Theta(j\omega)| dB/Hz');
grid on
```

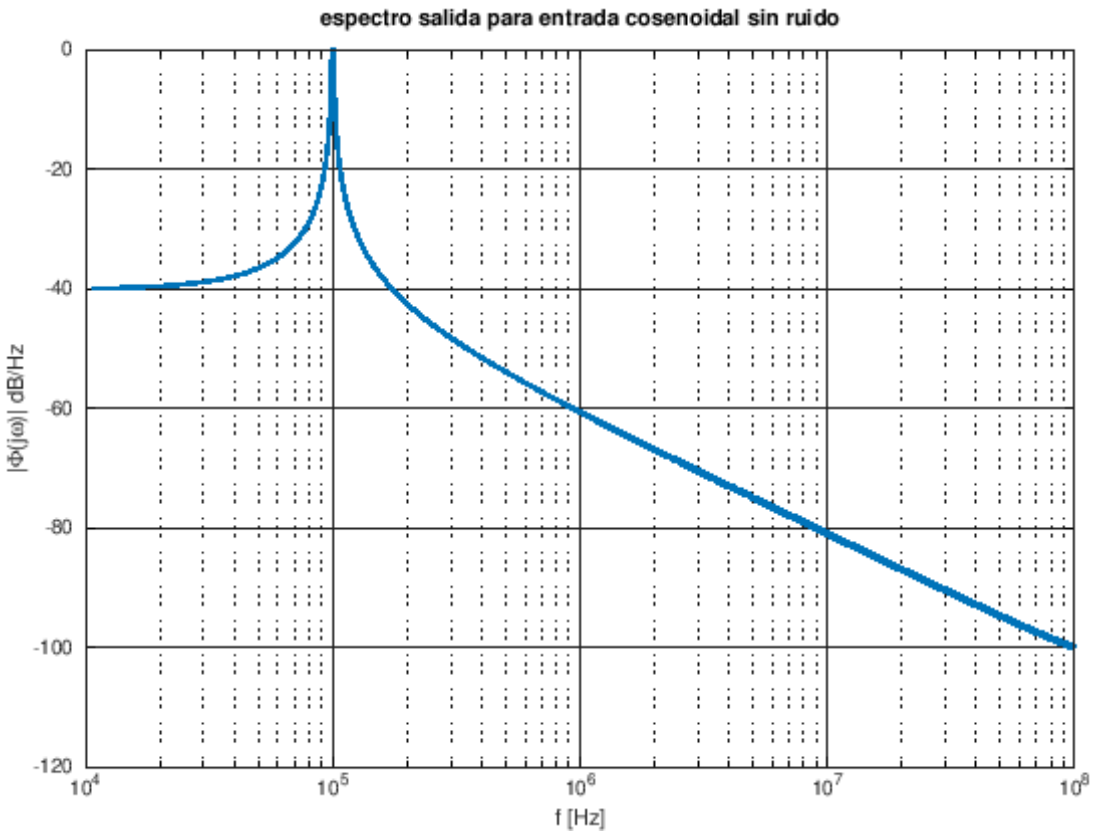


[64]:
```
phi_n=lsim(H1,tita_n_jitter); % Aquí aplicamos la entrada de jitter al sistema␣
↪H1
```

[65]:
```
plot(t(1:15000),tita_n_jitter(1:15000),'b',t(1:15000),phi_n(1:15000),'r');
title('Comparacion Fase Entrada / Salida')

h=legend('Entrada','Salida');
legend (h, "location", "northeastoutside");
```



[66]:
```
[H,F]=pwelch(phi_n, rectwin(2^18), [], 2^18, fs);
m=find((F>1e4)&(F<1e8));
h=semilogx(F(m),10*log10(H(m)/max(H)));
set(h,'Linewidth',6);
set(h,'Markersize',16);
title('espectro salida para entrada cosenoidal sin ruido')
xlabel('f [Hz]');
ylabel('|\Phi(j\omega)| dB/Hz');
grid on
```



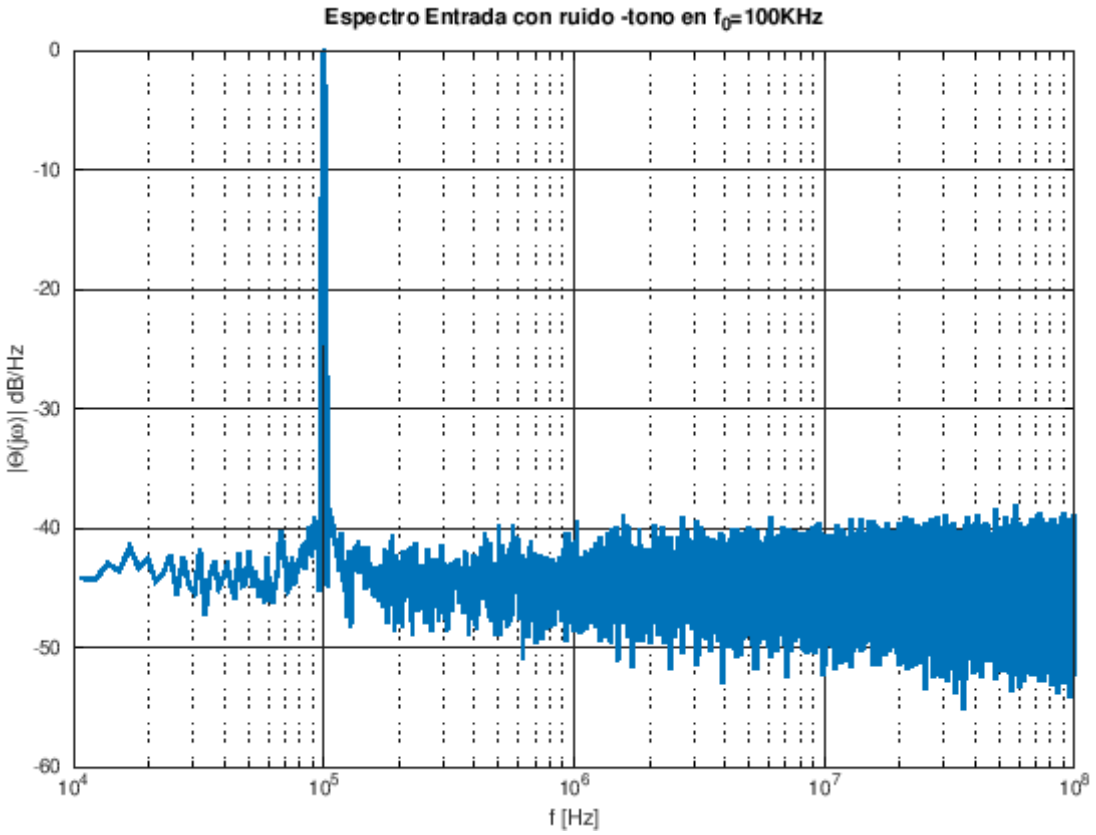
4.7.2 Simulación para entrada de tono cosenoidal + ruido

Aquí utilizando el RNG interno de Octave se genera un ruido que se acopla a la señal para un determinado nivel de señal-ruido.

Veremos que el **ruido a la entrada es blanco** y a la salida el ruido se ve coloreado por el perfil de la función de transferencia del lazo.

[67]:
```
N=1e6; % Cantidad de iteraciones para respuesta temporal
n=1:(N);  % Indice discreto para respuesta temporal
t=0:(1/fs):((N-1)/fs); % Indice de tiempo muestreado para respuesta temporal
f0=1e5;              %Frecuencia de Tono de Jitter
A=.0001;  % Amplitud de Tono de Jitter
SNR_TRdB=60; % Relación señal ruido en dB
SNR_TR=10^(SNR_TRdB/10);

noise=sqrt(.5)*randn(1,N); % ruido a  sumarse a tono cosenoidal de entrada

tita_n_jitter=2*pi*A*cos(2*pi*f0*t)+1.0*noise(n)/sqrt(SNR_TR);
```

[68]:
```
[H,F]=pwelch(tita_n_jitter, hamming(2^18), [], 2^18, fs);
m=find((F>1e4)&(F<1e8));
h=semilogx(F(m),10*log10(abs(H(m))/max(H)));
set(h,'Linewidth',6);
set(h,'Markersize',16);
title('Espectro Entrada con ruido -tono en f_0=100KHz');
xlabel('f [Hz]');
ylabel('|\Theta(j\omega)| dB/Hz');
grid on
```

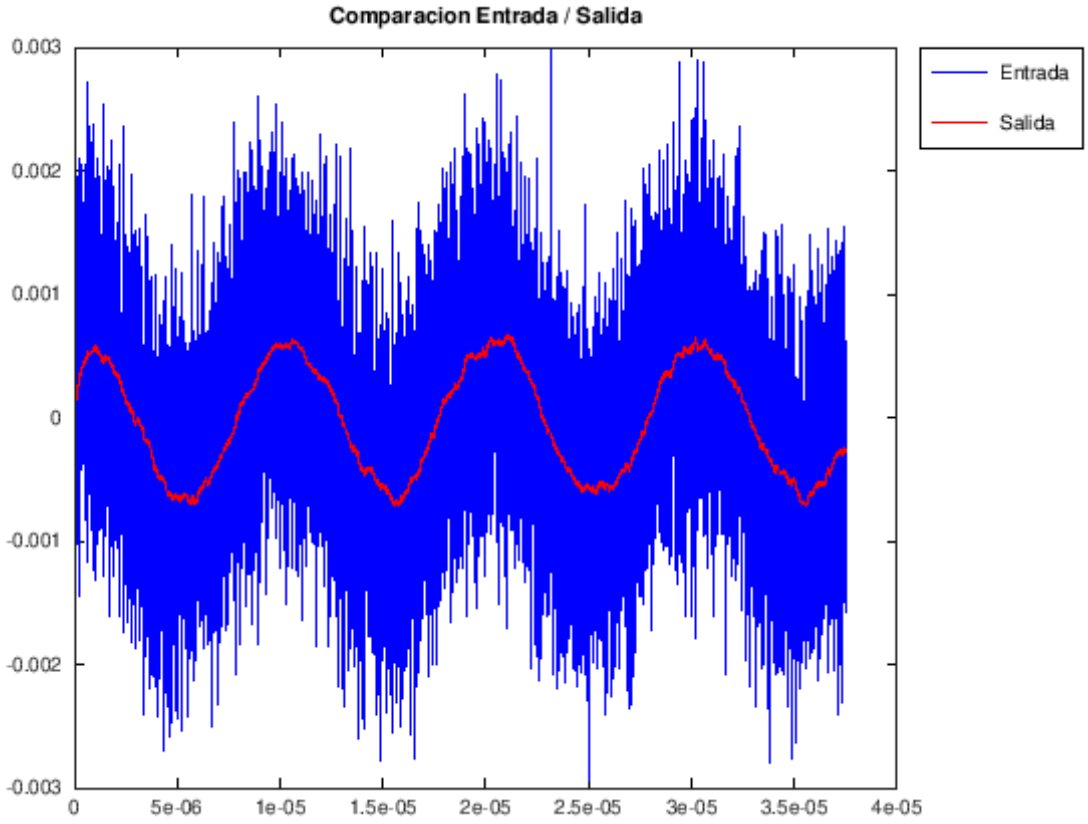


[69]:
```
phi_n=lsim(H2,tita_n_jitter); % Aquí aplicamos la entrada de jitter al sistema␣
↪H2
```

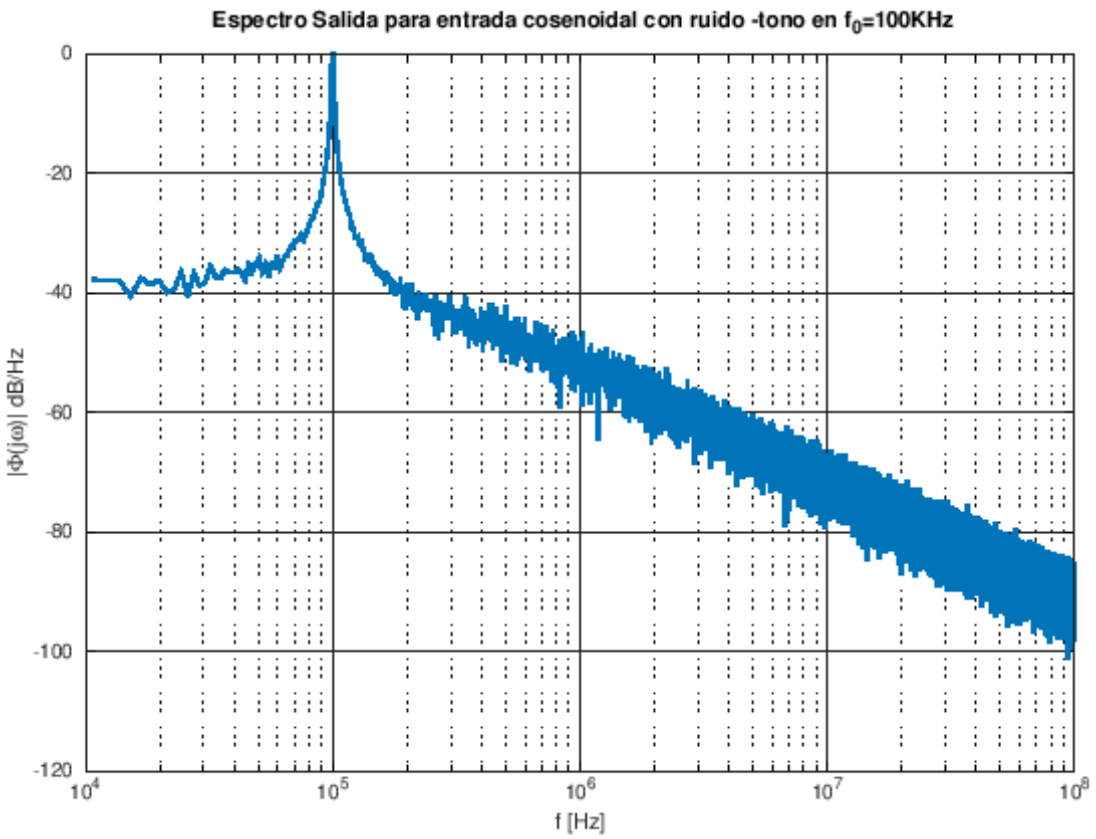
4.7.3 Comparación entre fase de entrada y fase de salida

El efecto de latencia en el sistema puede llegar a afectar,además de la estabilidad del sistema, la diferencia de fase entre entrada y salida. Por lo que puede ser relevante en un caso de diseño comparar las fases a la entrada y a la salida

[70]:
```
plot(t(1:15000),tita_n_jitter(1:15000),'b',t(1:15000),phi_n(1:15000),'r');
title('Comparacion Entrada / Salida')

h=legend('Entrada','Salida');
legend (h, "location", "northeastoutside");
```



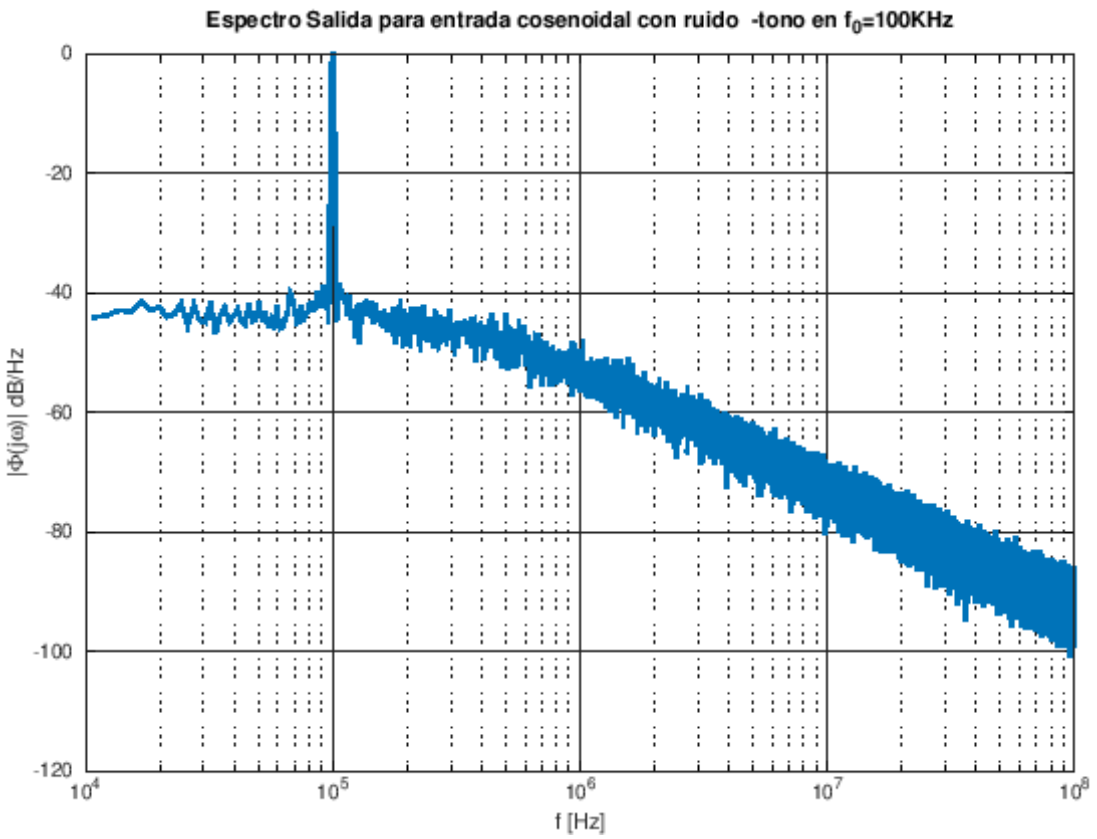
[71]:
```
[H,F]=pwelch(phi_n, rectwin(2^18), [], 2^18, fs);
m=find((F>1e4)&(F<1e8));
h=semilogx(F(m),10*log10(H(m)/max(H)));
set(h,'Linewidth',6);
set(h,'Markersize',16);
title(' Espectro Salida para entrada cosenoidal con ruido -tono en f_0=100KHz')
xlabel('f [Hz]');
ylabel('|\Phi(j\omega)| dB/Hz');

grid on
```



Filtrado de nivel de continua en la señal de salida Aquí hacemos lo mismo que en el apartado anterior pero usamos la versión de la salida que tiene filtrado su nivel de continua.

En realidad la idea de éste apartado es consultar al docente el motivo de hacer ésto en el código provisto.

[72]:
```
dc_phi_n=mean(phi_n); % Encontramos la media es decir el valor de DC de la␣
↪salida

phi_n_no_dc=phi_n-dc_phi_n; % Filtramos el valor de DC de la salida

[H,F]=pwelch(phi_n_no_dc, hamming(2^18), [], 2^18, fs);
m=find((F>1e4)&(F<1e8));
h=semilogx(F(m),10*log10(H(m)/max(H)));
set(h,'Linewidth',6);
set(h,'Markersize',16);
title('Espectro Salida para entrada cosenoidal con ruido  -tono en f_0=100KHz')
xlabel('f [Hz]');
ylabel('|\Phi(j\omega)| dB/Hz');
grid on
```

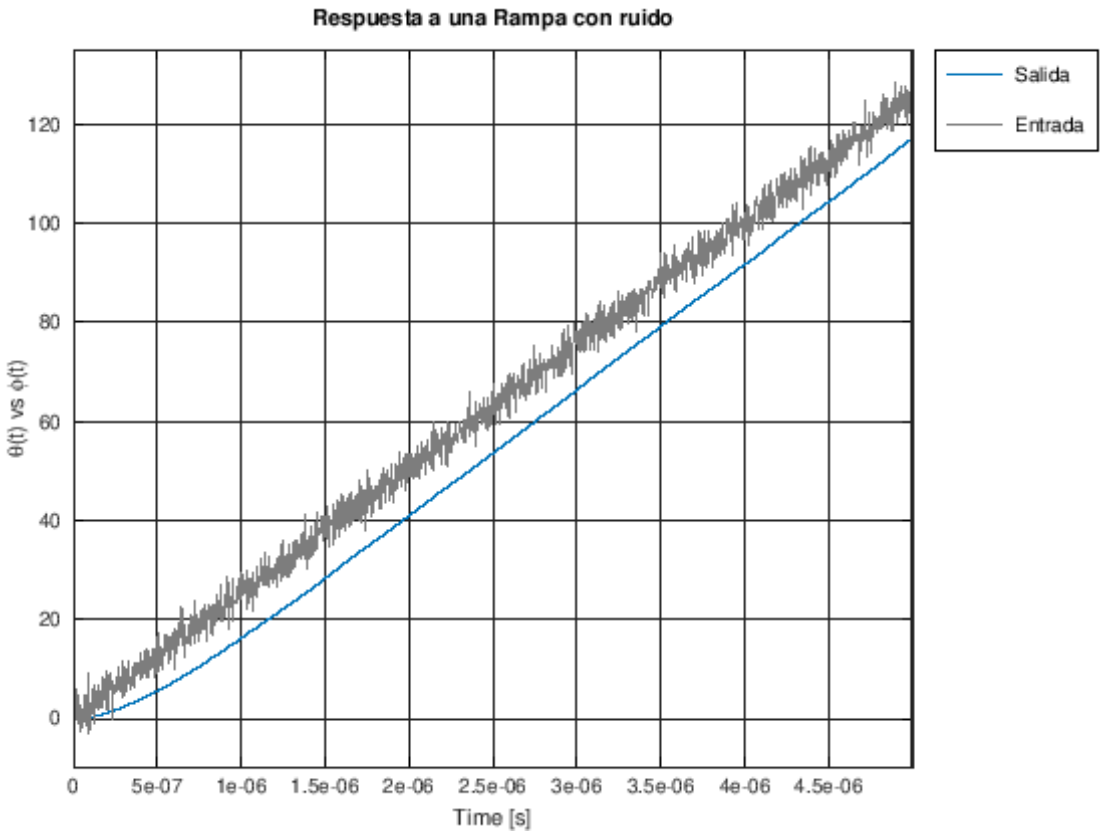


4.7.4 Simulación para entrada rampa

Ahora hacemos el caso estudiado en el apartado teórico con una entrada rampa para cierto offset de frecuencia.

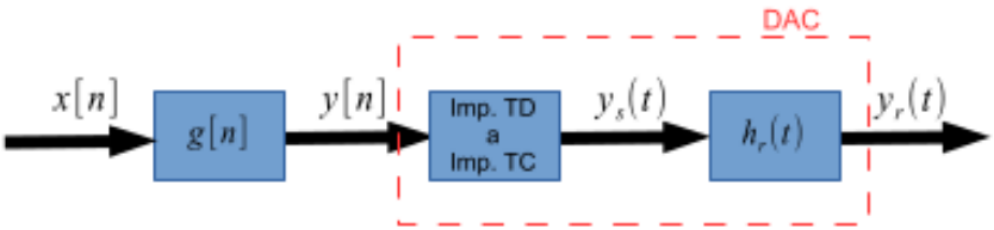
[73]:
```
N=1e6; % Cantidad de iteraciones para respuesta temporal
n=1:(N);  % Indice discreto para respuesta temporal
t=0:(1/fs):((N-1)/fs); % Indice de tiempo muestreado para respuesta temporal
Omega_0=2*pi*.01; %Offset de frecuencia (para entrada rampa)

tita_n_rampa=Omega_0*n;
```

[74]:
```
lsim(H2,tita_n_rampa(1:2e3)+3*noise(1:2e3));
title('Respuesta a una Rampa con ruido');
h=legend('Salida', 'Entrada');
legend (h, "location", "northeastoutside");
ylabel('\theta(t) vs \phi(t)');
```



5 Laboratorio N° 4

Se nos pide considerar modem elemental como el de la siguiente figura:



Siendo:

$$g[n] = g(nT_s)$$

$$g(t) = \text{sinc}(t/T)\frac{\cos(\pi\beta t/T)}{1-4\beta^2t^2/T^2}$$

Siendo T_s el período de muestreo y T el tiempo entre-símbolos, es decir siendo $1/T$ el baud-rate.

$h_r(t)$ es la respuesta al impulso del filtro reconstructor.

El laboratorio consiste en :

A. Mostrar que:

$$y_r(t) = \sum y[n]h_r\left(t - nT_s\right) \rightarrow Y_r(j\omega) = H_r(j\omega)Y\left(e^{j\omega T_s}\right)$$

Y que siendo $Y\left(e^{j\omega T_s}\right) = G\left(e^{j\omega T_s}\right)X\left(e^{j\omega T_s}\right)$

$$\begin{aligned} Y_r(j\omega) &= H_r(j\omega)Y\left(e^{j\omega T_s}\right) \\ &= \underbrace{H_r(j\omega)G\left(e^{j\omega T_s}\right)}_{H_t(j\omega)}X\left(e^{j\omega T_s}\right) \end{aligned}$$

B. Se nos pide también graficar $|H_t(j\omega)|^2$ para diferentes filtros reconstructores, es decir:

Siendo:

$$H_t(j\omega) = H_r(j\omega)G(e^{j\omega T_s})$$

Graficar usando $M = T/T_s$ para los siguientes filtros reconstructores:

$$\begin{aligned} &h_r(t) = \text{sinc}\left(t/T_s\right)\frac{\cos\left(\pi\beta t/T_s\right)}{1-4\beta^2t^2/T_s^2}, \quad \beta = 0, 10 \\ &h_r(t) = \text{rect}\left(t, T_s\right) \quad \text{(pulso rectangular de ancho } T_s\text{)} \\ &h_r(t) = \text{rect}\left(t, T_s\right) * f_{\omega_c,P}(t) \quad (f_{\omega_c,P}(t) : \text{ filtro pasabajo Butterworth de orden } P \end{aligned}$$

5.1 A. Deducción de expresión filtro equivalente tiempo continuo

Siguiendo el esquema anterior vemos que en tiempo discreto tenemos que:

$$y[n] = x[n] * g[n]$$

Y los impulsos en tiempo-discreto que conforman a y[n], en tiempo-continuo se pueden expresar de la siguiente manera:

$$y_s(t) = \sum_{n=-\infty}^{\infty} y[n]\delta\left(t - nT_s\right)$$

Donde T_s es el período de muestreo. La expresión anterior ya es de tiempo contínuo y nos permite continuar hacia delante en el esquema y convolucionar $y_s(t)$ con $h_r(t)$:

$$y_r(t) = y_s(t) * h_r(t)$$

$$y_r(t) = \sum_{n=-\infty}^{\infty} y[n]\delta\left(t - nT_s\right) * h_r(t)$$

$y[n]$ es independiente de t por lo que en la expresión anterior observamos tenemos una convolución por un impulso desplazado cuyo resultado es la señal desplazada:

$$\delta\left(t - nT_s\right) * h_r(t) = h_r(t - nT_s)$$

Quedándonos:

$$y_r(t) = \sum_{n=-\infty}^{\infty} y[n]h_r(t - nT_s)$$

Si hacemos la transformada de Fourier en tiempo-continuo de los elementos de la expresión anterior tenemos:

Para la señal reconstruída:

$$\mathscr{F}\left\{y_r(t)\right\} = Y_r(j\omega)$$

Para el filtro reconstructor:

$$\mathscr{F}\left\{h_r(t)\right\} = H_r(j\omega)$$

Pero el filtro reconstructor está desplazado, por lo que aplicamos la propiedad de desplazamiento en el tiempo de la transformada de Fourier quedando :

$$\mathscr{F}\left\{h_r(t - nT_s)\right\} = H_r(j\omega)e^{-j\omega nT_s}$$

Agrupando las expresiones de la transformada tenemos que la transformada de una combinación lineal es una combinación lineal de las transformada individuales:

$$Y_r(j\omega) = \sum_{n=-\infty}^{\infty} y[n]H_r(j\omega)e^{-j\omega nT_s}$$

$H_r(j\omega)$ es independiente de n por lo que lo extraemos de la sumatoria quedando:

$$Y_r(j\omega) = H_r(j\omega) \sum_{n=-\infty}^{\infty} y[n]e^{-j\omega nT_s}$$

La expresión anterior equivale a multiplicar la transformada de Fourier en tiempo-continuo del filtro reconstructor, por la transformada de Fourier en tiempo-discreto de $y[n]$ con su eje de frecuencia comprimido en T_s :

$$Y\left(e^{j\omega T_s}\right) = \sum_{n=-\infty}^{\infty} y[n]e^{-j\omega T_s n}$$

$$Y_r(j\omega) = H_r(j\omega)Y\left(e^{j\Omega_c}\right)$$

Siendo $\Omega_c = \omega T_S$:

$$Y_r(j\omega) = H_r(j\omega)Y\left(e^{j\omega T_s}\right)$$

Teniendo en cuenta de que:

$$Y\left(e^{j\omega T_s}\right) = X\left(e^{j\omega T_s}\right)G\left(e^{j\omega T_s}\right)$$

Nos queda la expresión completa:

$$Y_r(j\omega) = H_r(j\omega)X\left(e^{j\omega T_s}\right)G\left(e^{j\omega T_s}\right)$$

$$Y_r(j\omega) = H_t(j\omega)X\left(e^{j\omega T_s}\right)$$

De donde llamamos filtro equivalente total en tiempo-continuo $H_t(j\omega)$ a :

$$H_t(j\omega) = H_r(j\omega)G\left(e^{j\omega T_s}\right)$$

Ésta última expresión es la que iremos ploteando para diferentes $H_r(j\omega)$ en el siguiente apartado del práctico, siempre usando $G\left(e^{j\omega T_s}\right)$ fijo , siendo la respuesta en frecuencia de un coseno realzado visto en los laboratorios anteriores.

5.2 B. Plot de filtros equivalentes totales para diferentes casos

En éste apartado definimos los parámetros fijos a usar y creamos la función r_cosine para octave, para usarla a lo largo del laboratorio actual y el siguiente.

5.2.1 Definición de parámetros iniciales

[75]:
```
% Definimos los parámetros iniciales fijos

baud_rate = 1e6; % Usamos 1M símbolo/segundo
T=1/baud_rate ; % Tiempo entre símbolos
M=16; % Factor de sobremuestreo M=T/T_s Siendo T_s<T
sampling_rate=M*baud_rate; % Frecuencia de muestreo
T_s= 1/sampling_rate; % Período de muestreo
```

5.2.2 Función r_cosine para Octave

[76]:
```
% genero función que genera respuesta de un coseno realzado y su vector
% de tiempo sobremuestrado asociado

function [cos_r,t] = r_cosine(beta,T_p,L,M)
% beta es factor de roll-off
% T_p tiempo entre picos
% M factor de sobremuestreo
% L longitud de la respuesta

t = [-L:1/M:L]*T_p; % vector de tiempo sobremuestreado
cos_r = sinc(t/T_p).*cos(pi*beta*t/T_p)./(1-4*beta^2*t.^2/T_p^2);
stem(t,cos_r);
str=sprintf('Coseno realzado - Factor de roll-off :   %s ',num2str(beta));
title(str);
xlabel('Tiempo (seg)');
ylabel('Amplitud');
endfunction
```

5.2.3 Modificación a la FFT en Octave

Aquí generamos una función que realiza la transformada rápida de Fourier y devuelve el vector de frecuencias normalizado para la frecuencia máxima indicada y el módulo de la magnitud de la transformada en dB.

[77]:
```
function [Y,freq,X] = fast_ft(y,fmax,dbplot)
%fmax=1/T tiene que ser la frecuencia maxima de la señal
%dbplot = 1 plotea la magnitud en dB
%transforma usando fft pero entrega el modulo de la transformada y el vector de␣
 ↪frecuencias .

T=1/fmax;
N=length(y);
freq=(-0.5*(N-1):0.5*(N-1))'*4/(N*T) ;
X=fftshift(fft(y,N));
Y=abs(X);
if (dbplot==1)
    plot(freq,10*log10(Y));
    ylabel('Magnitud (dB)');
else
    plot(freq,Y);
    ylabel('Magnitud ');
endif
```

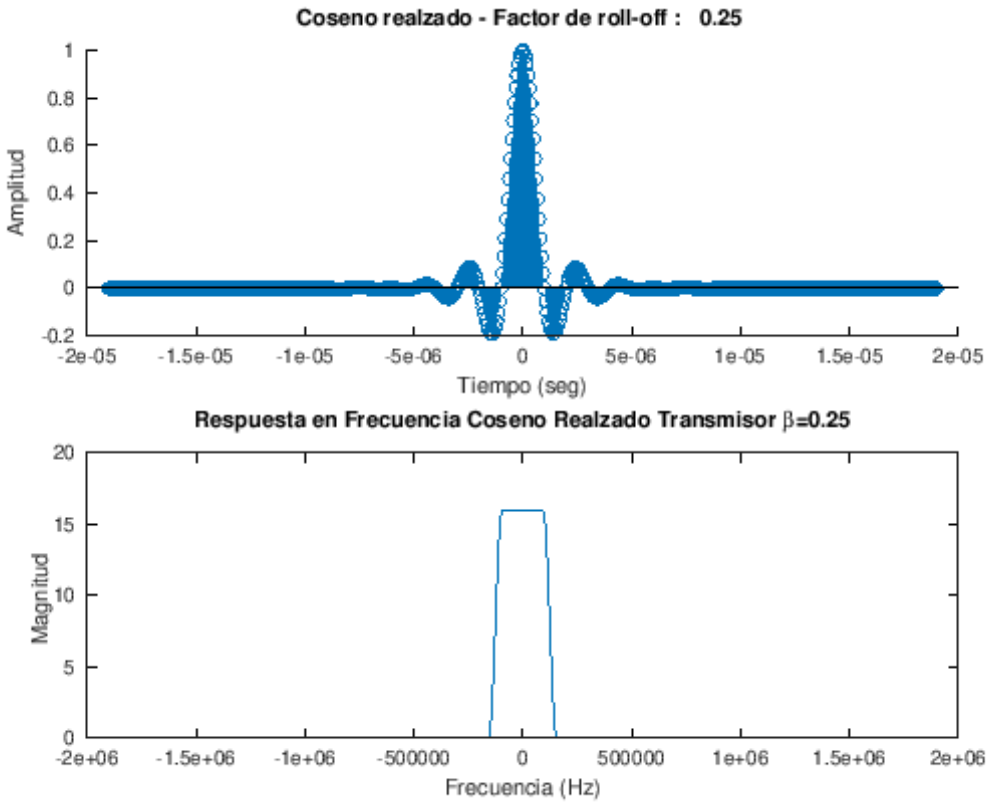
```
xlabel('Frecuencia (Hz)');
endfunction
```

5.2.4 Definición de filtro transmisor

$$g[n] = g(nT_s)$$

$$g(t) = \text{sinc}(t/T)\frac{\cos(\pi\beta t/T)}{1-4\beta^2t^2/T^2}$$

[78]:
```
beta_g=0.25+0.000001 ; %Factor de roll-off del filtro transmisor g[n]
                  %se agrega 0.xx0001 para q no de error el solver
L = 19;  % 2*L*M+1 es el largo del filtro sobremuestreado

% T usaremos la definida inicialmente como tiempo entre símbolos
% M Usaremos la definida inicialmente
subplot(2,1,1);
[gn,t] = r_cosine(beta_g,T,L,M); % Obtengo el filtro transmisor para los␣
 ↪parámetros dados
subplot(2,1,2);
[G_n,freq] = fast_ft(gn,1/T,0) ;
title('Respuesta en Frecuencia Coseno Realzado Transmisor \beta=0.25')
```



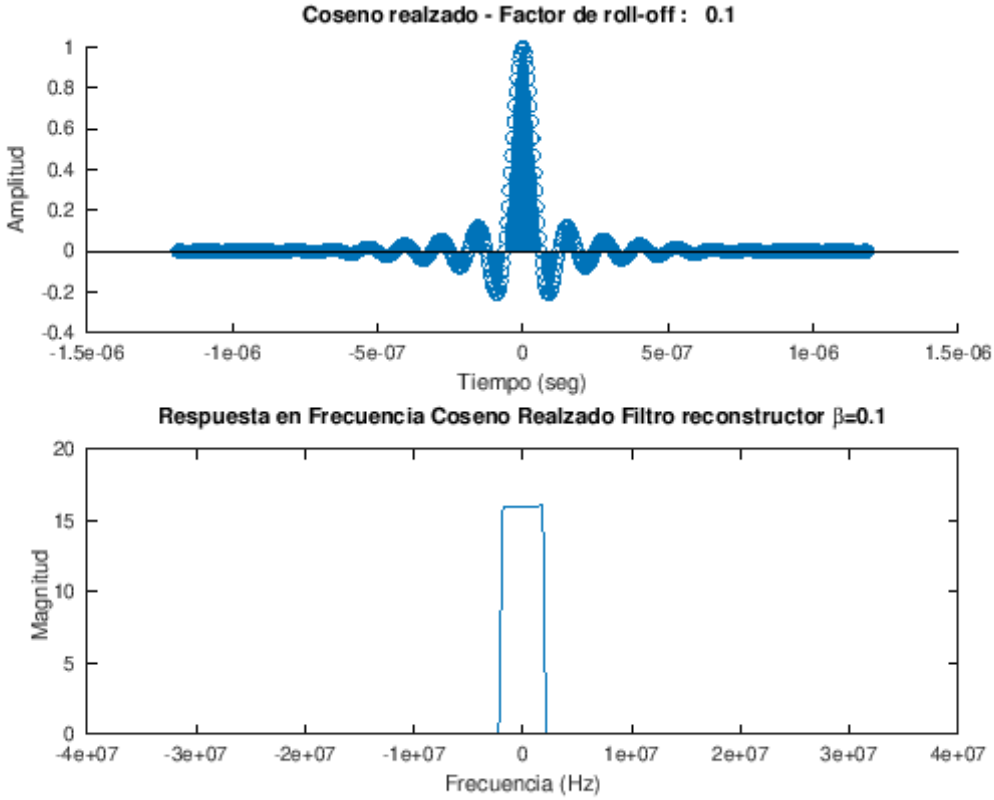
5.2.5 Filtro reconstructor coseno realzado

En éste caso nos piden que el filtro reconstructor sea:

$$h_r(t) = \text{sinc}(t/T_s)\frac{\cos(\pi\beta t/T_s)}{1-4\beta^2t^2/T_s^2}$$

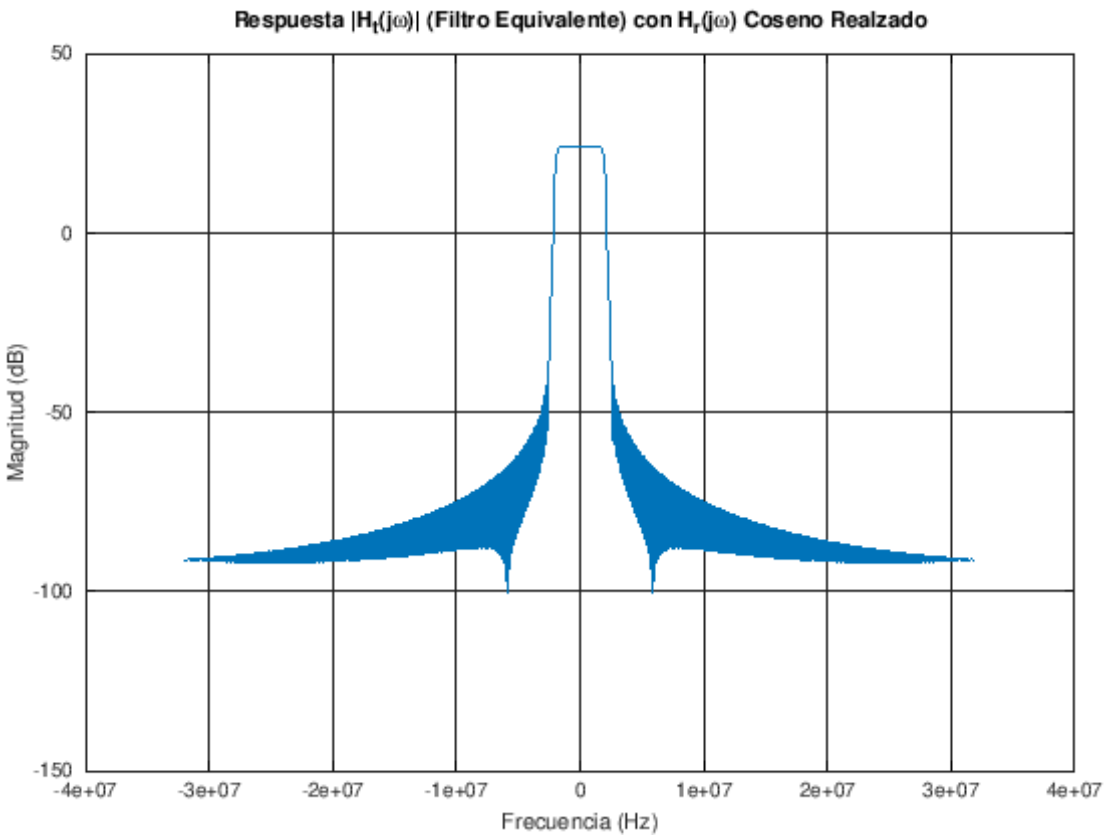
[79]:
```
beta_h_r=0.1+0.000001 ; %Factor de roll-off del filtro transmisor g[n]
                %se agrega 0.xx0001 para q no de error el solver


% T_s es el período de muestreo
% M y L Usaremos los definidos nicialmente

subplot(2,1,1);
[h_r_rcos,t_rcos] = r_cosine(beta_h_r,T_s,L,M); % Obtengo el filtro transmisor␣
 ↪para los parámetros dados
subplot(2,1,2);
[H_r_rcos,freq_rcos] = fast_ft(h_r_rcos,1/T_s,0) ;
```

```
title('Respuesta en Frecuencia Coseno Realzado Filtro reconstructor \beta=0.1')
```



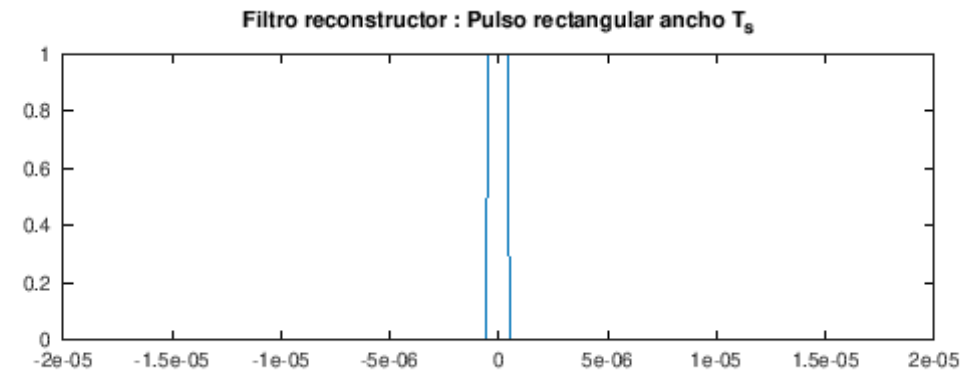
5.2.6 Respuesta equivalente con filtro reconstructor coseno realzado

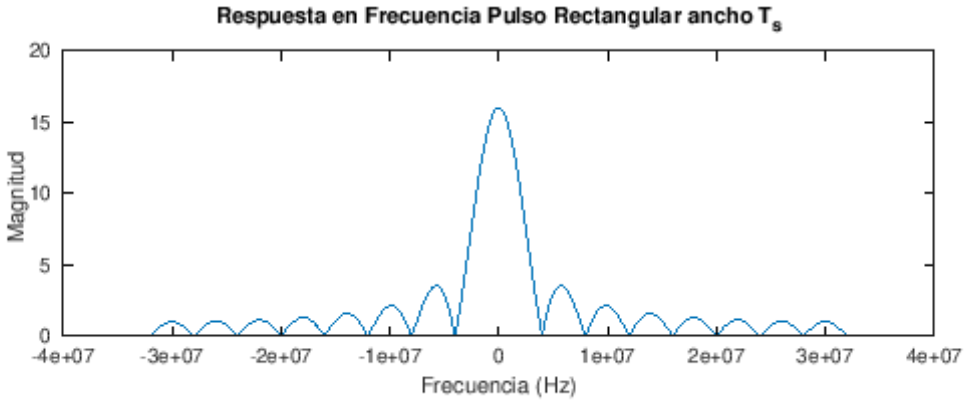
[80]:
```
h_t_rcos=conv(gn,h_r_rcos); % convoluciono ambos filtros
[H_t_rcos,freq_rcos_t] = fast_ft(h_t_rcos,1/T_s,1) ;
title('Respuesta |H_t(j\omega)| (Filtro Equivalente) con H_r(j\omega) Coseno␣
 ↪Realzado')
grid on
```



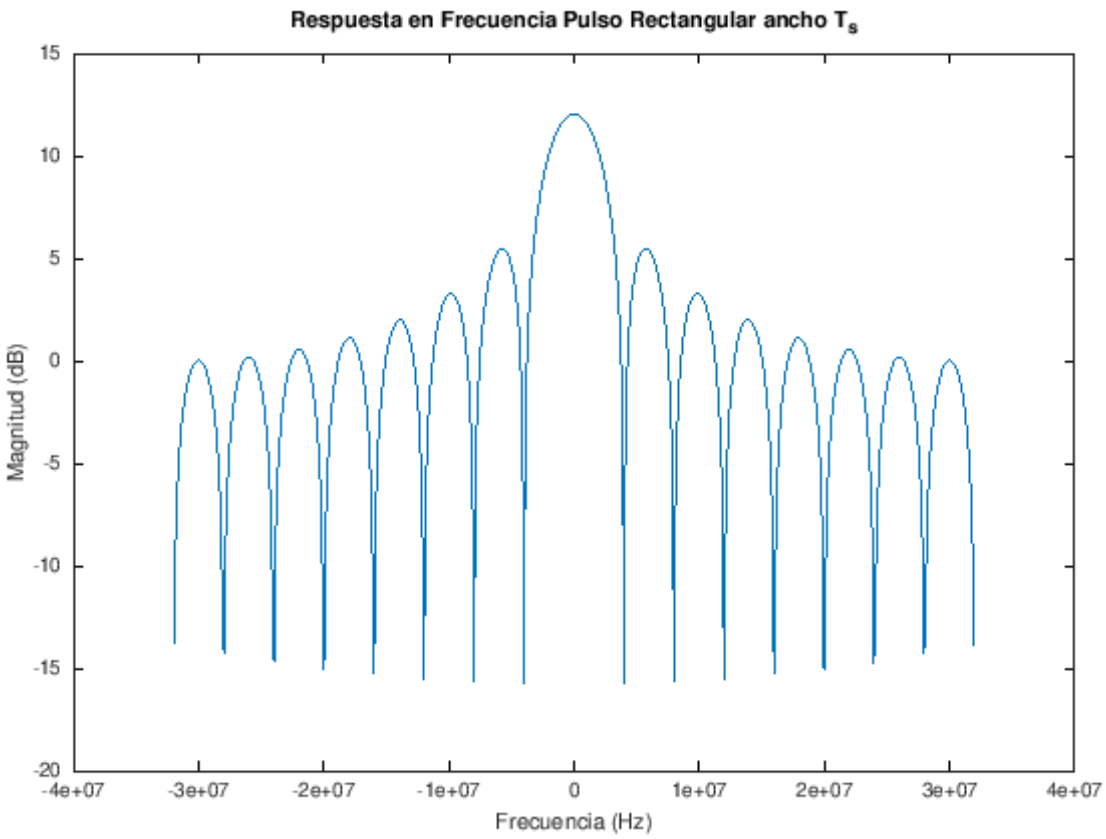
5.2.7 Filtro reconstructor pulso rectangular de ancho T_s

[81]:
```
subplot(2,1,1);
h_r_rectpulse=rectpuls(t_rcos,T_s); % uso el vector de tiempo t_rcos del filtro␣
↪anterior
plot(t,h_r_rectpulse)
title('Filtro reconstructor : Pulso rectangular ancho T_s')
subplot(2,1,2);
[H_r_rectpulse,freq_rectpulse,X] = fast_ft(h_r_rectpulse,1/T_s,0) ;
title('Respuesta en Frecuencia Pulso Rectangular ancho T_s')
```

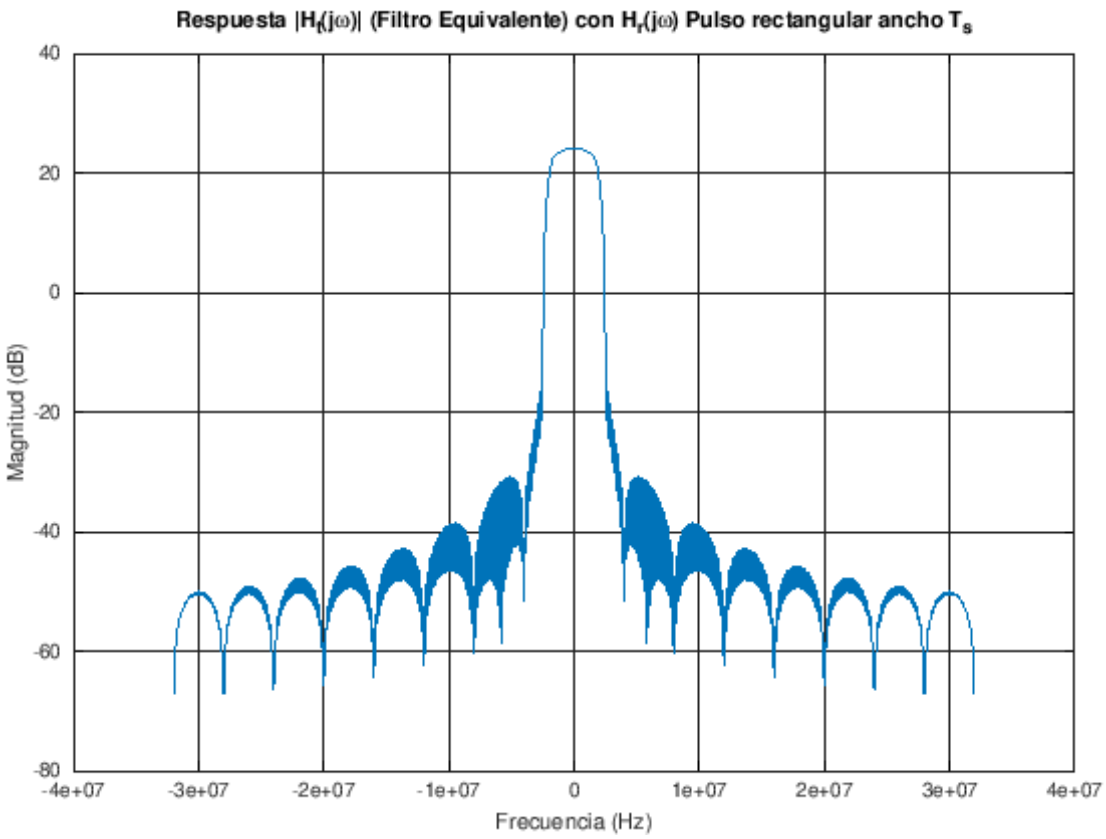




[82]:
```
[H_r_rectpulse,freq_rectpulse,X] = fast_ft(h_r_rectpulse,1/T_s,1) ;
title('Respuesta en Frecuencia Pulso Rectangular ancho T_s')
```



[83]:
```
h_t_rectpulse=conv(gn,h_r_rectpulse); % convoluciono ambos filtros
[H_t_rectpulse,freq_rectpulse] = fast_ft(h_t_rectpulse,1/T_s,1) ;
title('Respuesta |H_t(j\omega)| (Filtro Equivalente) con H_r(j\omega) Pulso␣
 ↪rectangular ancho T_s')
grid on
```

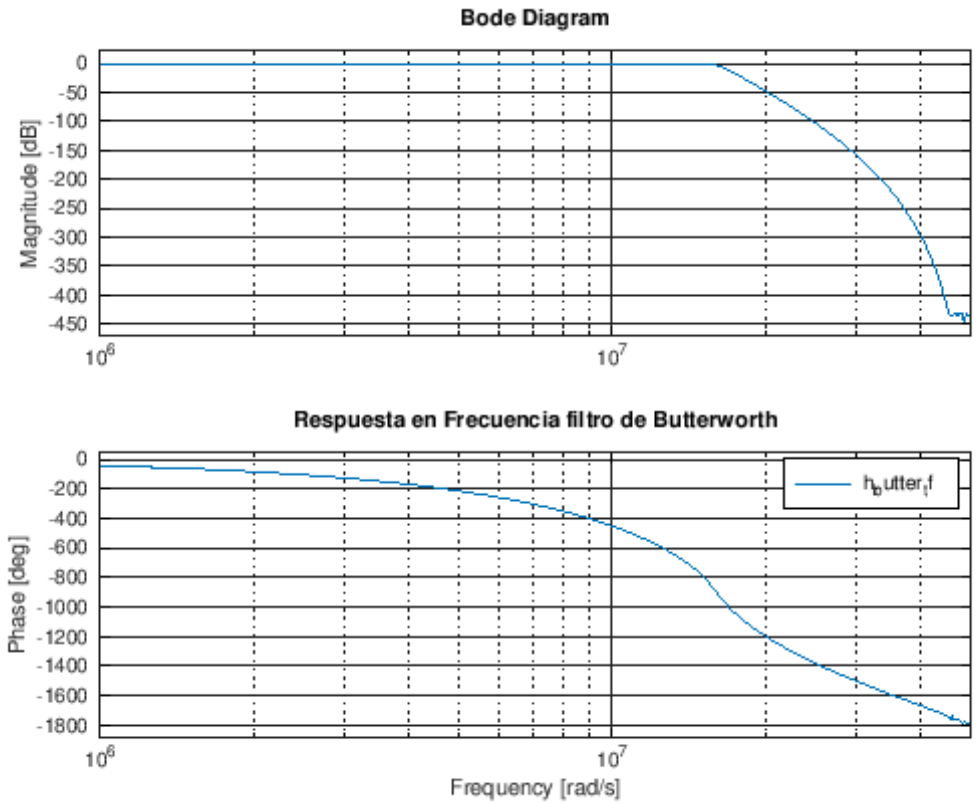


5.2.8 Filtro reconstructor Butterworth

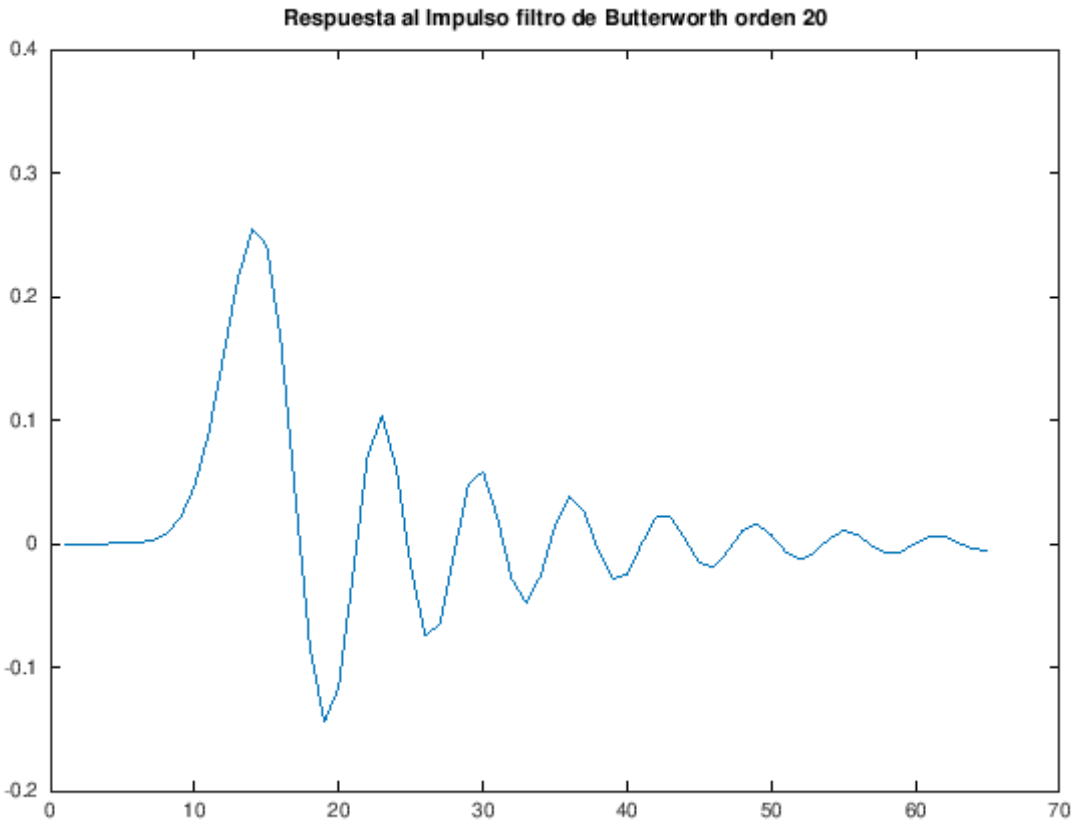
La idea de usar filtros de Butterworth es tener una respuesta en frecuencia lo más plana posible en la banda de paso. Para el caso del filtro reconstructor éste efecto se logra mejor aún convolucionando éste filtro con un filtro rectangular

[84]:
```
[num,den]=butter(20,1/pi);
```

[85]:
```
h_r_butter_tf=tf(num,den,T_s);

bode(h_r_butter_tf);
title('Respuesta en Frecuencia filtro de Butterworth')
```



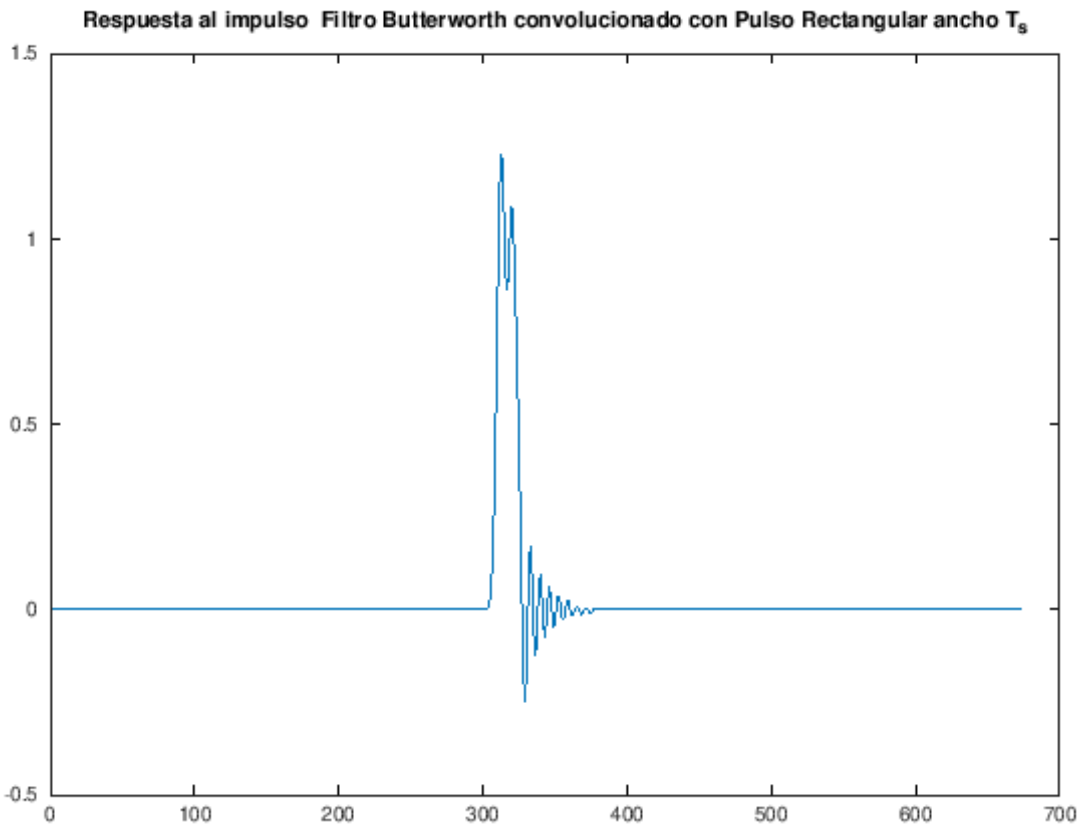
[86]:
```
h_r_butter=impulse(h_r_butter_tf);

plot(h_r_butter);
title('Respuesta al Impulso filtro de Butterworth orden 20')
```



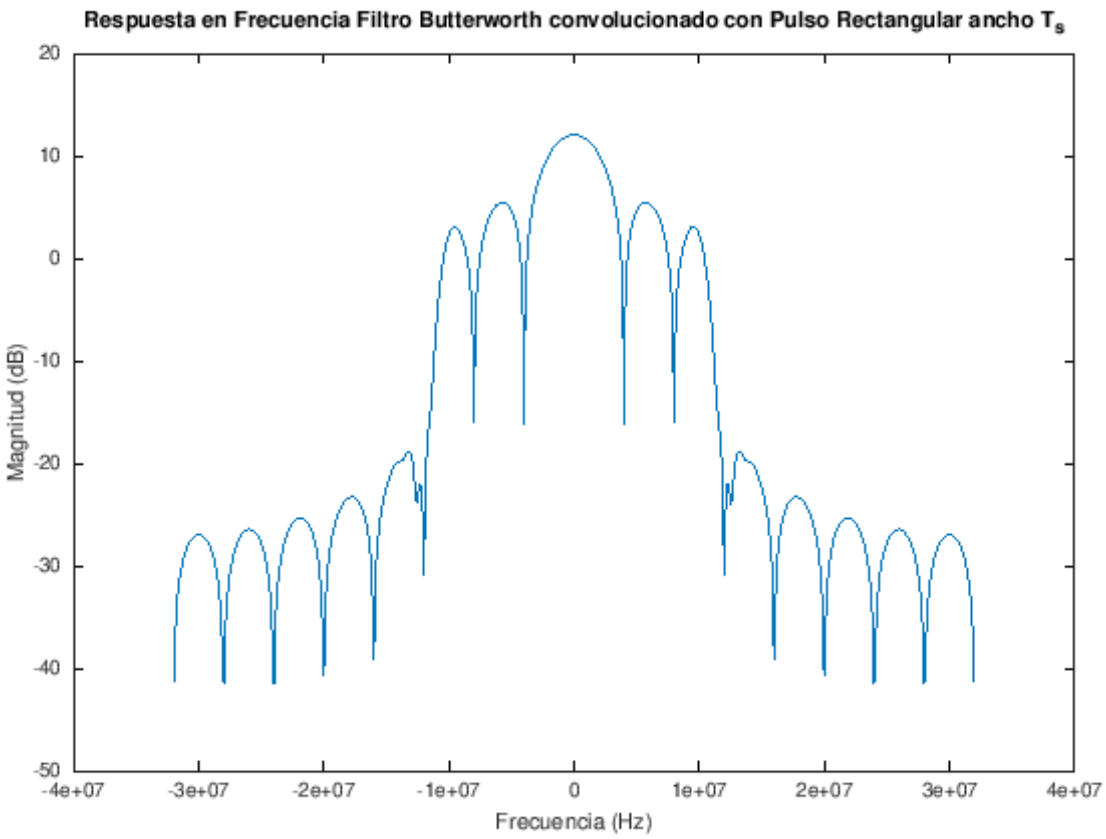
5.2.9 Convolución filtro de Butterworth con pulso rectangular

[87]:
```
h_r_butter_rectpulse=conv(h_r_rectpulse,h_r_butter);
```

[88]:
```
plot(h_r_butter_rectpulse)
title('Respuesta al impulso  Filtro Butterworth convolucionado con Pulso␣
 ↪Rectangular ancho T_s')
```



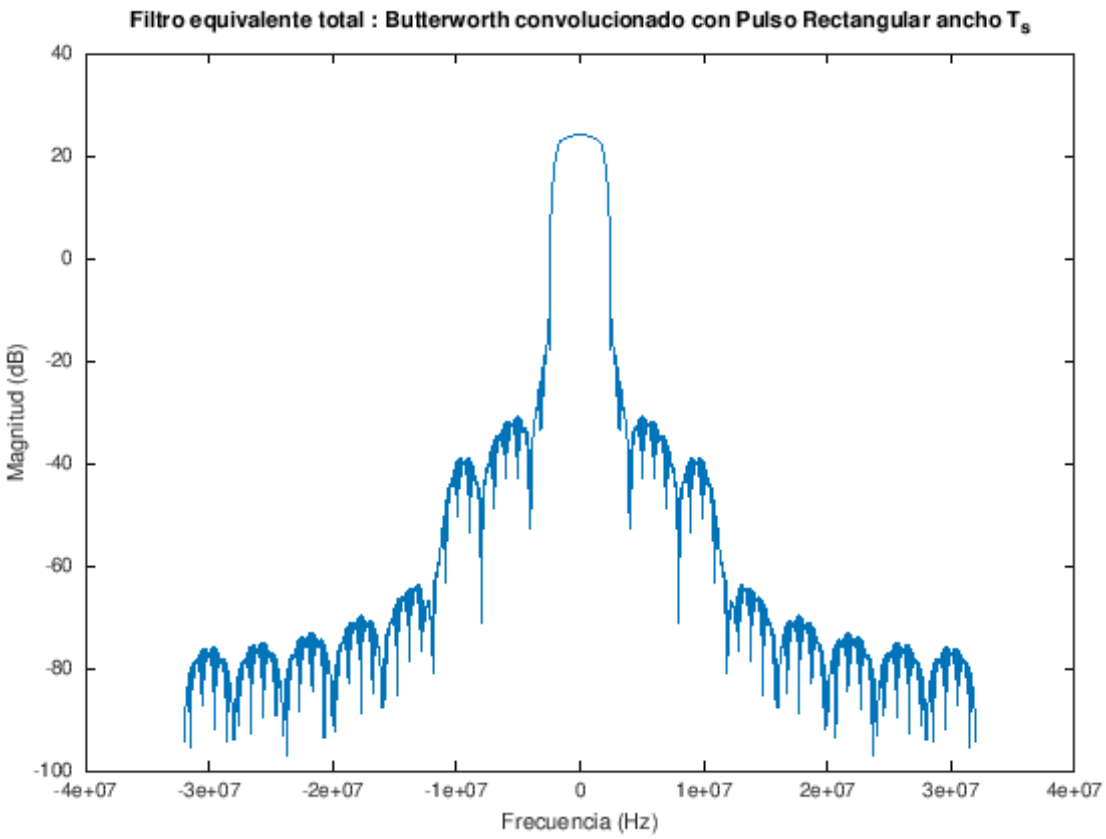
[89]:
```
[H_r_butter_rectpulse,freq_rectpulse] = fast_ft(h_r_butter_rectpulse,1/T_s,1) ;
title('Respuesta en Frecuencia Filtro Butterworth convolucionado con Pulso␣
 ↪Rectangular ancho T_s')
```



5.2.10 Filtro equivalente total con filtro de Butterworth convolucionado con pulso rectangular

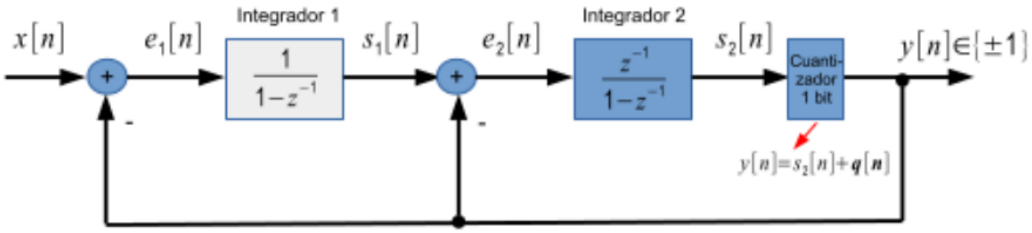
[90]:
```
h_t_butter_rectpulse=conv(gn,h_r_butter_rectpulse);
```

[91]:
```
[H_t_butter_rectpulse,freq_rectpulse] = fast_ft(h_t_butter_rectpulse,1/T_s,1) ;
title('Filtro equivalente total : Butterworth convolucionado con Pulso␣
 ↪Rectangular ancho T_s')
```

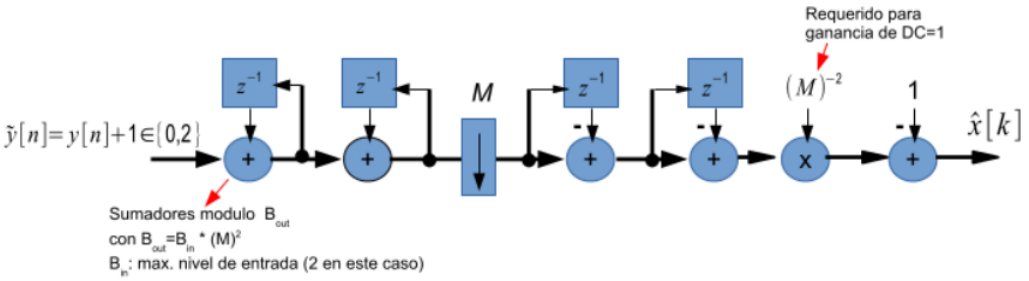


6 Laboratorio N°5

En éste práctico de laboratorio se nos pide considerar un modulador $\Sigma\Delta$ como el de la siguiente figura:



Cuya salida está conectada a un filtro CIC diezmador por un factor M como el de la siguiente figura:



A. Se nos pide mostrar que la transformada Z a la salida del modulador $\Sigma\Delta$ puede expresarse de la siguiente manera:

$$Y(z) = X(z)z^{-1} + Q(z)\left(1 - z^{-1}\right)^2$$

Siendo $Q(z)$ la transformada de la componente ruido del cuantizador $q[n]$.

B. Considerar la siguiente señal de entrada :

$$x[n] = \sum_{k=0}^{N-1} a[k]g[n - kRM], \quad a[k] \in \{\pm 1\}$$

Es decir que los pulsos de información se ven conformados por un filtro de caida consenoidal sobremuestreado por RM, siendo R el factor de sobremuestreo aún a la salida del filtro CIC.

Considerando el ruido de cuantización tenemos que la señal $x[n]$ tiene una componmente $r_q[n]$ quedando:

$$\hat{x}[n] = x[n] + r_q[n]$$

C. Obtener por simulación la densidad espectral de potencia a la salida del modulador $\Sigma\Delta$ y del filtro CIC

D. Generar diagrama de ojo a partir de $\hat{x}[n]$

E. Graficar el diagrama de ojo de la señal $\hat{x}[n]$ filtrada por un filtro de caida cosenoidal(pasa bajos), similar a $g[n]$ pero sobremuestreado por un factor R. Explicar porqué mejora el diagrama ojo y repetir para diferentes valores de M y R.

6.1 A. Deducción expresión salida del modulador $\Sigma\Delta$

Del diagrama en bloques del $\Sigma\Delta$ extraemos las siguientes expresiones transformadas:

$$\begin{gathered} Y(z) = S_2(z) + Q(z) \\ S_1(z) = \frac{E_1(z)}{1-z^{-1}} \\ S_2(z) = \frac{E_2(z)z^{-1}}{1-z^{-1}} \\ E_1(z) = X(z) - Y(z) \\ E_2(z) = S_1(z) - Y(z) \end{gathered}$$

Trabajando sobre las expresiones anteriores , buscamos una expresión que dependa sólo de $Y(z)$, $X(z)$ y $Q(z)$:

Primero obtenemos $E_2(z)$ en función de $X(z)$ e $Y(z)$:

$$\begin{aligned} E_2(z) &= \frac{E_1(z)}{1-z^{-1}} - Y(z) \\ &= \frac{X(z)}{1-z^{-1}} - \frac{Y(z)}{1-z^{-1}} - Y(z) \\ &= \frac{X(z)}{1-z^{-1}} - Y(z)\left(\frac{1}{1-z^{-1}} + 1\right) \\ &= \frac{X(z)}{1-z^{-1}} - Y(z)\frac{2-z^{-1}}{1-z^{-1}} \end{aligned}$$

Luego $S_2(z)$ en función de $X(z)$ e $Y(z)$:

$$S_2(z) = \frac{X(z)z^{-1}}{(1-z^{-1})^2} - Y(z)\frac{\left(2-z^{-1}\right)z^{-1}}{(1-z^{-1})^2}$$

Para luego reemplazar ésta última expresión de $S_2(z)$ en la primera ecuación para operar y obtener la expresión simplificada final sólo en función de $Y(z)$, $X(z)$ y $Q(z)$:

$$\begin{aligned} Y(z) &= \frac{X(z)z^{-1}}{(1-z^{-1})^2} - Y(z)\frac{\left(2-z^{-1}\right)z^{-1}}{(1-z^{-1})^2} + Q(z) \\ Y(z)\left[1 + \frac{\left(2-z^{-1}\right)z^{-1}}{(1-z^{-1})^2}\right] &= \frac{X(z)z^{-1}}{(1-z^{-1})^2} + Q(z) \\ Y(z)\left[\frac{\left(1-z^{-1}\right)^2 + \left(2-z^{-1}\right)z^{-1}}{(1-z^{-1})^2}\right] &= \frac{X(z)z^{-1}}{(1-z^{-1})^2} + Q(z) \\ Y(z)\frac{1}{(1-z^{-1})^2} &= \frac{X(z)z^{-1}}{(1-z^{-1})^2} + Q(z) \\ Y(z) &= X(z)z^{-1} + Q(z)\left(1-z^{-1}\right)^2 \end{aligned}$$

6.2 B. Señal de entrada al sistema

Se nos pide considerar la siguiente señal de entrada :

$$x[n] = \sum_{k=0}^{N-1} a[k]g[n-kRM], \quad a[k] \in \{\pm 1\}$$

Es decir que los pulsos de información se ven conformados por un filtro de caida consenoidal sobremuestreado por RM, siendo R el factor de sobremuestreo aún a la salida del filtro CIC.

Considerando el ruido de cuantización a la salida del filtro CIC tenemos que la señal $x[n]$ tiene una componente $r_q[n]$ quedando:

$$\hat{x}[n] = x[n] + r_q[n]$$

6.2.1 Defínción de parámetros iniciales

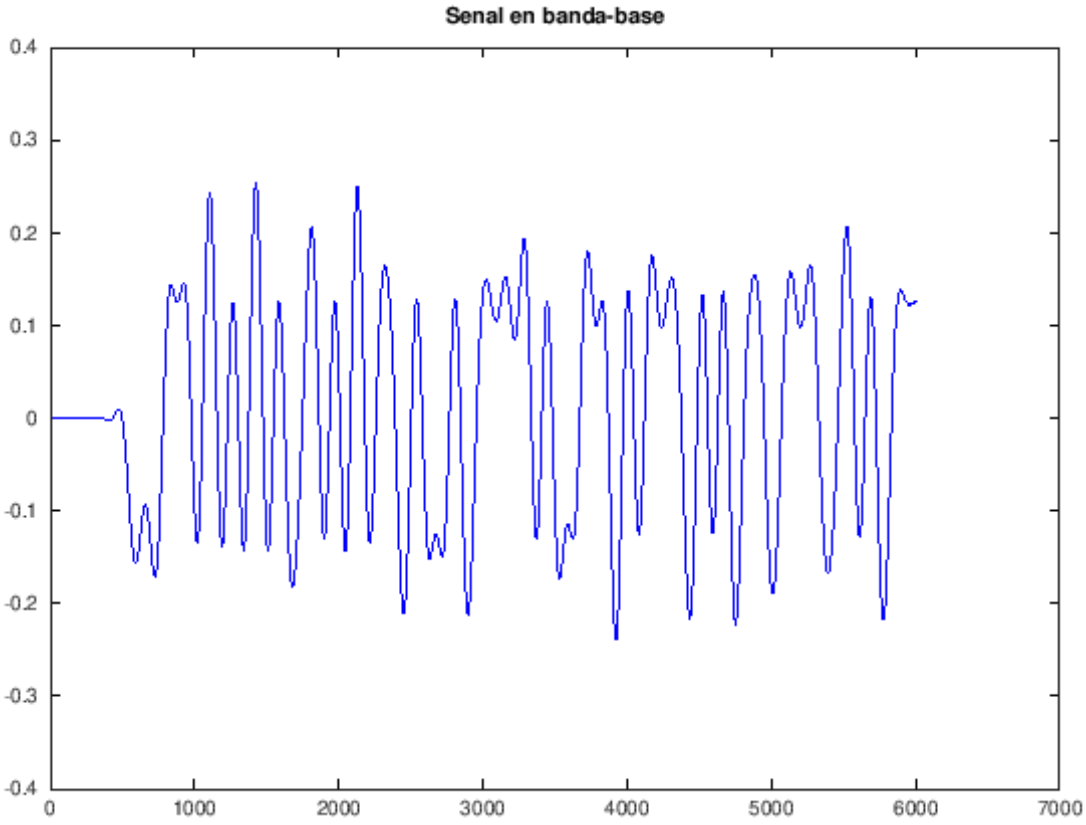
[152]:
```
f0=1.0e6;                % Ancho de banda, en Hz
M=2^4;                   % Factor de sobremuestreo del SD (sigma-delta)
R=4;                     % Factor de sobremuestreo despues del diezmado (>2 para␣
 ↪cumplir Nyquist)
fs=R*f0*M;               % Frecuencia de muestreo, in Hz
Ts=1/fs;                 %Período de muestreo
N=15000;                 % Cantidad de símbolos a generar
```

[142]:
```
% Filtro transmisor
%load('g.mat');
% rcosine de matlab y la aquí usada difieren en amplitud por un factor (fs/
 ↪f0=R*M)
g=(fs/f0)*rcosine(f0,fs,'default',.1,40);  % Filtro Tx Beta=0.1 Delay=40
%plot(g);
```

6.2.2 Generación de señal en banda-base

[143]:
```
ak = 2*(randi(2,1,N)-1)-1; % NModulador símbolos entre -1 y 1
%N*R*M es la cantidad de taps necesarios para los N símbolos sobremuestrados␣
 ↪por R*M
%fs/f0=R*M
xn = zeros(1,N*R*M); % fs/f0=R*M;
xn(1:R*M:end) = ak;
x=.125*filter(g,1,xn);%  filter(g,1,xn) equivale a conv(g,xn);
```

[144]:
```
plot(x(2000:8000),'b');
title('Senal en banda-base')
```



6.3 C. Modulador $\Sigma\Delta$ y filtro CIC

Aquí se presenta el código provisto por el docente para el modulador y el filtro . Se agregan algunos comentarios y pequeñas modificaciones.

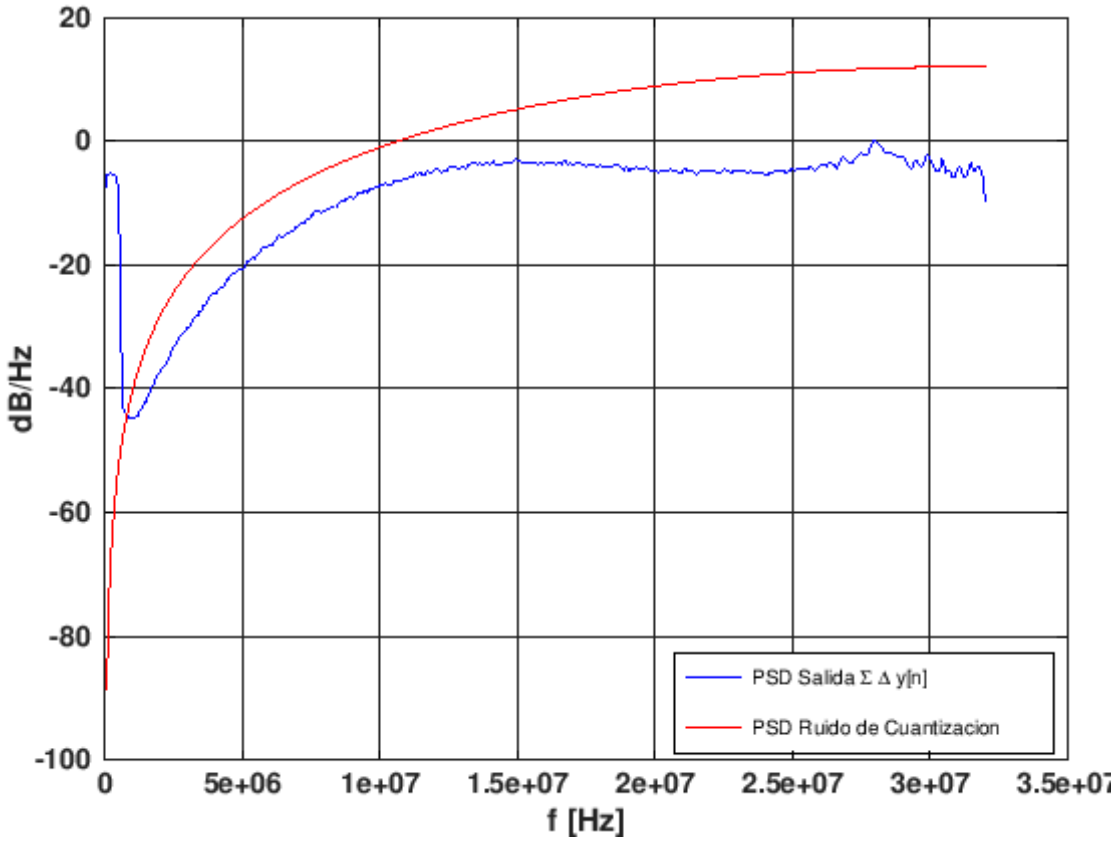
Notar que simplemente comentando el código correspondiente al cuantizador de 1 bit, y reemplazarlo por un equivalente que símplemente agrega ruido blanco a $s_2[n]$ obtendríamos resultados equivalentes

[145]:
```
e1=zeros(1,N*M);
e2=zeros(1,N*M);
s1=zeros(1,N*M);
s2=zeros(1,N*M);
y=ones(1,N*M);
xhat=zeros(1,N);
xhat1=zeros(1,N);
xhat2=zeros(1,N);
k=2;
y_acum1=0;
y_acum2=0;
```

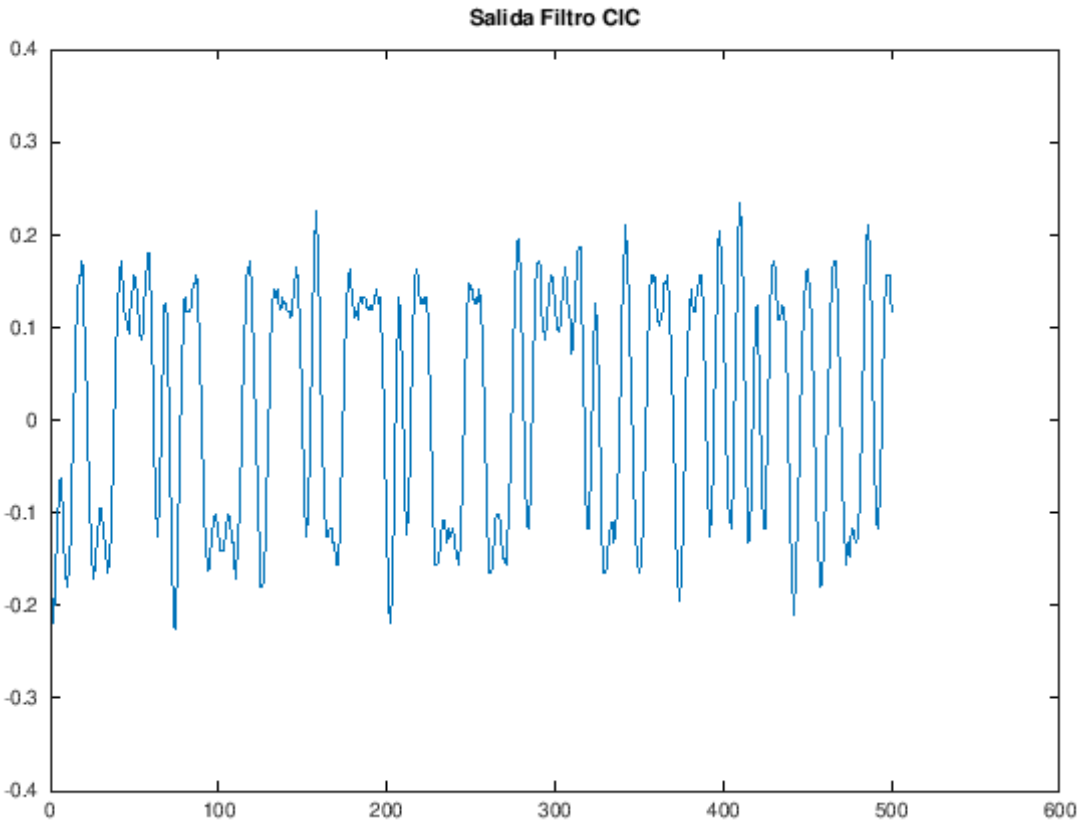
```
for n=2:N*M
    e1(n)=x(n)-y(n-1);    %diferencia
    s1(n)=e1(n)+s1(n-1); % integrador
    e2(n)=s1(n)-y(n-1);  %diferencia
    s2(n)=e2(n)+s2(n-1);  % integrador

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Mofificar aquí para comprobar
    % equivalencia de cuantizador de 1 bit con
    % la adicion de ruido-blanco
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    y(n)=(s2(n)>0)-(s2(n)<=0); % cuantizador de 1 bit
    %y(n)=s2(n)*1.0+4.*(rand-.5); % Ruido blanco Reemplazando efecto del␣
 ↪cuantizador de 1 bit


    % La senal de entrada del CIC es positiva [0 2] con media 1!
    y_acum1=mod(y_acum1+y(n)+1,2*M^2);
    y_acum2=mod(y_acum2+y_acum1,2*M^2);
    if mod(n,M)==0
        xhat1(k)=y_acum2;
        xhat2(k)=mod(xhat1(k)-xhat1(k-1),2*M^2);
        xhat(k)=mod(xhat2(k)-xhat2(k-1),2*M^2);
        k=k+1;
    endif
endfor
xhat=xhat(2000:end)/M^2-1; % Ajuste de ganancia (DC=1) y eliminacion de continua
%load('xhat.mat');
```

6.3.1 Salida ΣΔ

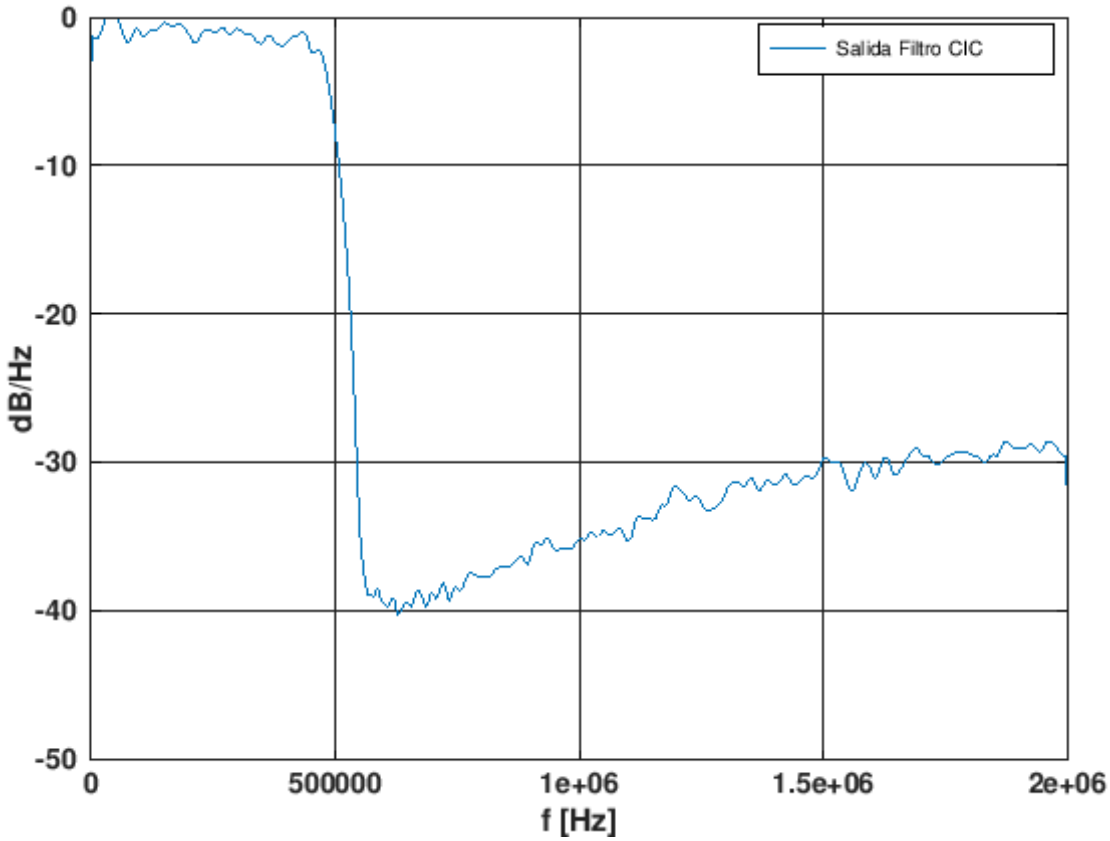
[146]:
```
%[H F]=psd(y(2000:end),2^10,fs); %matlab antiguo
%[H, F]=pwelch(y(2000:end), [], 512 ,2^10,fs); % matlab nuevo
[H, F]=pwelch(y(2000:end),2^10, [], 2^10, fs);
ruido_cuant=20*log10(abs((1-exp(-j*2*pi*F*Ts)).^2));
h=plot(F,10*log10(H/max(H)),'b',F,ruido_cuant,'r');
m=legend('PSD Salida \Sigma \Delta y[n]','PSD Ruido de Cuantizacion');
set(m,'location','southeast');
set(h,'Linewidth',2);
set(h,'Markersize',16);
set(gca,'XScale','lin','YScale','lin','FontWeight','bold','FontSize',14);
set(gca,'Linewidth',2);
xlabel('f [Hz]');
ylabel('dB/Hz');
grid on
```



[147]:
```
plot(xhat(3000:3500));
 title('Salida Filtro CIC');
```



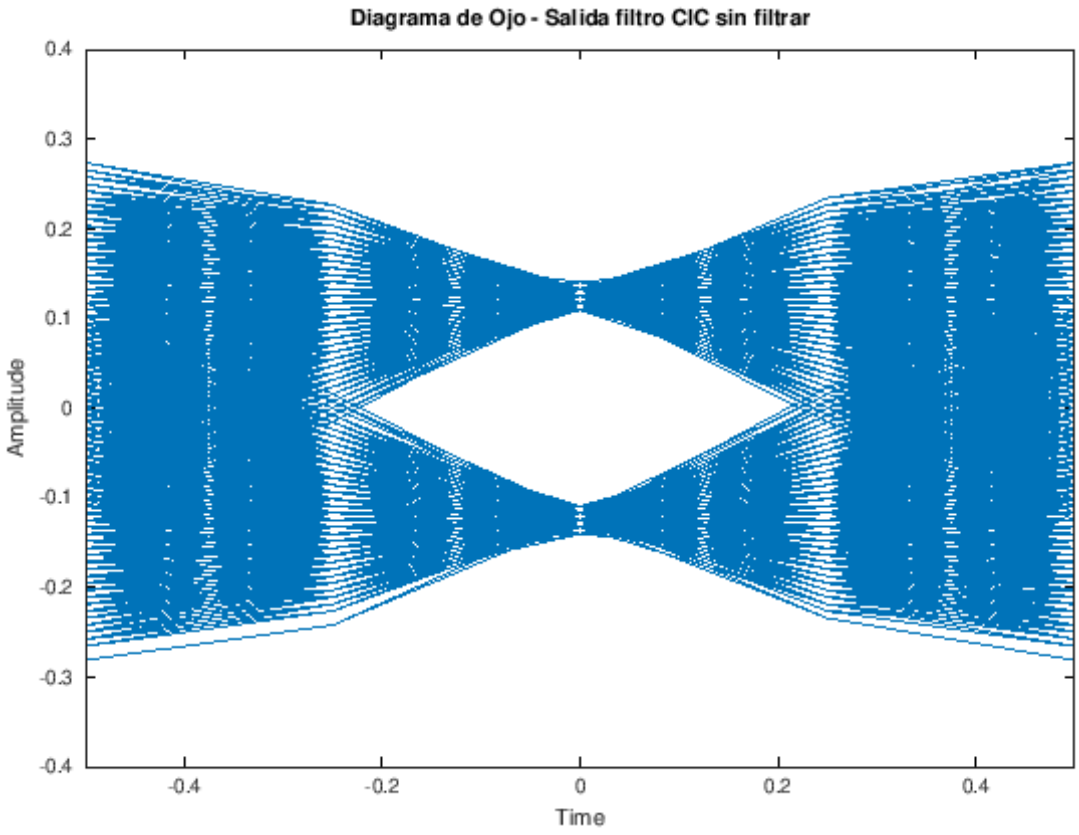
[148]:
```
%[H F]=psd(xhat,2^10,fs/M);
%[H, F]=pwelch(xhat, [], 512 ,2^10,fs/M); %matlab nuevo
[H, Fd]=pwelch(xhat,[], [], 2^10, fs/M);
h=plot(Fd,10*log10(H/max(H))); %salida filtro CIC xhat
m=legend('Salida Filtro CIC ');

set(h,'Linewidth',2);
set(h,'Markersize',16);
set(gca,'XScale','lin','YScale','lin','FontWeight','bold','FontSize',14);
set(gca,'Linewidth',2);
xlabel('f [Hz]');
ylabel('dB/Hz');
grid on
```



6.4 D. Diagrama de Ojo salida filtro CIC sin filtrar

Veremos en el próximo apartado como modificando los parámetros R y M podemos mejorar la apertura del ojo.

[149]:
```
eyediagram(xhat,R,1,2);
title('Diagrama de Ojo - Salida filtro CIC sin filtrar')
```



6.5 E. Filtro pasabajos a la salida del filtro CIC para reducir ruido de cuantización

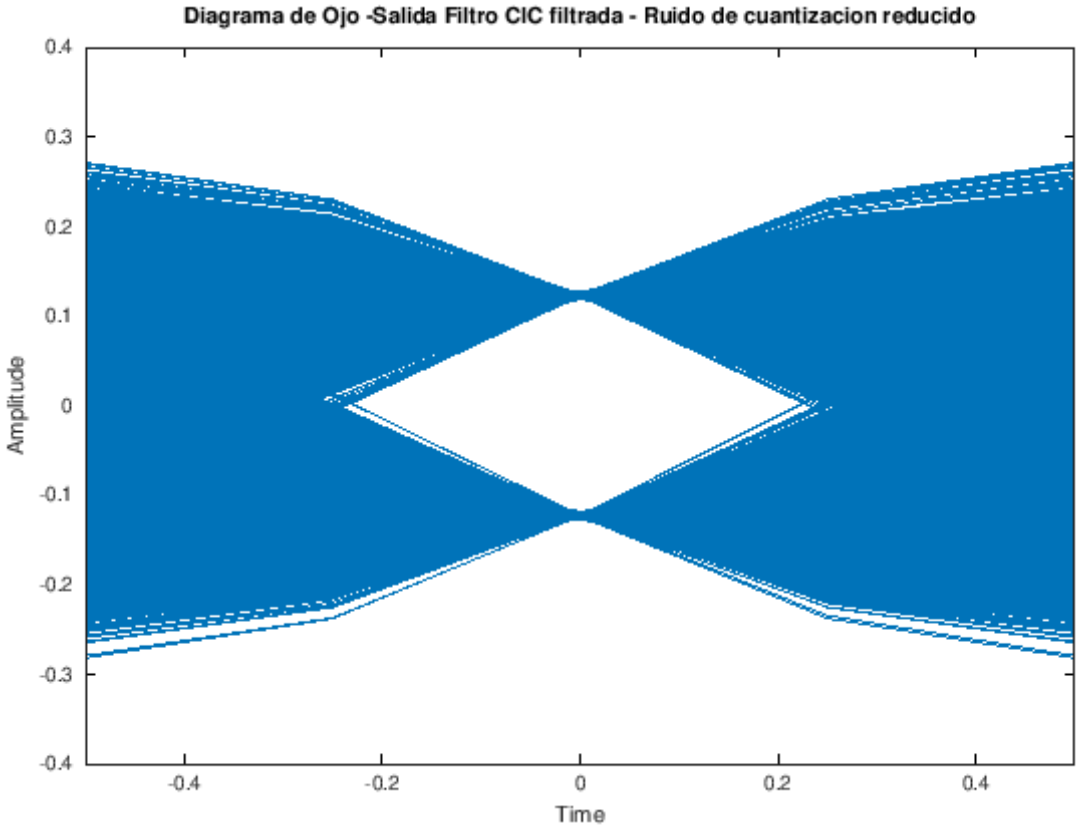
El código provisto por el docente para la creación del filtro está modificado por uno equivalente con el único propósito de entender mejor lo que hacía esa parter del código. (ver comentarios).

Se nos pide g raficar el diagrama de ojo de la señal $\hat{x}[n]$ filtrada por un filtro de caida cosenoidal(pasa bajos), similar a $g[n]$ pero sobremuestreado por un factor R.

Vemos aquí como usando un filtro pasabajos y ajustando los valores de M y R (codificados en el siguiente código como un factor de ajuste aplicado sobre la frecuencia de sobremuestreo en R del filtro pasabajos)

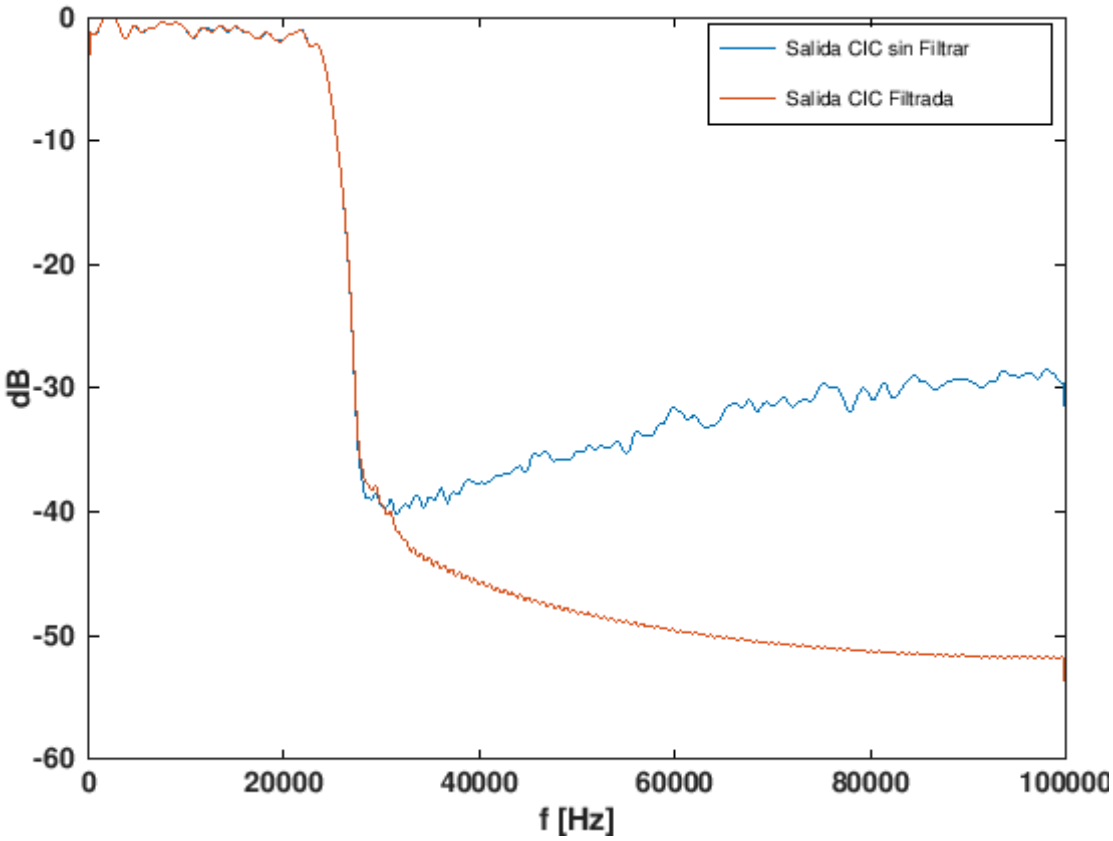
[164]:
```
factor_ajuste=1.25; %   Modificar este valor nos permite simular para distintos␣
 ↪valores de R*M ?
%n=[-30:factor_ajuste/R:30];
%f=sinc(n); % Filtro pasabajo (saca ruido de cuantizacion fuera de la banda de␣
 ↪la senal)

% Aquí para comprobar si había entendido bien lo que hacía el fitro anterior ,␣
 ↪cree el filtro con rcosine cambiando fs
```

```
%usamos  fs/M ya que la señal fue diezmada en un factor M en el filtro CIC␣
 ↪quedando R unicaente como factor de sobremuestreo
fs_filtro=(1/factor_ajuste)*fs/M; % equivalente a escribir (R/factor_ajuste)*f0

f=(fs_filtro/f0)*rcosine(f0,fs_filtro,'default',.1,40);  % Filtro Tx Beta=0.1␣
 ↪Delay=30
xhat_f=filter(f/sum(f),1,xhat);
eyediagram(xhat_f(50*R:end),R,1,2);
title('Diagrama de Ojo -Salida Filtro CIC filtrada - Ruido de cuantizacion␣
 ↪reducido')
```



[151]:
```
[H, F]=pwelch(xhat, [], [] ,2^10,fs/M);
[Hf, F]=pwelch(xhat_f, [],[]  ,2^10,fs/M);
h=plot(F,10*log10(H/max(H)),F,10*log10(Hf/max(Hf)));
legend('Salida CIC sin Filtrar','Salida CIC Filtrada')
set(h,'Linewidth',2);
set(h,'Markersize',16);
set(gca,'XScale','lin','YScale','lin','FontWeight','bold','FontSize',14);
set(gca,'Linewidth',2);
xlabel('f [Hz]');
```

```
ylabel('dB');
```


Salida CIC sin Filtrar
Salida CIC Filtrada
0
-10
-20
-30
-40
-50
-60
dB
0
20000
40000
60000
80000
100000
f [Hz]