T.P.Nº 2: Introducción al Procesamiento Digital de Señales

2.1- Señales básicas utilizadas en el procesamiento digital de señales

2.1.1- Señal impulso unitario (desplazado)

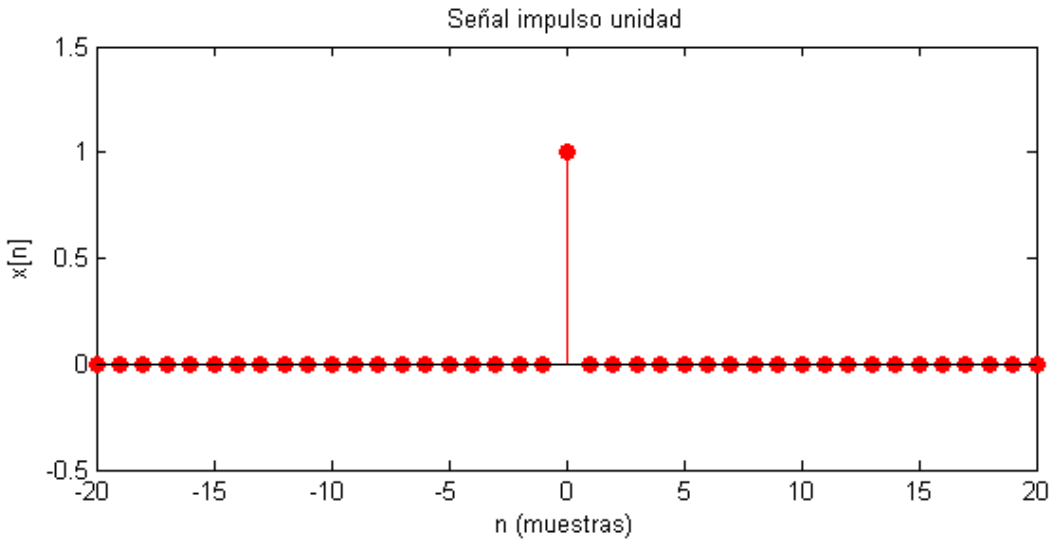
Genere en lenguaje script de Matlab y represente gráficamente una señal impulso la cual se usará para excitar un sistema lineal invariante en el tiempo causal de modo que se obtengan L muestras a la salida de dicho sistema.

n_o = 0
L = 20

Resolución:

El siguiente script genera una señal impulso unitario en MatLab:

```
% Cantidad de muestras
L=20;
% Vector para graficar el impulso
n= -L:1:L;
% Señal impulso todo en cero
impulso=zeros(size(n));
% Cargo el elemento correspondiente a la pos 0 en 1
impulso(n==0)=1;
% Le indico que voy a crear un obejto que va ser la figura numero 1
figure(1);
% grafico en la figura 1 el impulso
stem(n,impulso,'r','fill');
% Seteo los ejes del grafico
axis([-L L -0.5 1.5]);
title('Señal impulso unidad');
xlabel('n (muestras)');
ylabel('x[n]');
```



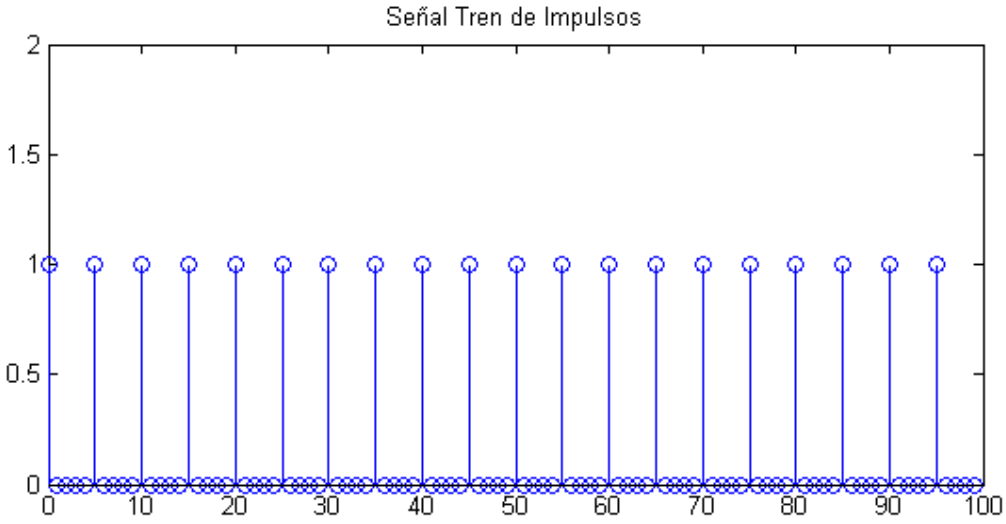
2.1.2 Tren de impulsos

Genere y represente gráficamente mediante Matlab un tren de impulsos de periodo 5 y longitud 100.

Resolución:

El siguiente script genera una señal tren de impulsos en MatLab:

```
% Tren de impulsos

P = 5;
L = 100;
M = L/P;
tren_imp=zeros(1,L);
for n=0:L-1
    for i=0:M-1
        if n==(i*P)
            tren_imp(n+1)=1;
        end
    end
end

figure(1);
n=0:L-1;
stem(n,tren_imp);
axis([0 100 0 2]);
title('Señal Tren de Impulsos');
```



2.1.3 Escalón unitario

El escalón unitario tiene como expresión

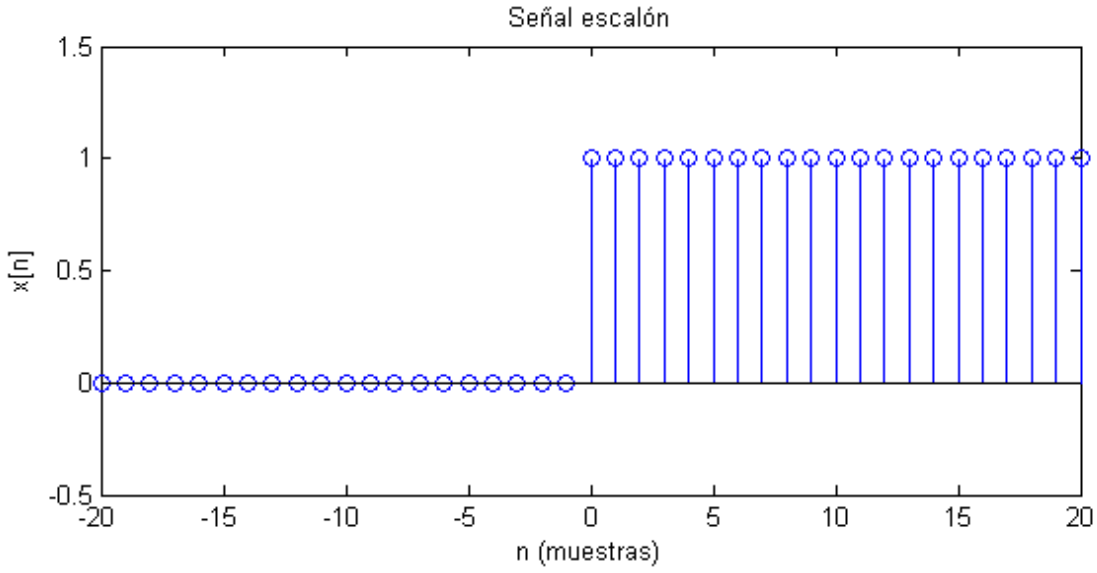
u[n] = 1 para n >= 0
u[n] = 0 para n < 0.

Genere y represente gráficamente la secuencia escalón unitario de longitud L=20.

Resolución:

El siguiente script genera una señal escalón unitario de longitud L=20 en MatLab:

```
% Escalon unitario

L = 20;

% Vector para graficar el escalón
n= -L:1:L;

% Crear señal "escalon_unitario" y setear todos sus elementos a cero
escalon_unitario = zeros(size(n));

% Setear todos los elementos con n>=0 a 1
escalon_unitario(n>=0) = 1;

figure(1);

%Grafico en la figura 1 el escalón
stem(n,escalon_unitario);

axis([-L L -0.5 1.5]);
title('Señal escalón');
xlabel('n (muestras)');
ylabel('x[n]');
```



2.1.4 Señal sinusoidal

Una señal sinusoidal de amplitud A, frecuencia ωo y fase φ en tiempo continuo esta dada por:

$X(t) = A \operatorname{sen}(\omega_o t + \varphi)$

En tiempo discreto t=nTs, donde Ts es el periodo de muestreo, entonces:

X[n] = A sen (ωo nTs + φ) = A sen (2 π fo/fs n + φ); ωo = 2 π fo

Donde a fo/fs se la denomina frecuencia normalizada y fs es la frecuencia de muestreo.

Genere y represente gráficamente mediante la función stem una señal cosenoidal en tiempo discreto de N muestras, cuyos parámetros son:

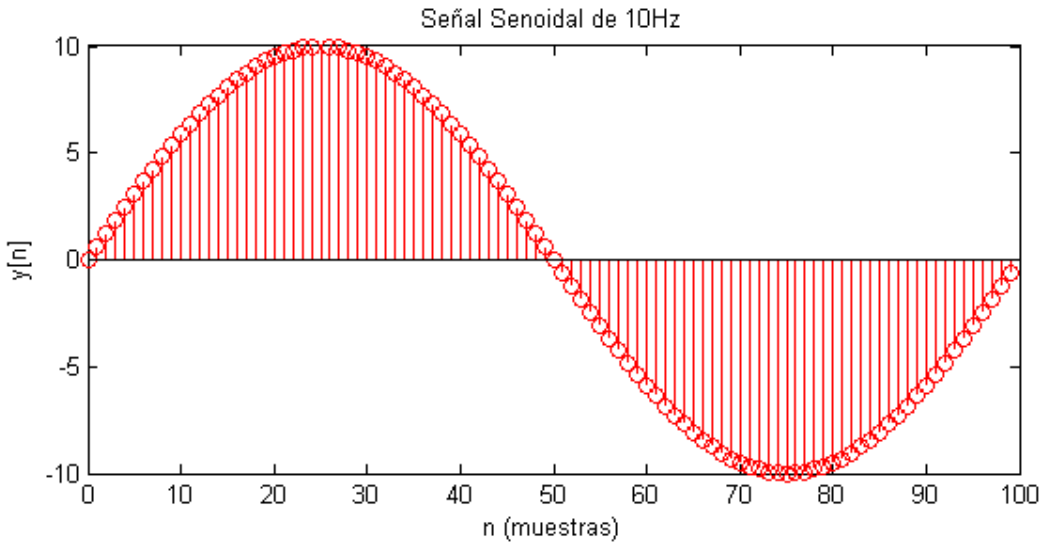
A = 10,
fo = 10 Hz,
fs = 1000 Hz,
φ = 0°

N = 100.

Resolución:

El siguiente script genera una señal senoidal de N=100 muestras en MatLab:

```
% Señal senoidal

% Señal en tiempo continuo
% X(t) = A sen(W0*t+phi)

% Señal discreta -> t = nTs, donde Ts es el período de muestreo
% X[n] = A sen (W0*nTs+hi) = A sen(2*pi*f0/fs*n +phi) W0 = 2*pi*f0
% fn = f0/fs es la frecuencia normalizada y fs la frecuencia de muestreo

A = 10;      % Amplitud
f0 = 10;     % Frec de la señal en Hz
fs = 1000;   % Frec de muestreo en Hz
phi = 0;     % pi/2; % Fase

N=100; % Número de muestras
n=0:N-1;

y=A*sin(2*pi*(f0/fs)*n+phi); % Señal senoidal
figure(1);
stem(n,y,'r');

title('Señal Senoidal de 10Hz');
xlabel('n (muestras)');
ylabel('y[n]');
```



2.1.5 Señales exponenciales

Una señal exponencial en tiempo discreto es de la forma:

$X[n] = a^n$
siendo a un número real y n= 0, 1, 2, 3, 4...

Si $0 < a < 1$ o $-1 < a < 0$, entonces la exponencial es decreciente
Si $a > 1$ o $a < -1$, entonces la exponencial es creciente

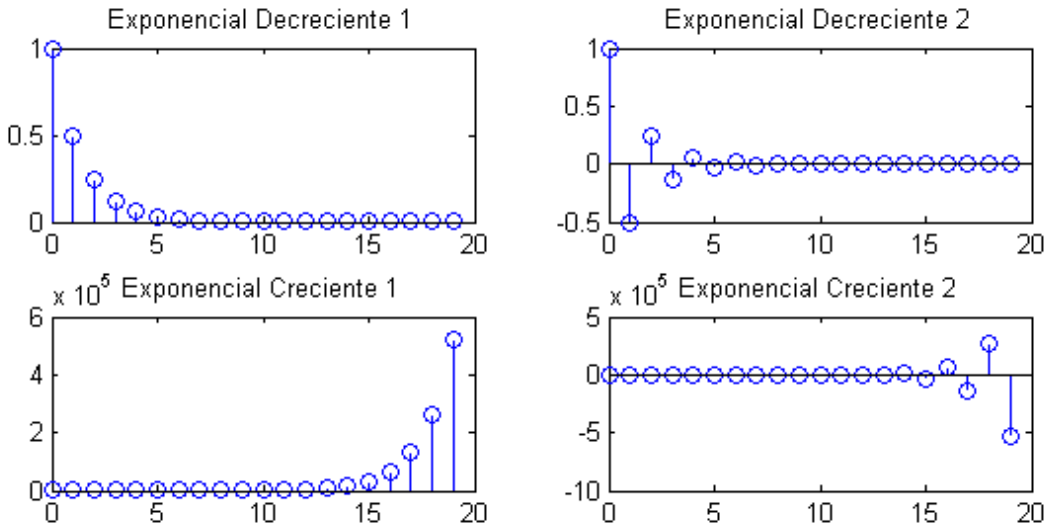
Genere y represente gráficamente las señales exponenciales decrecientes y crecientes de longitud L=20.

Resolución:

El siguiente script genera 4 señales exponenciales de longitud L=20 y las grafica juntas:

```
% Exponenciales tiempo discreto

figure(1);

L = 20;

% Exponencial Decreciente 1 con 0 < a < 1
a=0.5;
exp1 = zeros(1,L);
for n=0:L-1
    exp1(n+1)=a^n;
end

n=0:L-1;
subplot(2,2,1),
stem(n,exp1),
title('Exponencial Decreciente 1');

% Exponencial Decreciente 2 con -1 < a < 0
a=-0.5;
exp2 = zeros(1,L);
for n=0:L-1
    exp2(n+1)=a^n;
end

n=0:L-1;
subplot(2,2,2),
stem(n,exp2),
title('Exponencial Decreciente 2');

% Exponencial Creciente 1 con a > 1
a=2;
exp3 = zeros(1,L);
for n=0:L-1
    exp3(n+1)=a^n;
end

n=0:L-1;
subplot(2,2,3),
stem(n,exp3),
```

```
title('Exponencial Creciente 1');

% Exponencial Creciente 2 con a < -1
a=-2;
exp4 = zeros(1,L);
for n=0:L-1
    exp4(n+1)=a^n;
end

n=0:L-1;
subplot(2,2,4),
stem(n,exp4),
title('Exponencial Creciente 2');
```



2.1.6 Señales complejas

Si bien en el mundo real las señales tienen valores reales, es muy frecuente el uso de señales con partes real e imaginaria, es decir señales complejas, por ejemplo en sistemas de modulación.

Para el caso en que el valor a de la señal exponencial sea un valor complejo, tendremos:

$a = r.e^{j.\Theta}$

por lo tanto

$X[n] = r^n.e^{j.\Theta.n} = r^n (\cos \Theta.n + j.\text{sen } \Theta.n)$

Gráficamente se podrá ahora representar una parte real:

$XR[n] = r^n . \cos \Theta.n$

y una parte imaginaria:

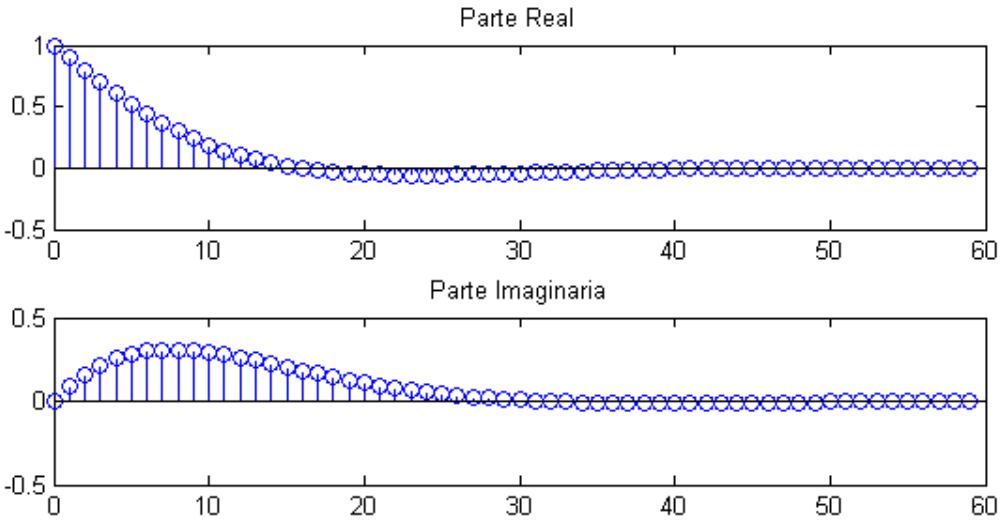
$XI[n] = r^n . \text{sen } \Theta.n$

Genere una señal exponencial compleja donde r = 0.9 y $\Theta = \pi/10$. ¿Que sucede sí r = 1?

Resolución:

El siguiente script genera una señal exponencial compleja con r=0,9 y N=60 muestras en MatLab:

```
% Exponencial compleja

figure(1)

L=60;

r=0.9;
Xre = zeros(1,L);
Xim = zeros(1,L);

for n=0:L-1
    Xre(n+1)=(r^n)*cos((1/10)*n);
    Xim(n+1)=(r^n)*sin((1/10)*n);
end

n=0:L-1;
subplot(2,1,1),
stem(n,Xre),
title('Parte Real');

subplot(2,1,2),
stem(n,Xim),
title('Parte Imaginaria');
```



Si r=1, la parte real se convierte en una señal cosenoidal pura, y la parte imaginaria en una señal senoidal pura.

2.2- Muestreo de señales en el dominio del tiempo – Aliasing

2.2.1 Represente la señal senoidal tomando

fo = 100 Hz ; frecuencia de la sinusoide

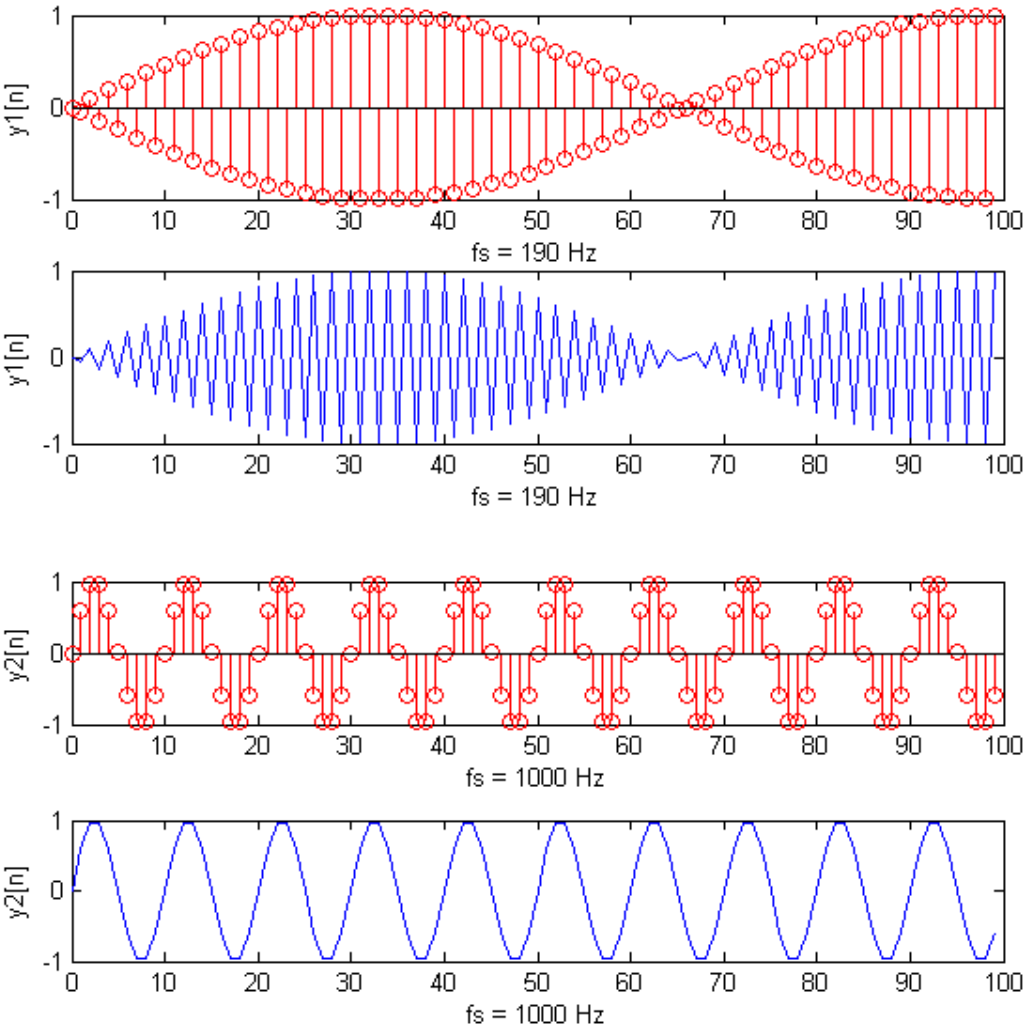
Ts = 1 mS ; periodo de muestreo
A = 1 ; amplitud

La fase puede ser cualquier valor entre 0 y 2Pi.
Utilice la función stem para visualizar la función muestreada x[n].
Luego utilice la función plot para visualizar la función de modo que las muestras sean unidas (interpoladas) mediante líneas rectas.
Pruebe con fs próxima a 2fo y con fs >> fo, por ejemplo fs=10Khz.

Resolución:

El siguiente script grafica una función senoidal muestreada a 4 frecuencia distintas, donde se puede apreciar el aliasing para frecuencias menores a 2fo:

```
% Muestreo a distintas frecuencias de Señales senoidales

A = 1;       % Amplitud
f0 = 100;    % Frec de la señal en Hz
phi = 0;     % pi/2; % Fase


%Obsérvese el aliasing en la siguiente señal
figure(1);
subplot(2,1,1);
n=0:N-1;
fs = 197;  % Nueva frec de muestreo en Hz
y1=A*sin(2*pi*(f0/fs)*n+phi); % Señal senoidal
stem(n,y1,'r');
xlabel('fs = 190 Hz');
ylabel('y1[n]');
subplot(2,1,2);
plot(n,y1);
xlabel('fs = 190 Hz');
ylabel('y1[n]');

%Señal sin aliasing
figure(2);
subplot(2,1,1);
n=0:N-1;
fs = 1000;  % Nueva frec de muestreo en Hz
y2=A*sin(2*pi*(f0/fs)*n+phi); % Señal senoidal
stem(n,y2,'r');
xlabel('fs = 1000 Hz');
ylabel('y2[n]');
subplot(2,1,2);
plot(n,y2);
xlabel('fs = 1000 Hz');
ylabel('y2[n]');
```



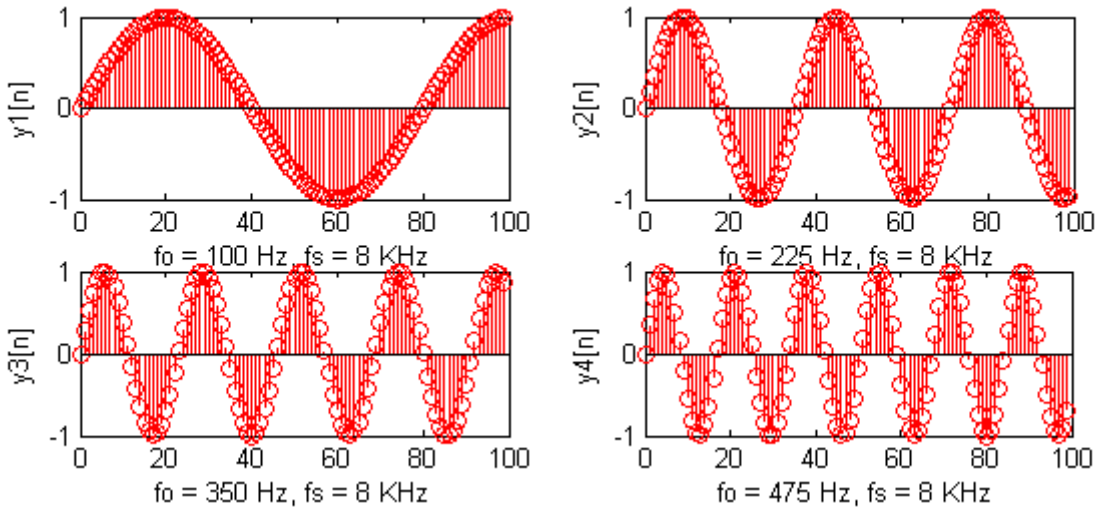
2.2.2 Varíe fo entre 100 Hz y 475 Hz en saltos de 125Hz, para fs = 8 Khz.
Utilice la función subplot para representar los cuatro casos en una misma ventana.
¿Que sucede con la frecuencia de la señal senoidal?

Resolución:

El siguiente script grafica funciones senoidales de distintas frecuencias muestreadas a 8 KHz:

```
% Muestreo de señales senoidales

A = 1;       % Amplitud
fs = 8000;    % Frec de muestreo en Hz
phi = 0;     % pi/2; % Fase
N = 100;     % Nueva cantidad de muestras para mantener la escala
n=0:N-1;

figure(1);

subplot(2,2,1);
f0 = 100;
y1=A*sin(2*pi*(f0/fs)*n+phi); % Señal senoidal
stem(n,y1,'r');
xlabel('fo = 100 Hz, fs = 8 KHz');
```

```
ylabel('y1[n]');

subplot(2,2,2);
f0 = 225;
y2=A*sin(2*pi*(f0/fs)*n+phi); % Señal senoidal
stem(n,y2,'r');
xlabel('fo = 225 Hz, fs = 8 KHz');
ylabel('y2[n]');

subplot(2,2,3);
f0 = 350;
y3=A*sin(2*pi*(f0/fs)*n+phi); % Señal senoidal
stem(n,y3,'r');
xlabel('fo = 350 Hz, fs = 8 KHz');
ylabel('y3[n]');

subplot(2,2,4);
f0 = 475;
y4=A*sin(2*pi*(f0/fs)*n+phi); % Señal senoidal
stem(n,y4,'r');
xlabel('fo = 475 Hz, fs = 8 KHz');
ylabel('y4[n]');
```



Se observa que la frecuencia de la señal senoidal no se ve alterada y puede calcularse la frecuencia correcta en cualquiera de los cuatro gráficos si se conoce previamente fs. Esto se debe a que ha sido respetado el criterio de Nyquist.

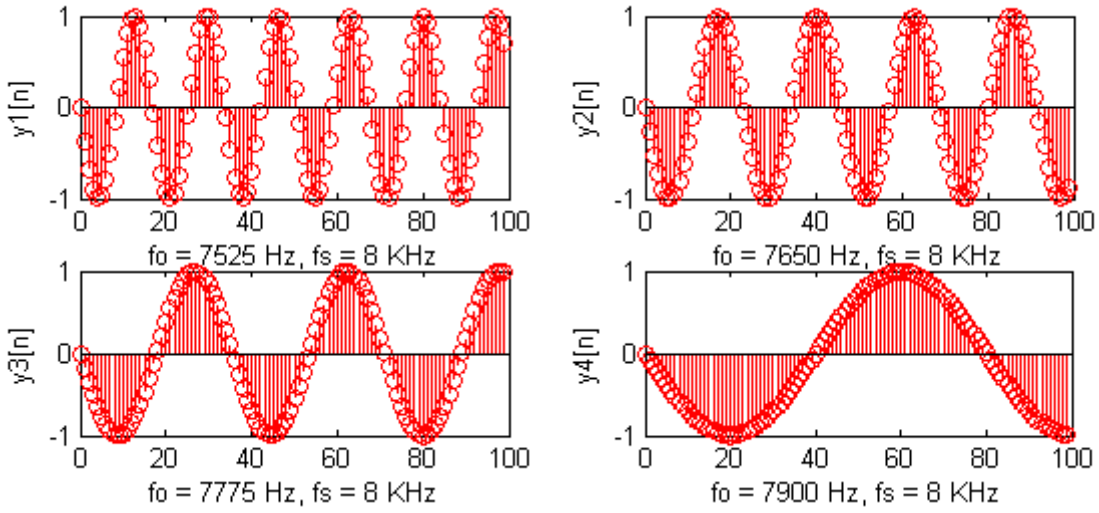
2.2.3 Varíe fo entre 7525 y 7900 en saltos de 125 Hz.
¿Que sucede ahora con la frecuencia de la señal senoidal representada?, ¿Porque?

Resolución:

El siguiente script grafica funciones senoidales de distintas frecuencias muestreadas a 8 KHz:

```
% Muestreo de señales senoidales

A = 1;       % Amplitud
fs = 8000;    % Frec de muestreo en Hz
phi = 0;     % pi/2; % Fase
```

```
N = 100;     % Nueva cantidad de muestras para mantener la escala
n=0:N-1;

figure(1);

subplot(2,2,1);
f0 = 7525;
y1=A*sin(2*pi*(f0/fs)*n+phi); % Señal senoidal
stem(n,y1,'r');
xlabel('fo = 7525 Hz, fs = 8 KHz');
ylabel('y1[n]');

subplot(2,2,2);
f0 = 7650;
y2=A*sin(2*pi*(f0/fs)*n+phi); % Señal senoidal
stem(n,y2,'r');
xlabel('fo = 7650 Hz, fs = 8 KHz');
ylabel('y2[n]');

subplot(2,2,3);
f0 = 7775;
y3=A*sin(2*pi*(f0/fs)*n+phi); % Señal senoidal
stem(n,y3,'r');
xlabel('fo = 7775 Hz, fs = 8 KHz');
ylabel('y3[n]');

subplot(2,2,4);
f0 = 7900;
y4=A*sin(2*pi*(f0/fs)*n+phi); % Señal senoidal
stem(n,y4,'r');
xlabel('fo = 7900 Hz, fs = 8 KHz');
ylabel('y4[n]');
```



En este caso se observa que al aumentar la frecuencia de la señal, en la gráfica se observa el efecto contrario, es decir, una aparente disminucion de frecuencia. Esto de debe a que, al no cumplir con el Criterio de Nyquist, se produce el fenómeno de aliasing, es decir, el solapamiento en frecuencia hace que la secuencia muestreada tenga mas de una posible reconstruccion en el tiempo.

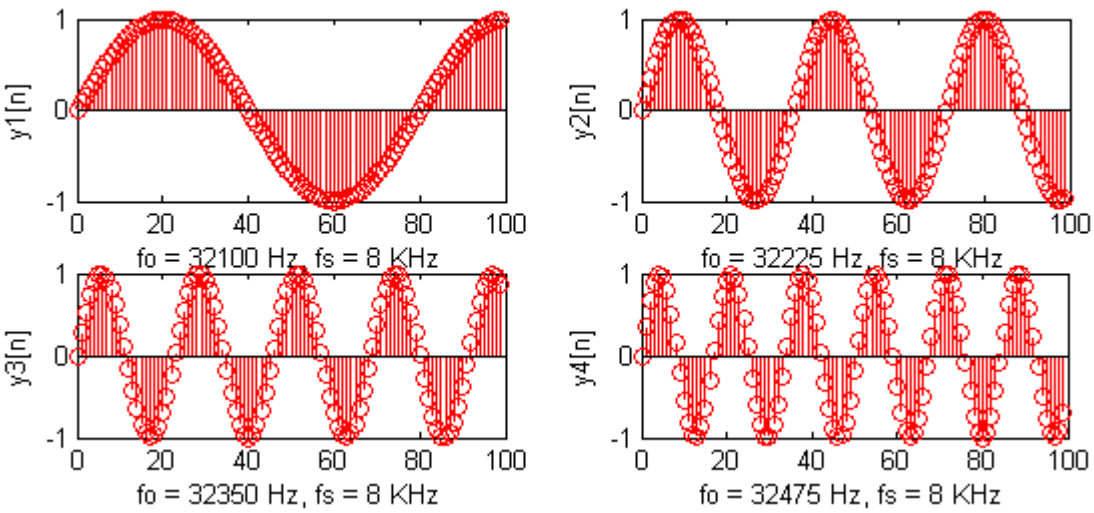
2.2.4 Nuevamente varíe fo entre 32100 Hz y 32475 Hz en saltos de 125 Hz.

¿Que sucede ahora con la frecuencia de la señal senoidal representada?, ¿Porque?

Resolución:

El siguiente script grafica funciones senoidales de distintas frecuencias muestreadas a 8 KHz:

```
% Muestreo de señales senoidales

A = 1;       % Amplitud
fs = 8000;    % Frec de muestreo en Hz
phi = 0;     % pi/2; % Fase
N = 100;     % Nueva cantidad de muestras para mantener la escala
n=0:N-1;

figure(1);

subplot(2,2,1);
f0 = 32100;
y1=A*sin(2*pi*(f0/fs)*n+phi); % Señal senoidal
stem(n,y1,'r');
xlabel('fo = 32100 Hz, fs = 8 KHz');
ylabel('y1[n]');

subplot(2,2,2);
f0 = 32225;
y2=A*sin(2*pi*(f0/fs)*n+phi); % Señal senoidal
stem(n,y2,'r');
xlabel('fo = 32225 Hz, fs = 8 KHz');
ylabel('y2[n]');

subplot(2,2,3);
f0 = 32350;
y3=A*sin(2*pi*(f0/fs)*n+phi); % Señal senoidal
stem(n,y3,'r');
xlabel('fo = 32350 Hz, fs = 8 KHz');
ylabel('y3[n]');

subplot(2,2,4);
f0 = 32475;
y4=A*sin(2*pi*(f0/fs)*n+phi); % Señal senoidal
stem(n,y4,'r');
xlabel('fo = 32475 Hz, fs = 8 KHz');
ylabel('y4[n]');
```



En este caso el aumento de la frecuencia de la señal se corresponde con las gráficas, pero las frecuencias graficadas son en realidad mucho menores a las originales. Esto de debe a que, al no cumplir con el Criterio de Nyquist, se produce el fenómeno de aliasing, es decir, el solapamiento en frecuencia hace que la secuencia muestreada tenga mas de una posible reconstruccion en el tiempo.

2.3- Muestreo multifrecuencia: diezmado e interpolación.

2.3.1 Analice las siguientes funciones de Matlab: decimate, interp y resample.

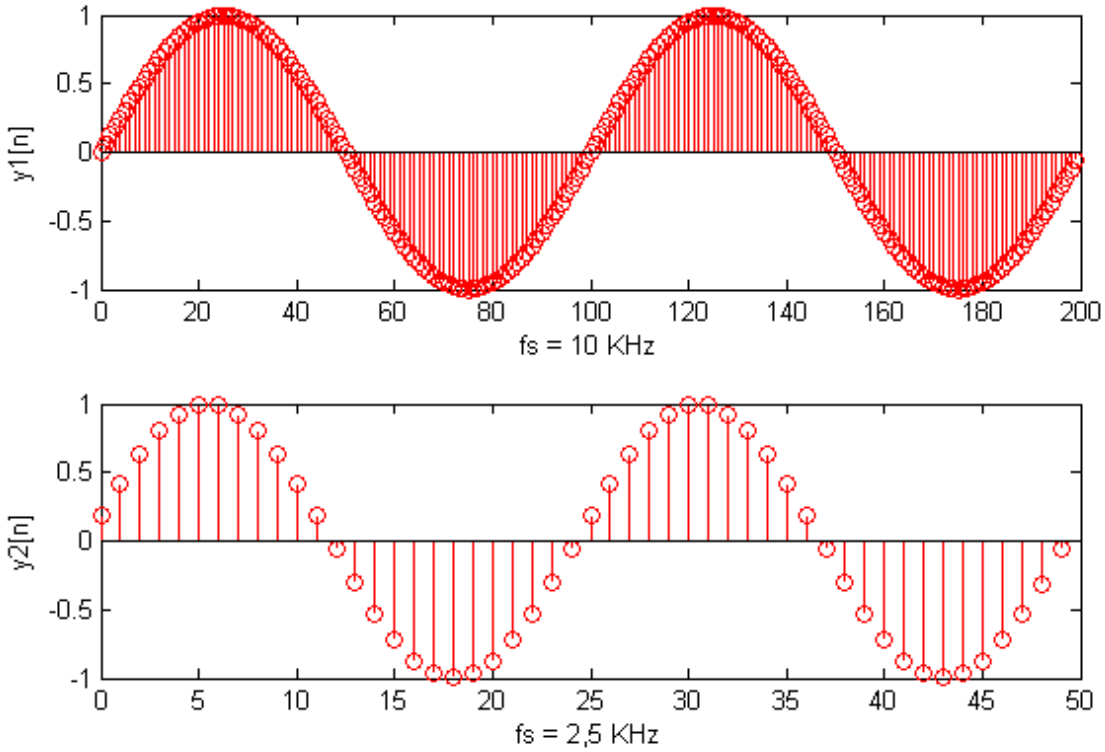
2.3.2 Genere una señal senoidal de frecuencia fo=100Hz, muestreada con una Fs=10KHz.

2.3.2.1 Decremente el número de muestras de la señal senoidal con un factor de 4.

Resolución:

El siguiente script grafica una función senoidal de 100 Hz muestreada a 10 Khz y luego la misma señal, pero muestreada a una frecuencia de 2,5 KHz:

```
% Muestreo y decimación de señales senoidales

A = 1;       % Amplitud
f0 = 100;    % Frecuencia de la senoide
fs = 10000; % Frec de muestreo en Hz
phi = 0;     % pi/2; % Fase
N = 200;     % Nueva cantidad de muestras para mantener la escala
n=0:N-1;

figure(1);

subplot(2,1,1);
y1=A*sin(2*pi*(f0/fs)*n+phi); % Señal senoidal
stem(n,y1,'r');
xlabel('fs = 10 KHz');
ylabel('y1[n]');

subplot(2,1,2);
y2 = decimate(y1,4);
N = 50;
n=0:N-1;
stem(n,y2,'r');
xlabel('fs = 2,5 KHz');
ylabel('y2[n]');
```



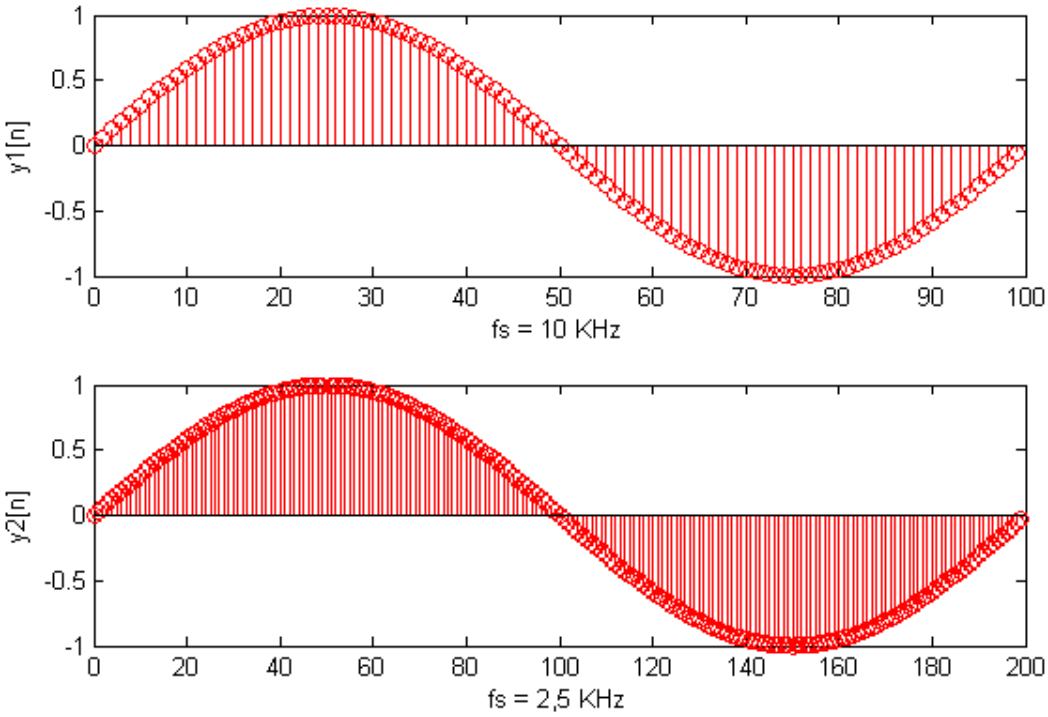
2.3.2.2 Incremente el número de muestras de la señal senoidal con un factor de 2.

Resolución:

El siguiente script grafica una función senoidal de 100 Hz muestreada a 10 Khz y luego la misma señal, pero muestreada a una frecuencia de 20 KHz:

```
% Muestreo e interpolación de señales senoidales

A = 1;       % Amplitud
f0 = 100;    % Frecuencia de la senoide
fs = 10000; % Frec de muestreo en Hz
phi = 0;     % pi/2; % Fase
N = 100;     % Nueva cantidad de muestras para mantener la escala
n=0:N-1;

figure(1);

subplot(2,1,1);
y1=A*sin(2*pi*(f0/fs)*n+phi); % Señal senoidal
stem(n,y1,'r');
xlabel('fs = 10 KHz');
ylabel('y1[n]');

subplot(2,1,2);
y2 = interp(y1,2);
N = 200;
n=0:N-1;
stem(n,y2,'r');
axis([0 200 -1 1]);
```

```
xlabel('fs = 2,5 KHz');
ylabel('y2[n]');
```



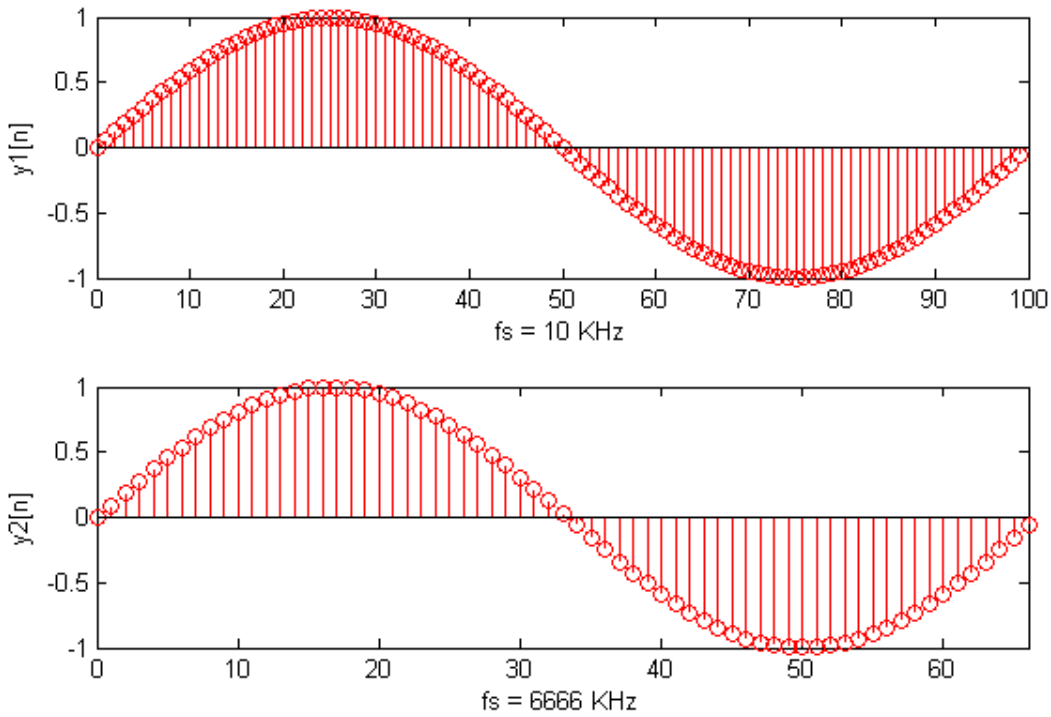
2.3.2.3 Modifique la frecuencia de muestreo en un factor de 2/3.

Resolución:

El siguiente script grafica una función senoidal de 100 Hz muestreada a 10 Khz y luego la misma señal, pero muestreada a una frecuencia de 6666 Hz:

```
% Muestreo e interpolación de señales senoidales

A = 1;       % Amplitud
f0 = 100;    % Frecuencia de la senoide
fs = 10000; % Frec de muestreo en Hz
phi = 0;     % pi/2; % Fase
N = 100;     % Nueva cantidad de muestras para mantener la escala
n=0:N-1;

figure(1);

subplot(2,1,1);
y1=A*sin(2*pi*(f0/fs)*n+phi); % Señal senoidal
stem(n,y1,'r');
xlabel('fs = 10 KHz');
ylabel('y1[n]');

subplot(2,1,2);
y2 = resample(y1,2,3);
N = 67;
n=0:N-1;
```

```
stem(n,y2,'r');
axis([0 66 -1 1]);
xlabel('fs = 6666 KHz');
ylabel('y2[n]');
```



2.4- Manipulaciones simples de señales en tiempo discreto

2.4.1 Desplazamiento en el tiempo.

Introducción:
Sea una señal discreta x(n), esta puede ser desplazada en el tiempo reemplazando la variable independiente n por n-k donde k es un entero.

Si k es un entero positivo el desplazamiento provoca un retraso de la señal en k unidades de tiempo.
Si k es un entero negativo el desplazamiento temporal resulta en un adelanto de la señal en |k| unidades de tiempo.
Note que en el muestreo en tiempo real es imposible obtener un adelanto, es decir, no podemos contar con muestras futuras de la señal.

2.4.1.1 Dado un sistema al cual se le aplica una señal x(n) y cuya salida es y(n) = x(n-2), dicho sistema, ¿adelanta o retarda?

Resolución: El sistema se retrasa en (Ts * 2) segundos.

2.4.1.2 Dadas las siguientes muestras de una señal,

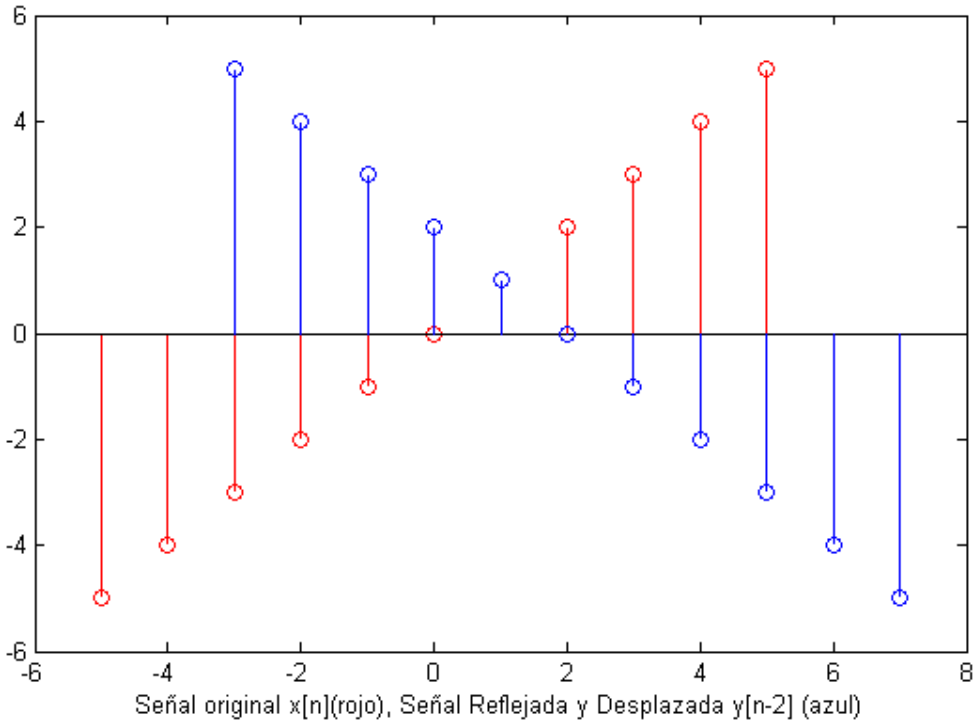
x[n] = [0 3 2 1 **0** 1 2 3 0]; el elemento en negrita indica el origen de tiempo, es decir, n=0.

Grafique en una misma figura x(n) e y(n) = x(n-k) siendo k=2.

Resolución:

El siguiente script grafica una secuencia de valores y luego ésta misma secuencia, pero desplazada (retardada) en el tiempo:

```
% Reflexión de señales

x = [-5 -4 -3 -2 -1 0 1 2 3 4 5 ];
k=-2;

n=-5:5;
figure(1);

stem(n,x,'r');
hold;

n = -n;
n = n-k;

stem(n,x,'b');
xlabel('Señal original x[n](rojo), Señal Reflejada y Desplazada y[n-2]
(azul)');
axis([-6 8 -6 6]);
```



2.4.2 Reflexión

Esta operación consiste en cambiar la variable independiente n por –n, el resultado de esto es la reflexión de la señal con respecto al origen de tiempos n = 0.

2.4.2.1 Dada la siguiente señal,

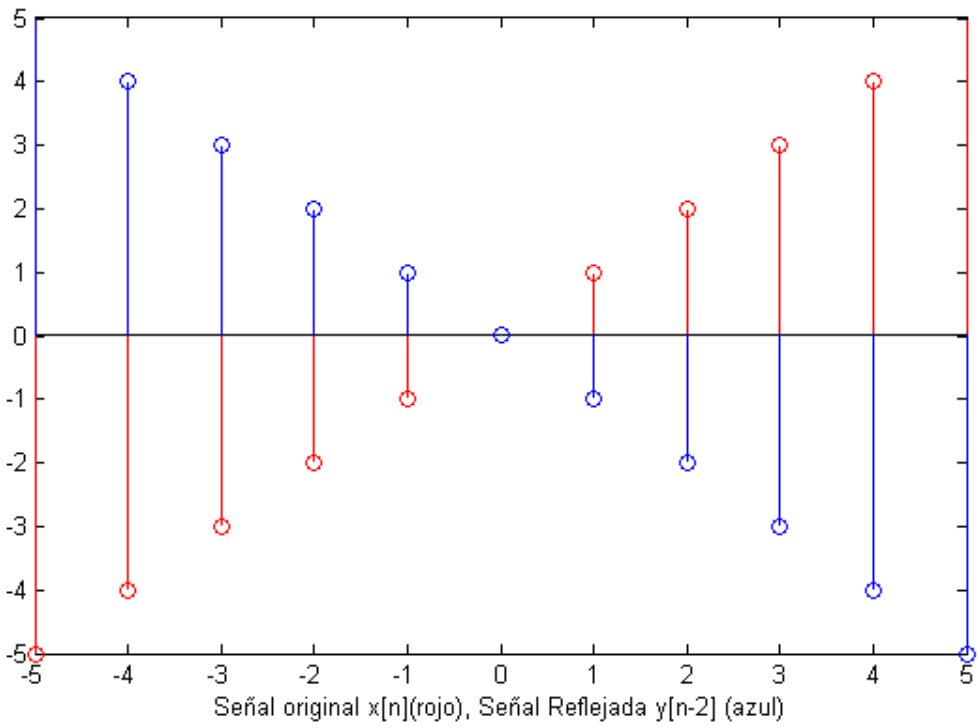
x[n] = [-5 –4 –3 –2 –1 **0** 1 2 3 4 5]

Obtenga la reflexión y(-n) de x(n). Grafique ambas señales en una misma figura.

Resolución:

El siguiente script grafica una secuencia de valores y luego ésta misma secuencia reflajada:

```
% Reflexión de señales

x = [-5 -4 -3 -2 -1 0 1 2 3 4 5 ];

n=-5:5;
figure(1);

stem(n,x,'r');
hold;

n = -n;

stem(n,x,'b');
xlabel('Señal original x[n](rojo), Señal Reflejada y[n-2] (azul)');
```

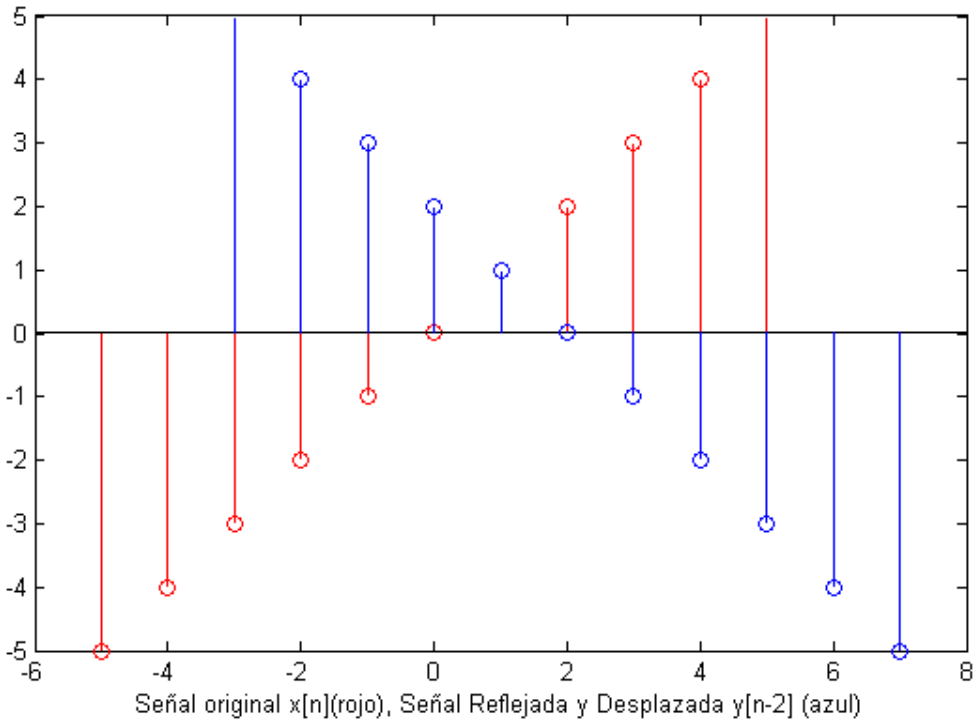


2.4.2.2 Dada una señal x(n), ¿Qué operaciones debe realizar para obtener y(n) = x(-n+2)? Compruebe su deducción en forma gráfica.

Resolución:

Para obtener y(n) = x(-n+2) debemos realizar una inversión de la señal original seguida de un desplazamiento. A continuación se da un script para realizar esto:

```
% Reflexión de señales

x = [-5 -4 -3 -2 -1 0 1 2 3 4 5 ];
k=-2;

n=-5:5;
figure(1);

stem(n,x,'r');
hold;

n = -n;
n = n-k;

stem(n,x,'b');
xlabel('Señal original x[n](rojo), Señal Reflejada y Desplazada y[n-2]
(azul)');
```



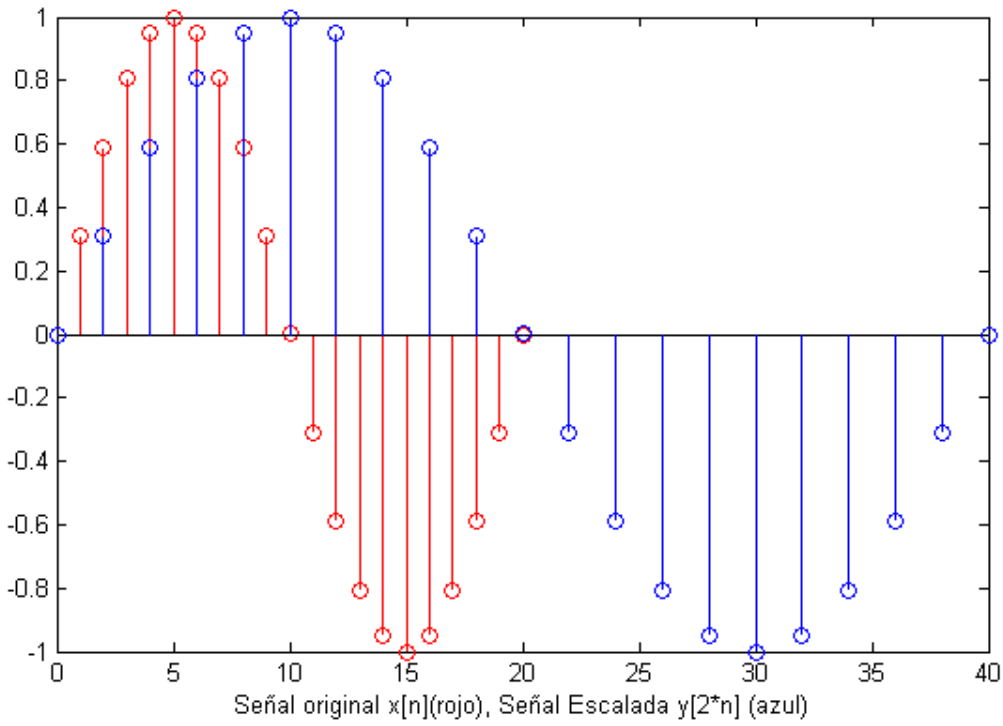
2.4.3 Escalado temporal

Introducción:
Esta operación consiste en reemplazar la variable independiente n por an, siendo a un entero. Esta operación también suele llamarse submuestreo.
Dada una señal senoidal x(n) de 50 Hz muestreada con una frecuencia de muestreo de 1000 muestras/seg., grafique en una misma figura las señales x(n) y y(n) = x(an) siendo a = 2.

Resolución:

El siguiente script grafica una señal senoidal y una versión escalada de la misma:

```
% Escalado de Frecuencia (submuestreo)

A = 1;       % Amplitud
f0 = 50;     % Frecuencia de la Señal
fs = 1000;   % Frec de muestreo en Hz
phi = 0;     % pi/2; % Fase
N = fs/f0 + 1;     % Nueva cantidad de muestras para mantener la escala
a = 2;       % Factor de escalado
n=0:N-1;

figure(1);

x=A*sin(2*pi*(f0/fs)*n+phi); % Señal senoidal
stem(n,x,'r');
hold;

n=a*n;
stem(n,x,'b');

xlabel('Señal original x[n](rojo), Señal Escalada y[2*n] (azul)');
```



2.4.4 Suma, multiplicación y escalado de amplitud

2.4.4.1 Dada la siguiente señal,

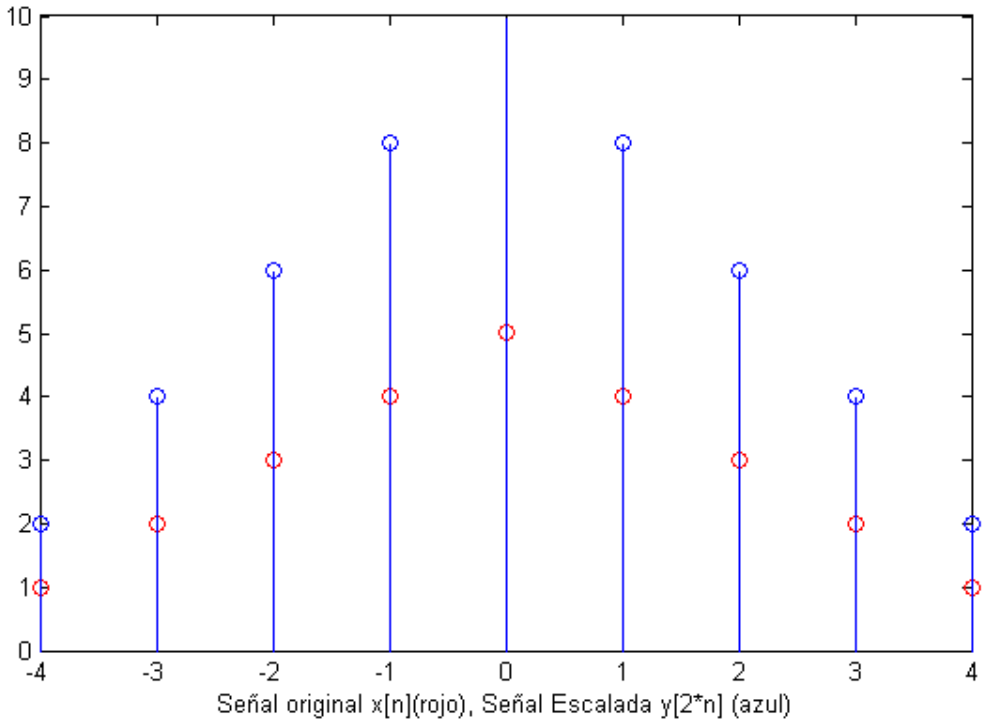
x[n] = [1 2 3 4 5 4 3 2 1]

Obtenga la señal y(n) = A x(n), siendo A = 2. Grafique x(n) e y(n) en una misma figura.

Resolución:

El siguiente script grafica una señal x[n] y luego la misma señal pero multiplicada por un factor de amplitud A:

```
% Escalado de Amplitud

x = [1 2 3 4 5 4 3 2 1];
n=-4:4;
A=2;

figure(1);

stem(n,x,'r');
hold;

y = A*x;

stem(n,y,'b');

xlabel('Señal original x[n](rojo), Señal Escalada y[2*n] (azul)');
```



2.4.4.2 Dadas las siguientes señales,

x1[n] = [0 1 1 1 1 0]

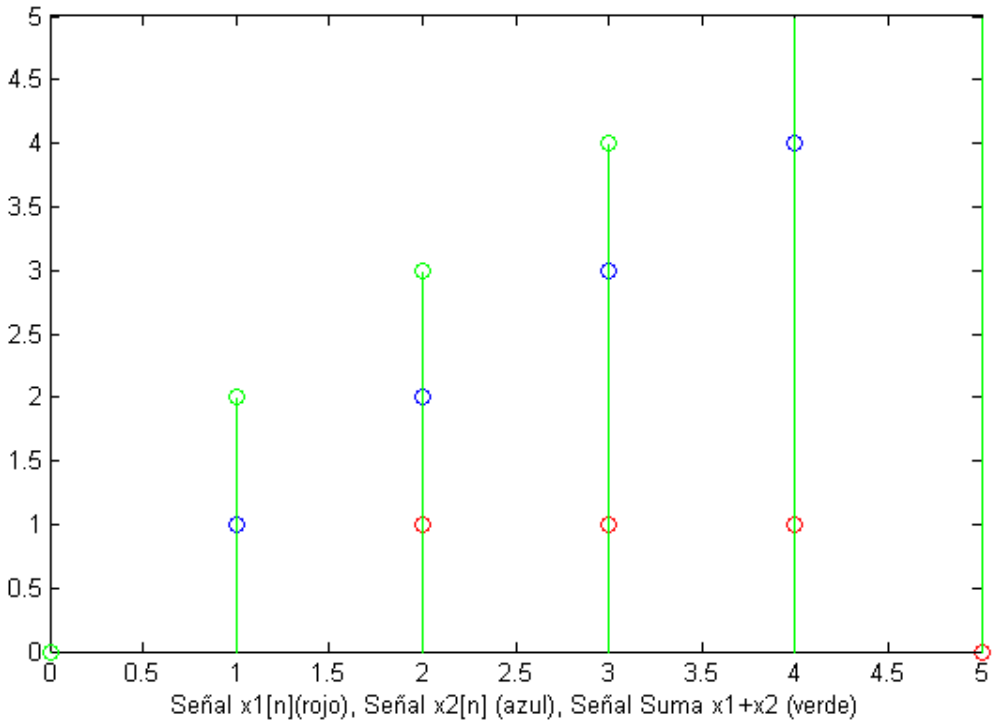
x2[n] = [0 1 2 3 4 5]

Obtenga la señal y(n) = x1(n)+x2(n). Grafique x1(n), x2(n) e y(n) en una misma figura.

Resolución:

El siguiente script grafica la señal x1[n], otra x2[n] y luego la suma de ambas y[n]:

```
% Suma de Señales discretas

x1 = [0 1 1 1 1 0 ];
x2 = [0 1 2 3 4 5 ];
n=0:5;

figure(1);

stem(n,x1,'r');
hold on;
stem(n,x2,'b');
hold on;

y = x1+x2;

stem(n,y,'g');

xlabel('Señal x1[n](rojo), Señal x2[n] (azul), Señal Suma x1+x2 (verde)');
```



2.4.4.3 Dadas las siguientes señales,

x1[n] = [1 2 3 4 5 6]

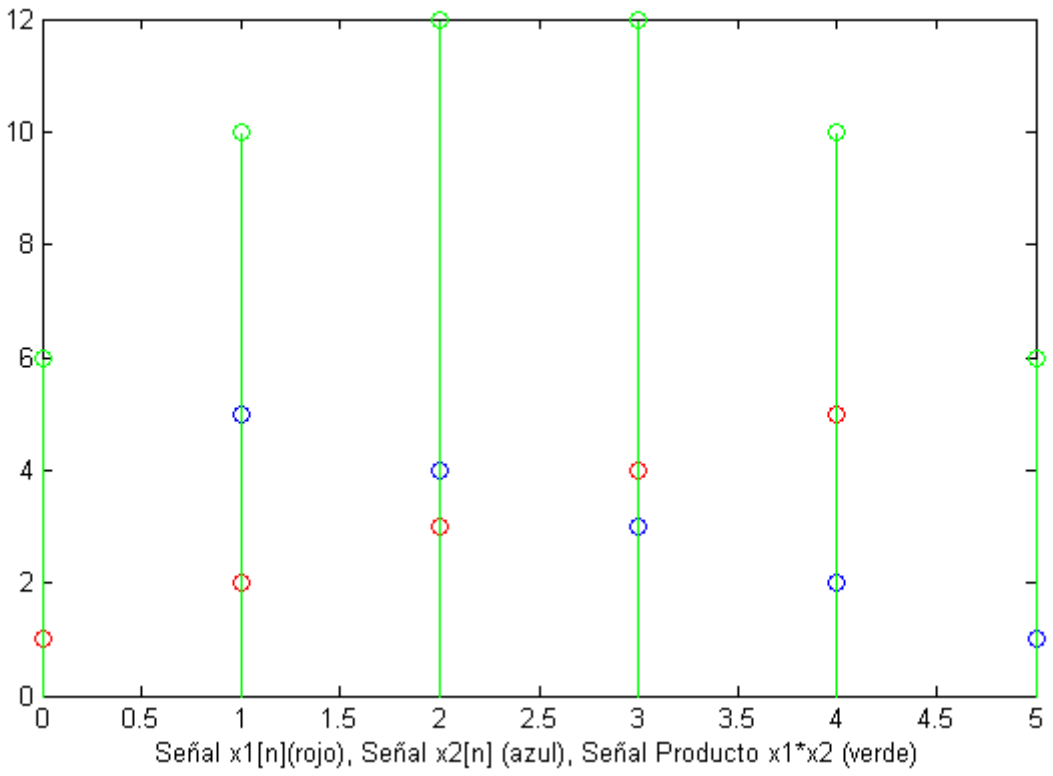
x2[n] = [6 5 4 3 2 1]

Obtenga la señal y(n) = x1(n)*x2(n). Grafique x1(n), x2(n) e y(n) en una misma figura.

Resolución:

El siguiente script grafica señal x1[n], otra x2[n] y luego el producto de ambas y[n]:

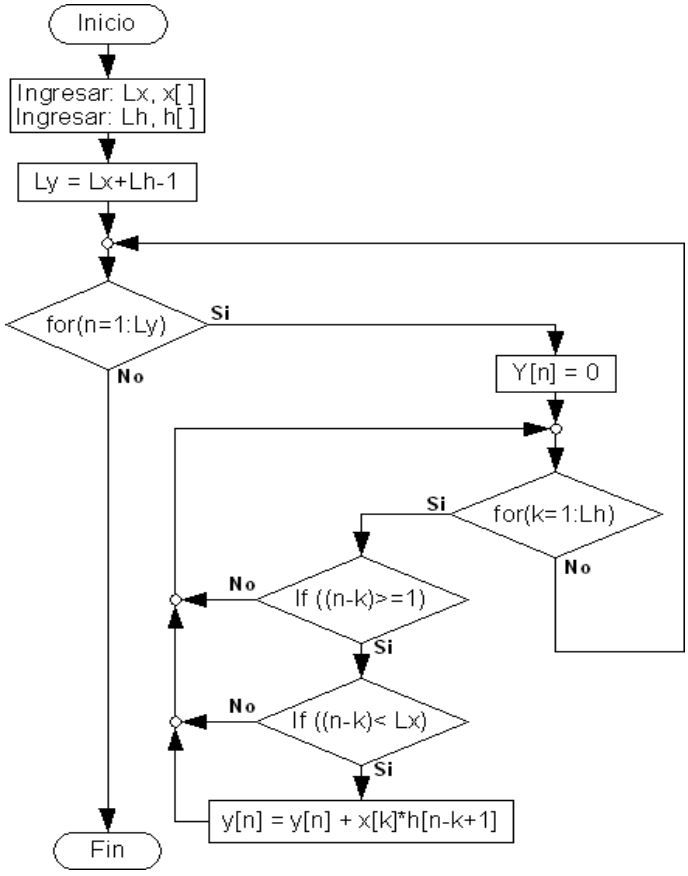
```
% Multiplicación de Señales discretas

x1 = [1 2 3 4 5 6];
x2 = [6 5 4 3 2 1];

n=0:5;

figure(1);

stem(n,x1,'r');
hold on;
stem(n,x2,'b');
hold on;

y = x1.*x2;

stem(n,y,'g');

xlabel('Señal x1[n](rojo), Señal x2[n] (azul), Señal Producto x1*x2
(verde)');
```



2.5- Convolución

2.5.1 Realice un diagrama de flujo del algoritmo de convolución.

Resolución: Diagrama de flujo del algoritmo de convolución:

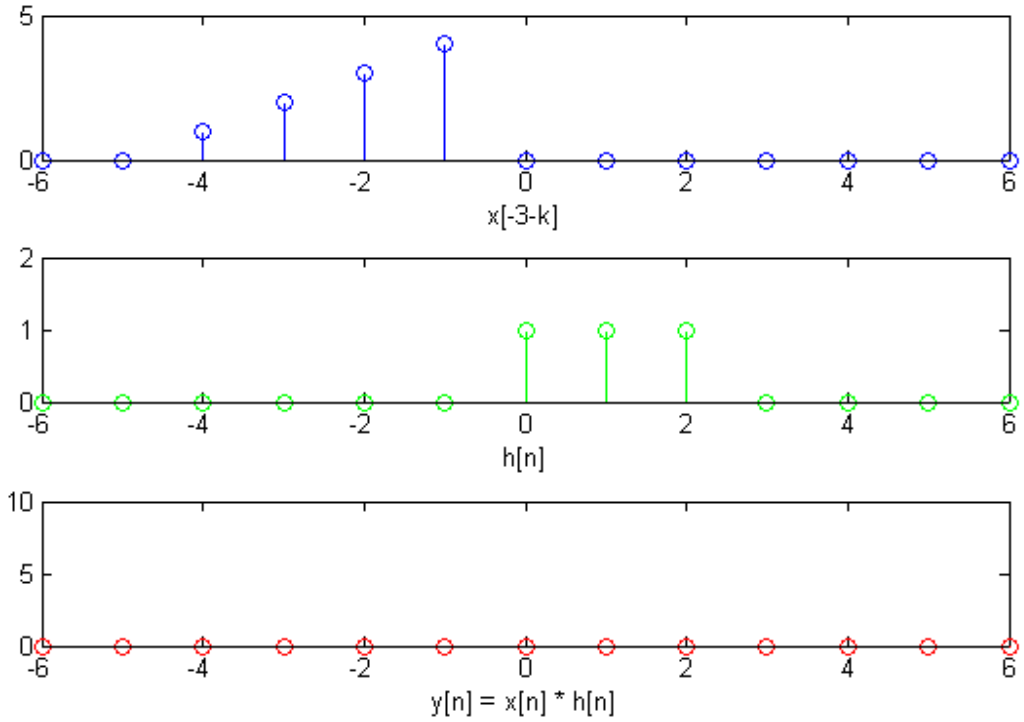


2.5.2 Calcule y grafique manualmente la convolución entre la señal x(n) y respuesta al impulso de un sistema h(n):

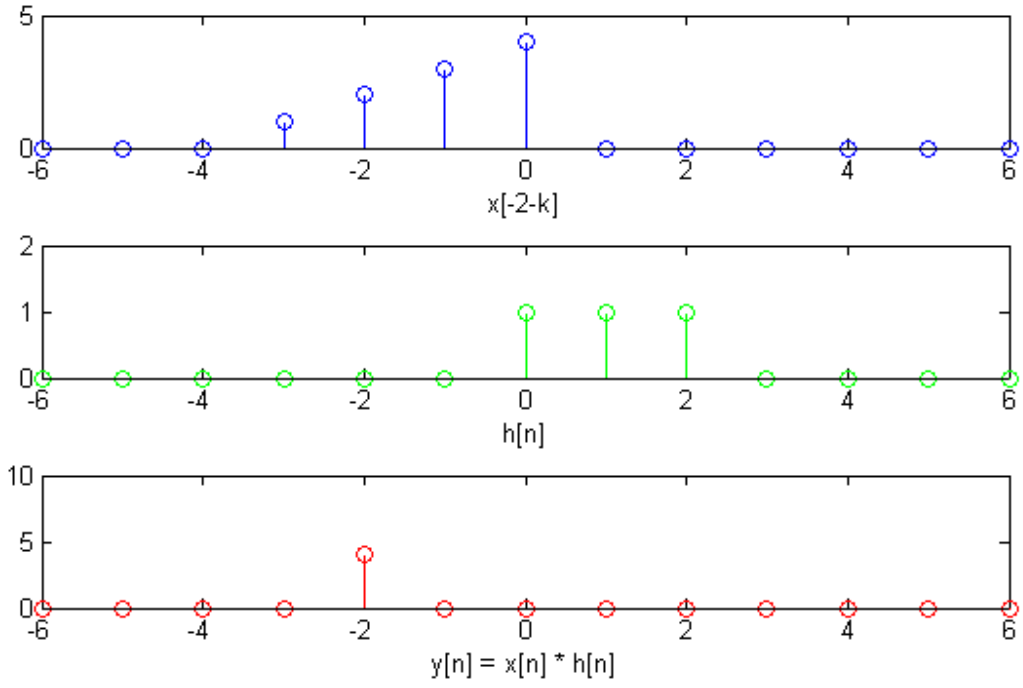
x[n] = {4, 3, **2**, 1, 0}
h[n]= {0, 0, **1**, 1, 1}

Resolución:

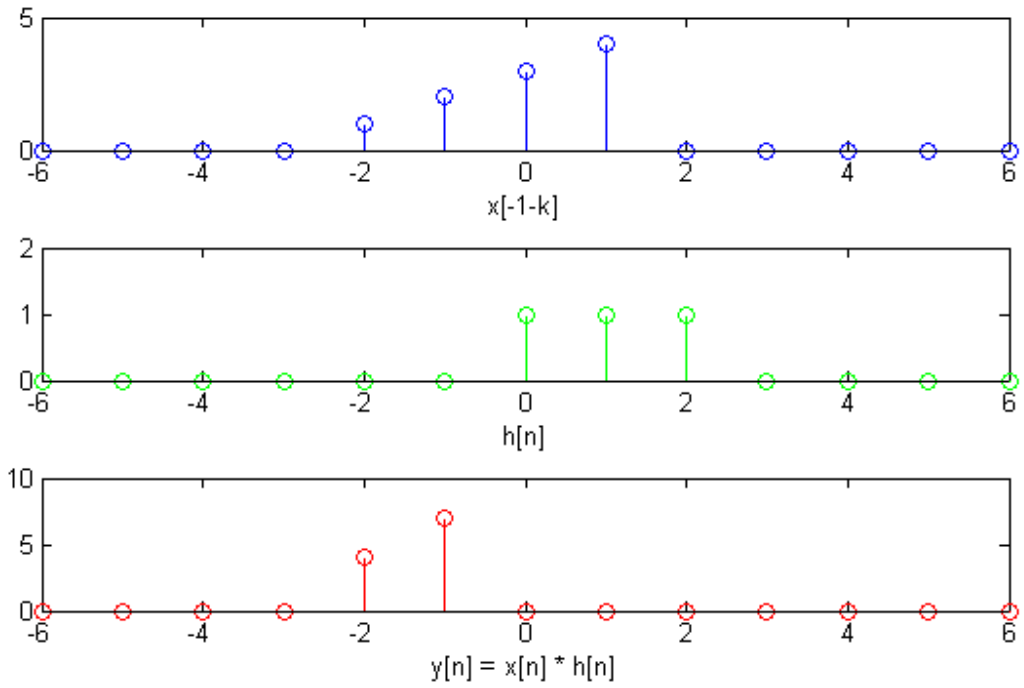
Para n=-3:



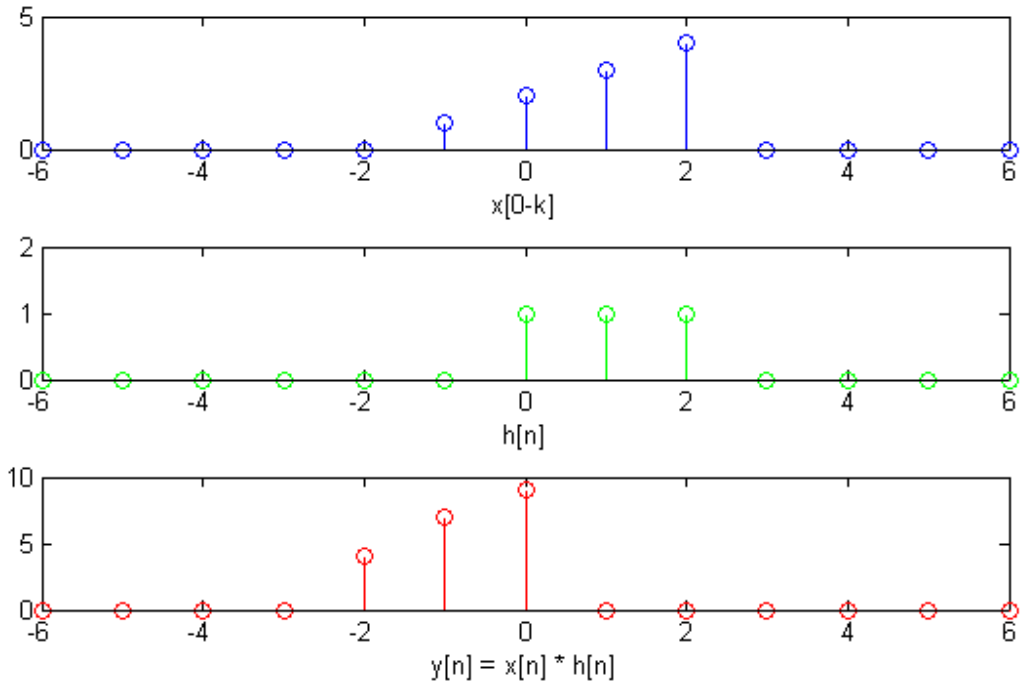
Para n=-2:



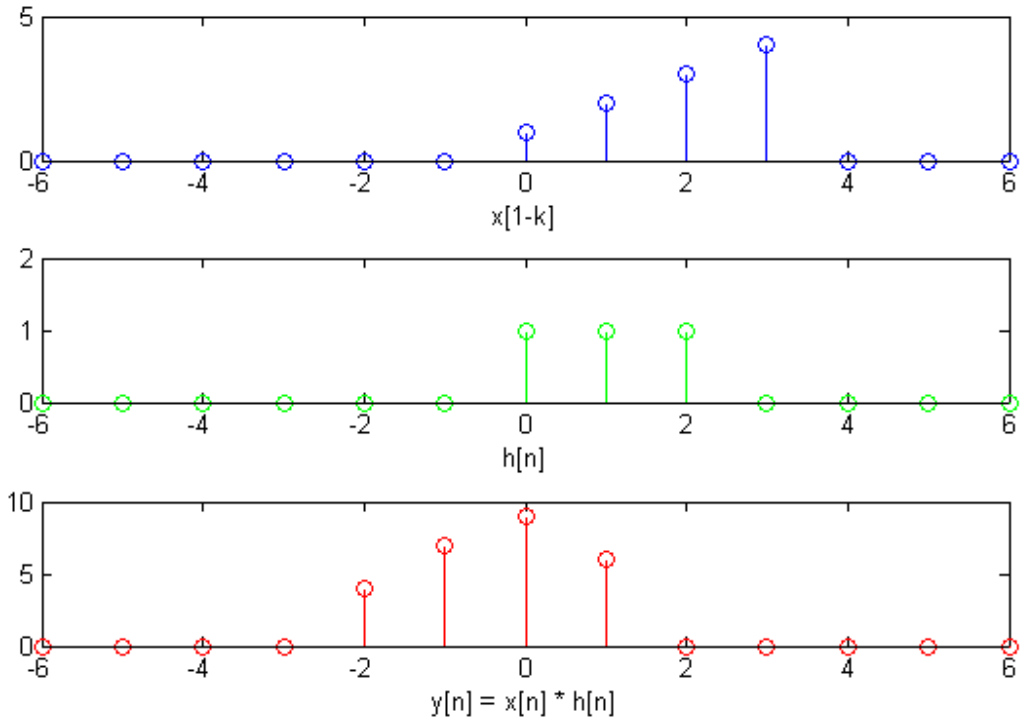
Para n=-1:



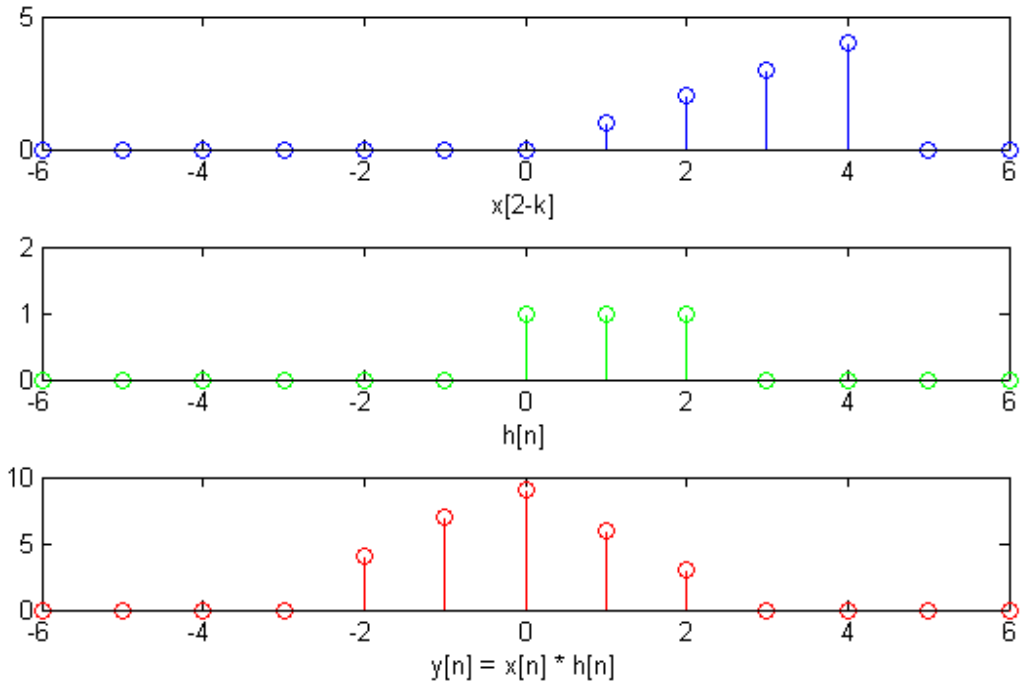
Para n=0:



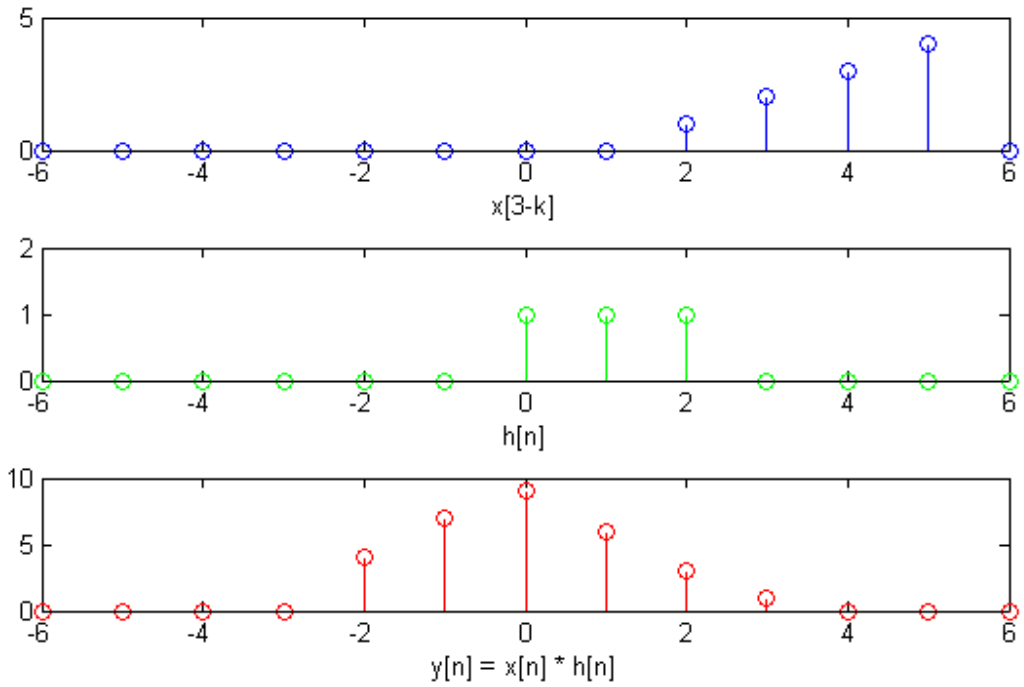
Para n=1:



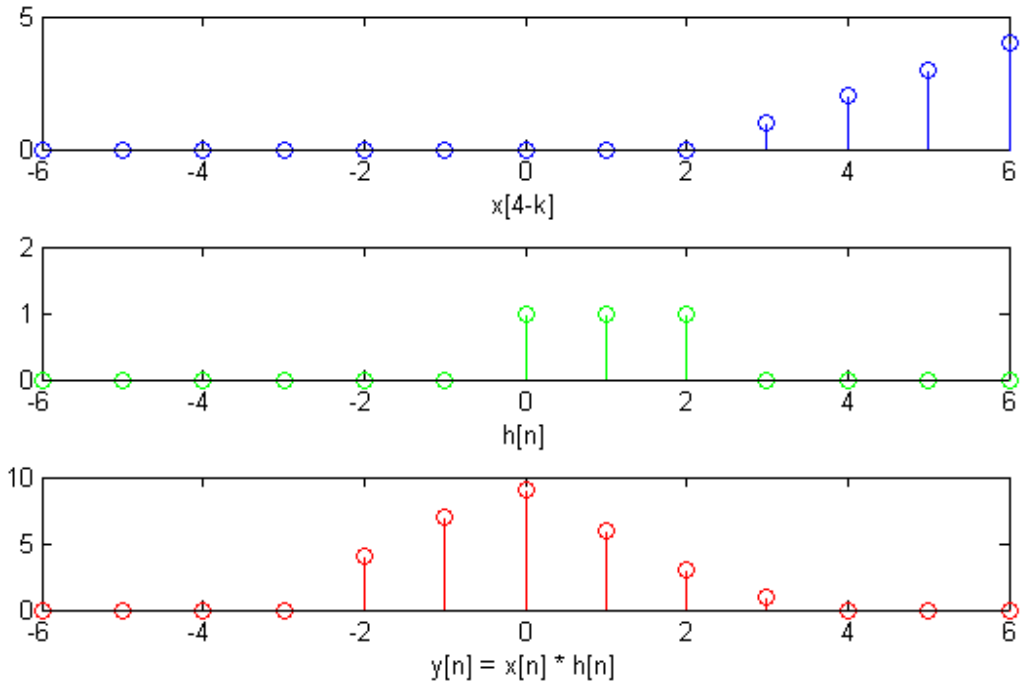
Para n=2:



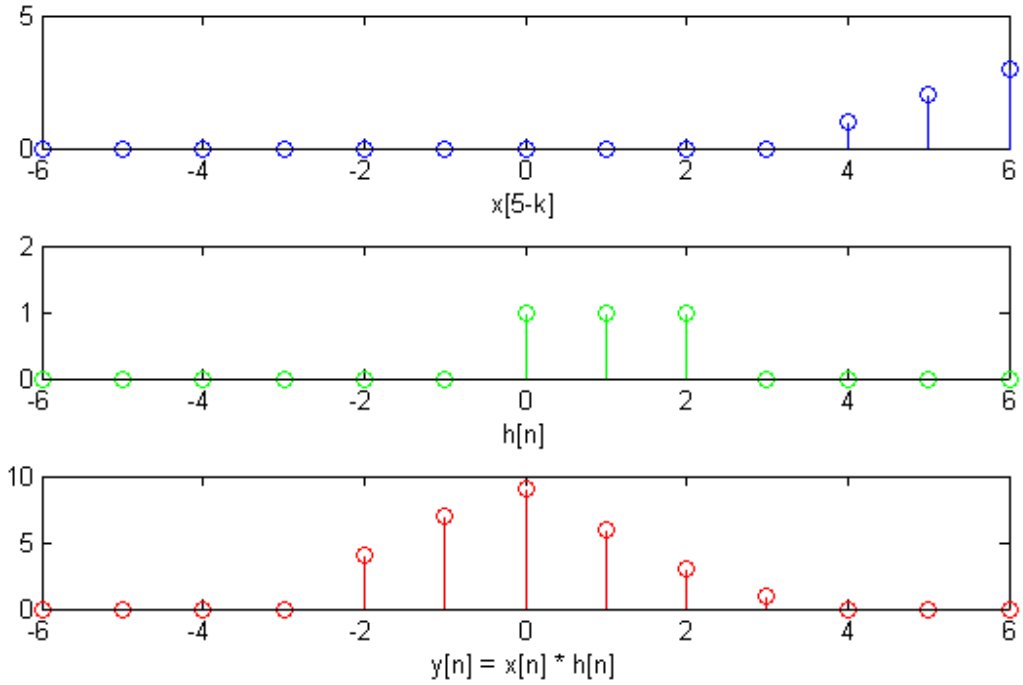
Para n=3:



Para n=4:



Para n=5:



2.5.3 Codifique el algoritmo realizado en el punto 2.5.1 mediante Lenguaje C o el lenguaje script de Matlab. Verifique los resultados del punto 2.5.2.

Resolución:

El siguiente script en Matlab realiza la convolución entre las señales x[n] y h[n] y almacena el resultado en y[n]:

```
% Convolución hecha manualmente

Lx = input('Ingrese numero de Muestras X=');
x=zeros(1,Lx);

x = input('Ingrese las Muestras de X( entre [] separadas por comas)=');

Lh = input('Ingrese numero de Muestras H=');
h=zeros(1,Lh);

h = input('Ingrese las Muestras de H( entre [] separadas por comas)=');

y = zeros(1,Lx+Lh-1);

for n=1:Lx+Lh-1;
    y(n)=0;
    for k=1:Lh
        if(n-k)>=0
            if(n-k)<Lx
                y(n)=y(n)+(h(k)*x(n-k+1));
            end
        end
```

```
    end
end

% La calculo con la funcion de MAtlab

y1=conv(x,h);

n=1:Lx;
figure(1);
subplot(2,2,1);
stem(n,x);
title('X(n)');
n=1:Lh;
subplot(2,2,2);
stem(n,h);
title('H(n)');
n=1:Lh+Lx-1;
subplot(2,2,3);
stem(n,y);
title('Y(n)=X(n)*H(n) calculada');
subplot(2,2,3);
stem(n,y);
title('Y(n)=X(n)*H(n) calculada');
subplot(2,2,4);
stem(n,y1);
title('Y1(n)=X(n)*H(n) por conv');
```

2.5.4 Verifique los resultados del punto 2.5.2 mediante la función conv de Matlab.

Resolución:

Los resultados obtenidos fueron los mismos que los del punto anterior.

2.5.5 Demuestre las siguientes propiedades de la convolución:

Ley Conmutativa: x(n) * h(n) = h(n) * x(n)

Ley Asociativa: [x(n) * h1(n)] * h2(n) = x(n) * [h1(n) * h2(n)]

Ley Distributiva: x(n) * [h1(n) + h2(n)] = x(n) * h1(n) + x(n) * h2(n)

Siendo:

x[n] = [1 2 3 4 3 2 1]

h[n] = h1[n] = [1 1 1]

h2[n] = [1 2 1]

Resolución:

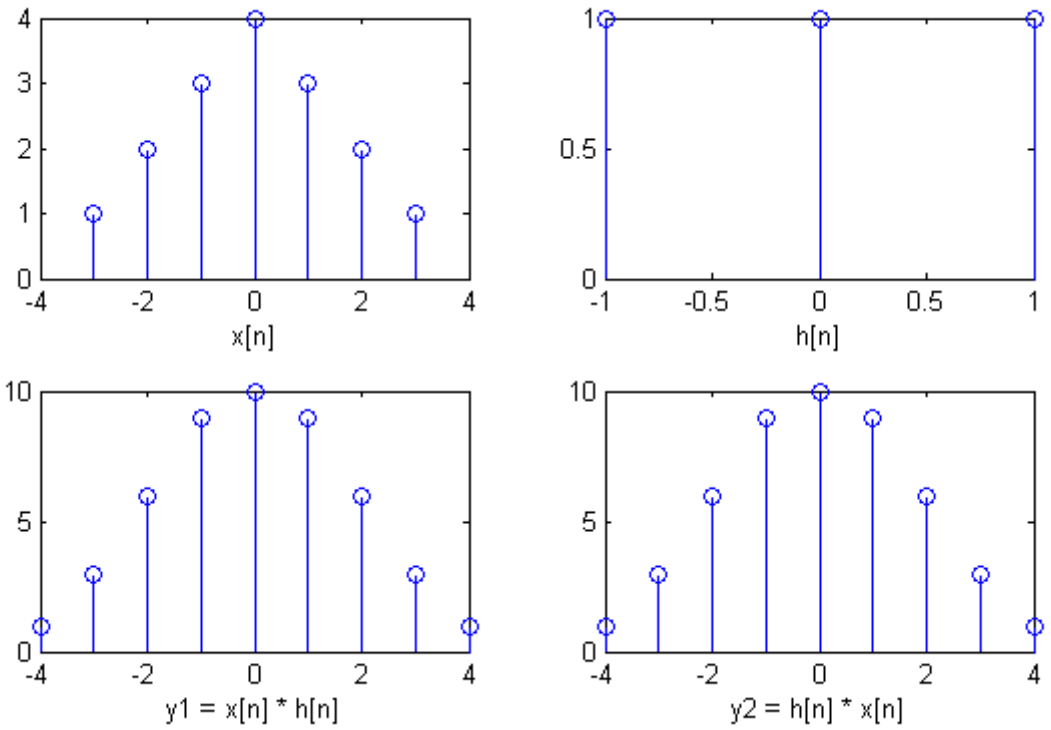
El siguiente script demuestra las tres propiedades mediante un ejemplo:

```
% Demostración de propiedades

x = [1 2 3 4 3 2 1];
```

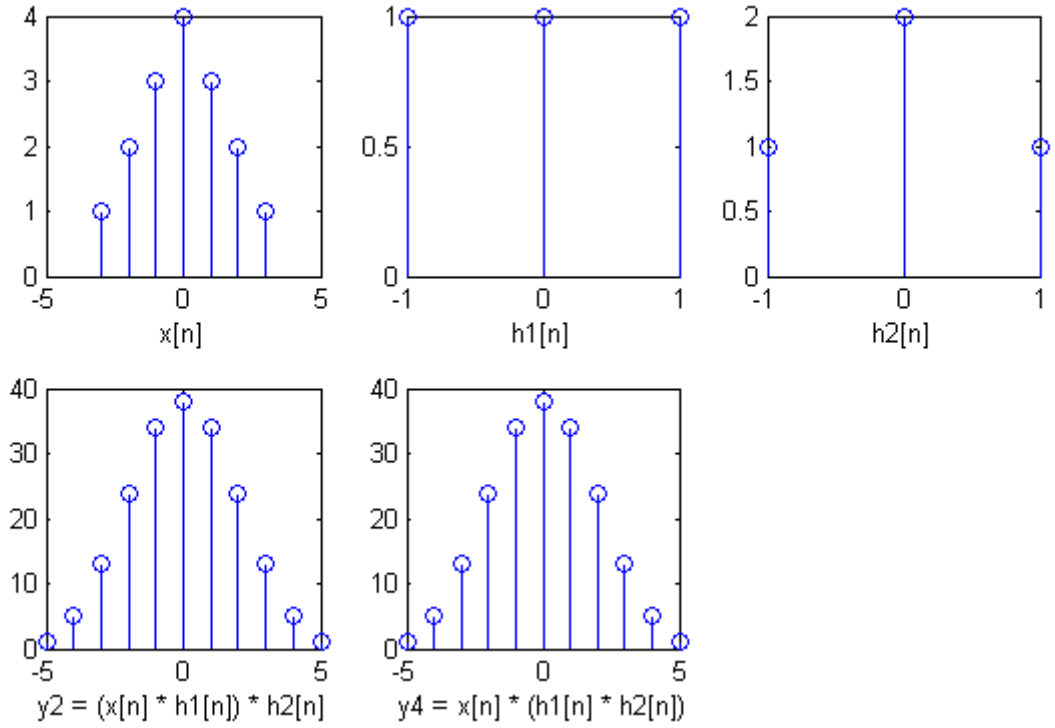
```
h1 = [1 1 1];

h2 = [1 2 1];

n=-4:4;
n1=-3:3;
n2=-1:1;
n3=-5:5;
k=-20:20;

%Propiedad Conmutativa
y1=conv(x,h1);
y2=conv(h1,x);

figure(1);
subplot(2,2,1);
stem(n1,x);
xlabel('x[n]');
subplot(2,2,2);
stem(n2,h1);
xlabel('h[n]');
subplot(2,2,3);
stem(n,y1);
xlabel('y1 = x[n] * h[n]');
subplot(2,2,4);
stem(n,y2);
xlabel('y2 = h[n] * x[n]');

%Propiedad Asociativa
y1=conv(x,h1);
y2=conv(y1,h2);

y3=conv(h1,h2);
y4=conv(x,y3);

figure(2);
subplot(2,3,1);
stem(n1,x);
xlabel('x[n]');
subplot(2,3,2);
stem(n2,h1);
xlabel('h1[n]');
subplot(2,3,3);
stem(n2,h2);
xlabel('h2[n]');
subplot(2,3,4);
stem(n3,y2);
xlabel('y2 = (x[n] * h1[n]) * h2[n]');
subplot(2,3,5);
stem(n3,y4);
xlabel('y4 = x[n] * (h1[n] * h2[n])');

%Propiedad Distributiva
y1=h1 + h2;
y2=conv(x,y1);

y3=conv(x,h1);
y4=conv(x,h2);
y5= y3 + y4;
```

```
figure(3);
subplot(2,3,1);
stem(n1,x);
xlabel('x[n]');
subplot(2,3,2);
stem(n2,h1);
xlabel('h1[n]');
subplot(2,3,3);
stem(n2,h2);
xlabel('h2[n]');
subplot(2,3,4);
stem(n,y2);
xlabel('y2 = x[n] * (h1[n] + h2[n])');
subplot(2,3,5);
stem(n,y5);
xlabel('y5 = x[n] * h1[n] + x[n] * h2[n]');
```

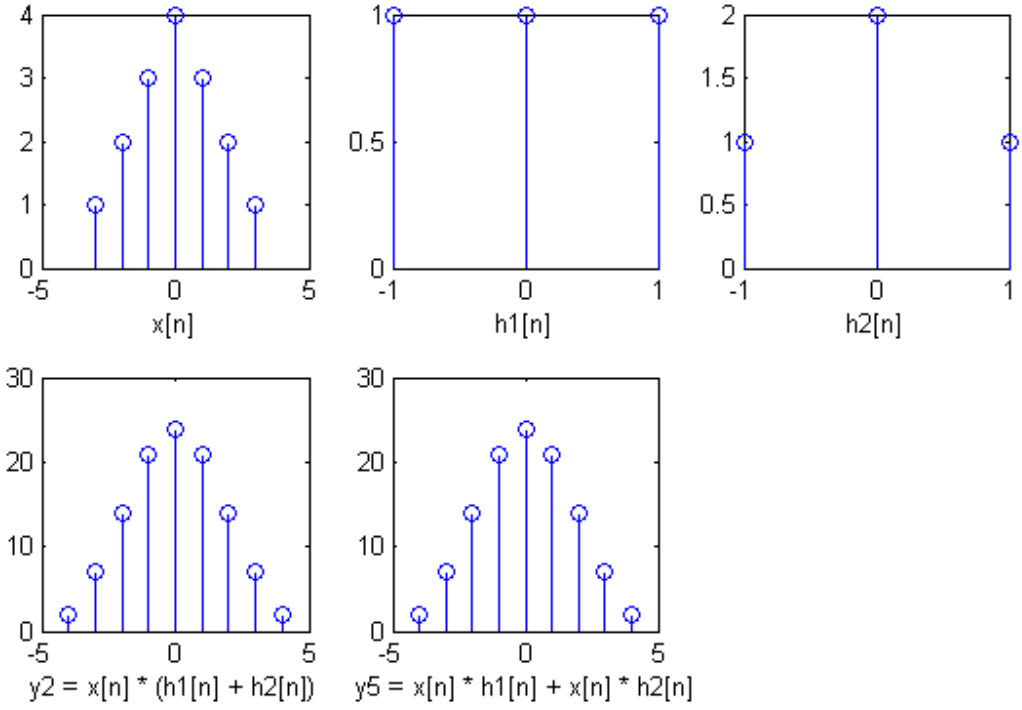
Propiedad Conmutativa:



Propiedad Asociativa:



Propiedad Distributiva:



2.5.6 Si un sistema tiene una respuesta al impulso h(n) y salida y(n), determine mediante la deconvolución, función deconv de Matlab, la entrada x(n), siendo:

h[n] = [1 2 3]

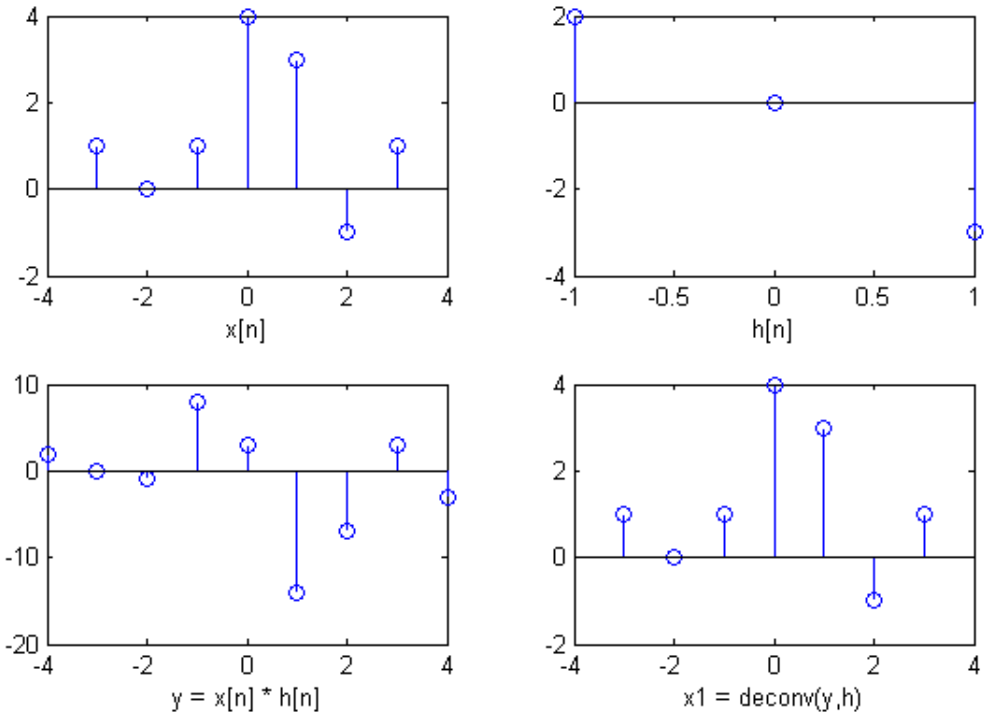
y[n] = [4 13 28 27 18]

Antes de comenzar analice la función deconv.

Resolución:

El siguiente script deconvoluciona la respuesta al impulso de un sistema con su salida, para determinar cual fue la entrada:

```
% Convolucion-Deconvolucion

x = [1 0 1 4 3 -1 1];

h = [2 0 -3];

n=-4:4;
n1=-3:3;
n2=-1:1;
n3=-5:5;
k=-20:20;

%Convolucion de x(excitación) con h(kernel)
y=conv(x,h);

%Deconvolucion de y(salida) con h(kernel) para obtener x(excitación)
[x1,r] = deconv(y,h);

figure(1);
subplot(2,2,1);
stem(n1,x);
xlabel('x[n]');
subplot(2,2,2);
stem(n2,h);
xlabel('h[n]');
subplot(2,2,3);
stem(n,y);
xlabel('y = x[n] * h[n]');
subplot(2,2,4);
stem(n1,x1);
xlabel('x1 = deconv(y,h)');
```



2.5.7 Genere 10 ciclos de una señal senoidal de frecuencia fo=100 Hz de amplitud 10, muestreada a una frecuencia fs = 10000Hz. Luego genere ruido de amplitud ±4, mediante la función rand y súmelo a la señal senoidal.
Ahora convolucione la señal senoidal con ruido con la siguiente respuesta al impulso h(n):

h[n] = [0.0040 0.0045 0.0060 0.0083 0.0115 0.0155 0.0202 0.0253 0.0306 0.0361 0.0414 0.0463 0.0507 0.0543 0.0570 0.0586 0.0592 0.0586 0.0570 0.0543 0.0507 0.0463 0.0414 0.0361 0.0306 0.0253 0.0202 0.0155 0.0115 0.0083 0.0060 0.0045 0.0040];

Utilice la función conv de MATLAB.
Grafique la señal senoidal con ruido y la señal obtenida como resultado de la convolución en una misma figura con las mismas escalas de amplitud en ambos gráficos para poder comparar.

¿Que sucede con el ruido en la señal obtenida?

Obtenga una conclusión con la ayuda de la función freqz de Matlab para grafícar la respuesta
en frecuencia del sistema h[n].

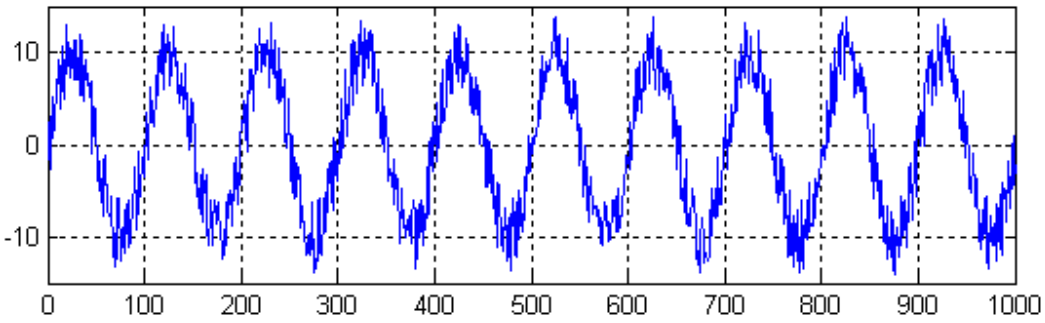
El ruido es atenuado por que lo convolucionamos con un filtro pasabajos.

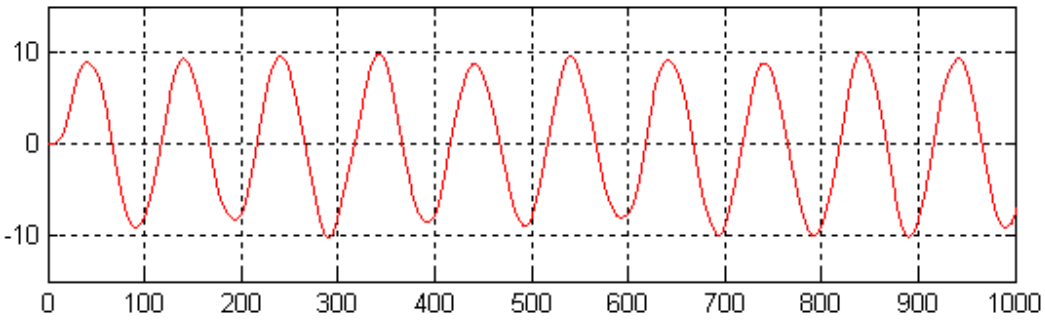
¿Cual es la longitud de la señal obtenida?

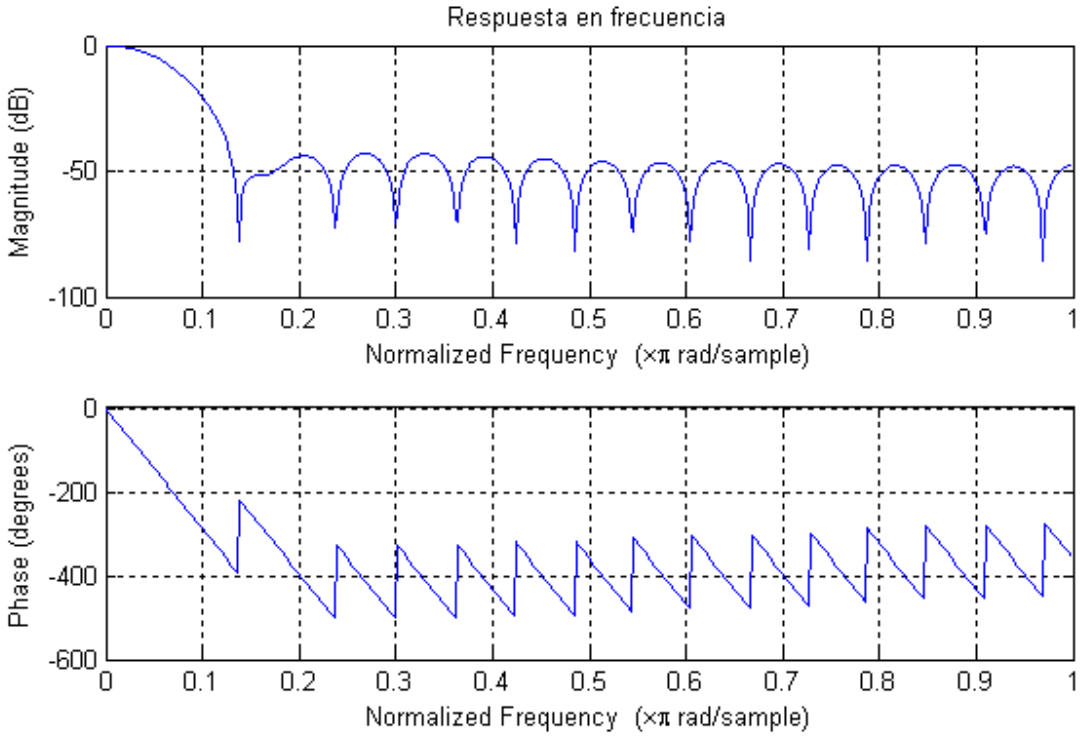
La longitud de la señal obtenida es L convolución = Lh+LS-1 = 33+1000-1=1032 muestras.

Resolución:

```
A = 10;       % Amplitud
f0 = 100;    % Frecuencia de la senoide
fs = 10000; % Frec de muestreo en Hz
phi = 0;     % pi/2; % Fase
N = 1000;    % Cantidad de muestras para 10 ciclos de la señal

%Respuesta al impulso
h = [0.0040  0.0045  0.0060  0.0083  0.0115  0.0155  0.0202  0.0253
0.0306  0.0361  0.0414  0.0463  0.0507  0.0543  0.0570  0.0586  0.0592
0.0586  0.0570  0.0543  0.0507  0.0463  0.0414  0.0361  0.0306  0.0253
0.0202 0.0155   0.0115   0.0083   0.0060   0.0045  0.0040];

n=0:N-1;

seno=A*sin(2*pi*(f0/fs)*n+phi); % Señal senoidal
ruido=4*(-1).^round(10*rand(1,N)).*rand(1,N);
senoruido = seno + ruido;

convolucion = conv(senoruido,h);

figure(1);
subplot(2,1,1), plot(senoruido,'b'), grid on, axis([0 N -15 15]);
subplot(2,1,2), plot(convolucion,'r'), grid on, axis([0 N -15 15]);

[p,W]=freqz(convolucion,senoruido);

figure(2);
subplot(4,1,4);
freqz(h),title('Respuesta en frecuencia');
```







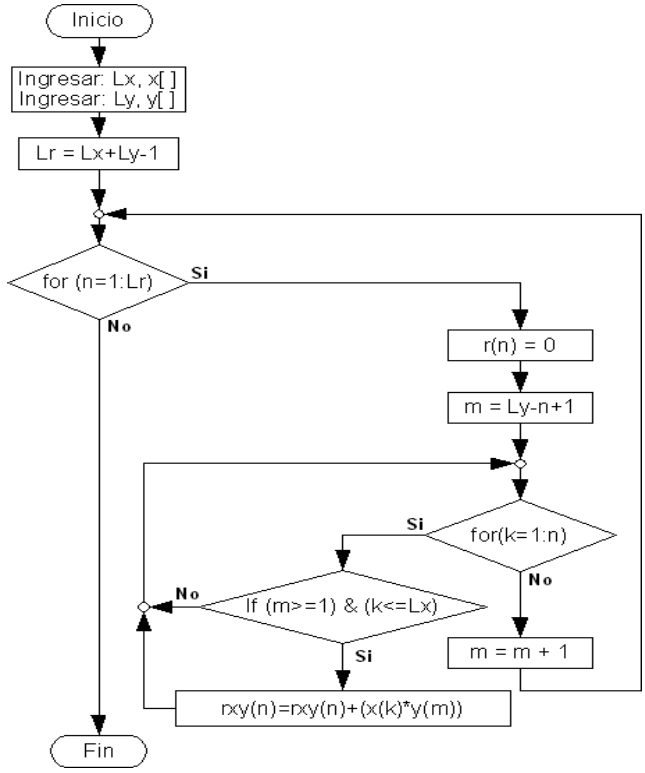
2.6- Correlación

Objetivo: comprender el algoritmo de correlación de señales.

Introducción: la correlación es una operación matemática similar a la convolución. La correlación utiliza dos señales de entrada para producir una tercera señal denominada correlación cruzada. Si una señal es correlacionada consigo misma el resultado es una señal denominada autocorrelación. Una aplicación típica es la detección de una forma de onda conocida en una señal con ruido.

2.6.1 Realice un diagrama de flujo del algoritmo de correlación, luego codifique dicho algoritmo mediante lenguaje C o el lenguaje script de Matlab.

Resolución: diagrama de flujo del algoritmo de correlación:



El diagrama de flujo anterior implementado en lenguaje script de MatLab sería:

```
%{
    Correlacion Cruzada de 2 señales

    RXY[l] = Sum{X[n]Y[n-l]}  n=i N-|K|-1  para i=l K=0 l >=0  i=0 k=l l<0
    RXY[l] = Sum{X[n+l]Y[n]}

Ejemplo
X = [0,1,-1,0,0,1,2,1,0,1,0] 15
Y = [1,2,1] 3
%}

% Ejemplo para verificar la tecnica

Lx = input('Ingrese numero de Muestras X=');

x=zeros(1,Lx);

x = input('Ingrese las Muestras de X( entre [] separadas por comas)=');

Ly = input('Ingrese numero de Muestras Y=');

y=zeros(1,Ly);

y = input('Ingrese las Muestras de Y( entre [] separadas por comas)=');

L = Lx+Ly-1;

rxy = zeros(1,L);

for n=1:L;
```

```
        m=Ly-n+1;
        for k=1:n
            if m>=1 & k<=Lx
              rxy(n)=rxy(n)+(x(k)*y(m));
            end
        m=m+1;
        end
    end

    % La calculo con la funcion de MAtlab

    rxy1=xcorr(x,y);

    figure(1);
    subplot(2,2,1);
    stem(x);
    title('X(n)');
    subplot(2,2,2);
    stem(y);
    title('Y(n)');
    subplot(2,2,3);
    stem(rxy);
    title('RXY(n) calculada');
    subplot(2,2,4);
    stem(rxy1);
    title('RXY1(n) por matlab');
```

2.6.2 Determine en forma manual, mediante gráficos la correlación cruzada de las siguientes señales:

X(n) = {0, 1, -1, 0, 0, 1, 2, 1, 0, 0, 0, 1, 0, -1, 0}

Y(n) = {1, 2, 1}

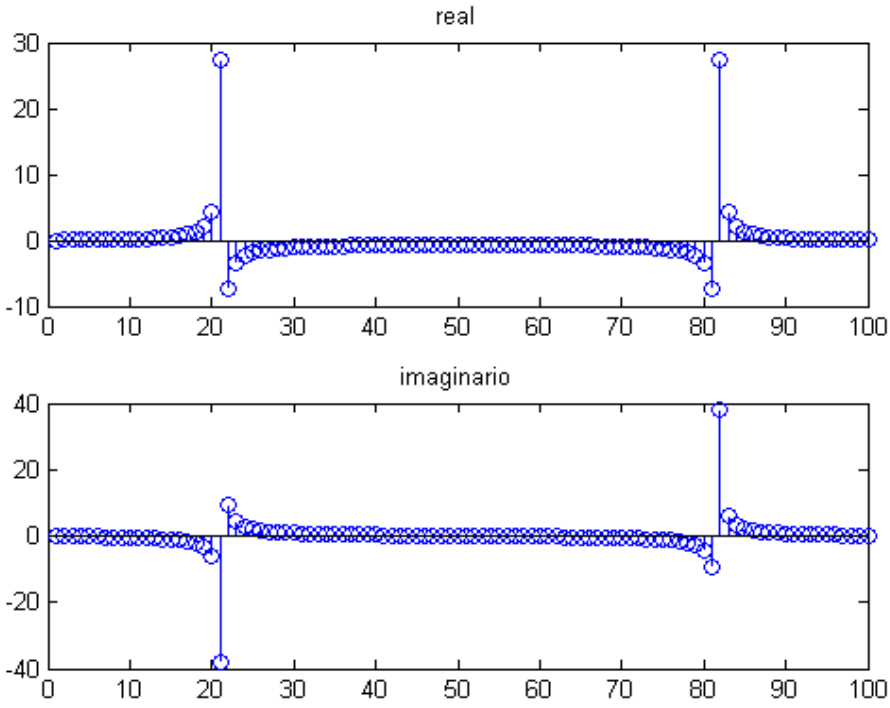
Luego aplique el algoritmo codificado en el punto 2.6.1 para verificar los resultados.

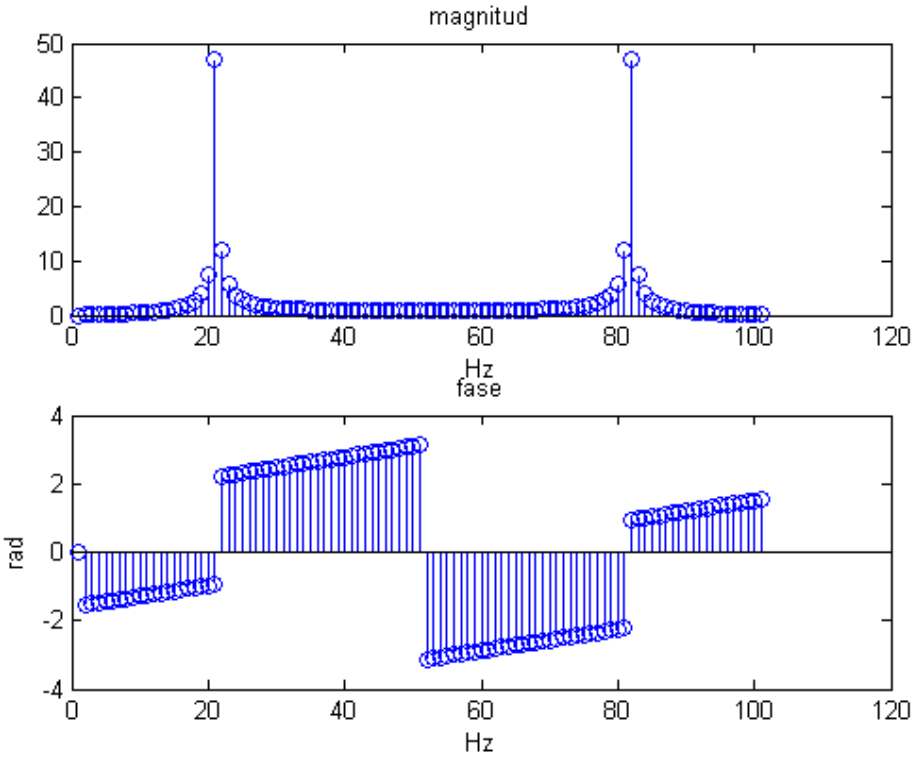
2.7- Muestreo en el dominio de la frecuencia - Transformada de Fourier Discreta

2.7.1 Analice la función fft de Matlab utilizada en el análisis frecuencial de señales en tiempo discreto.

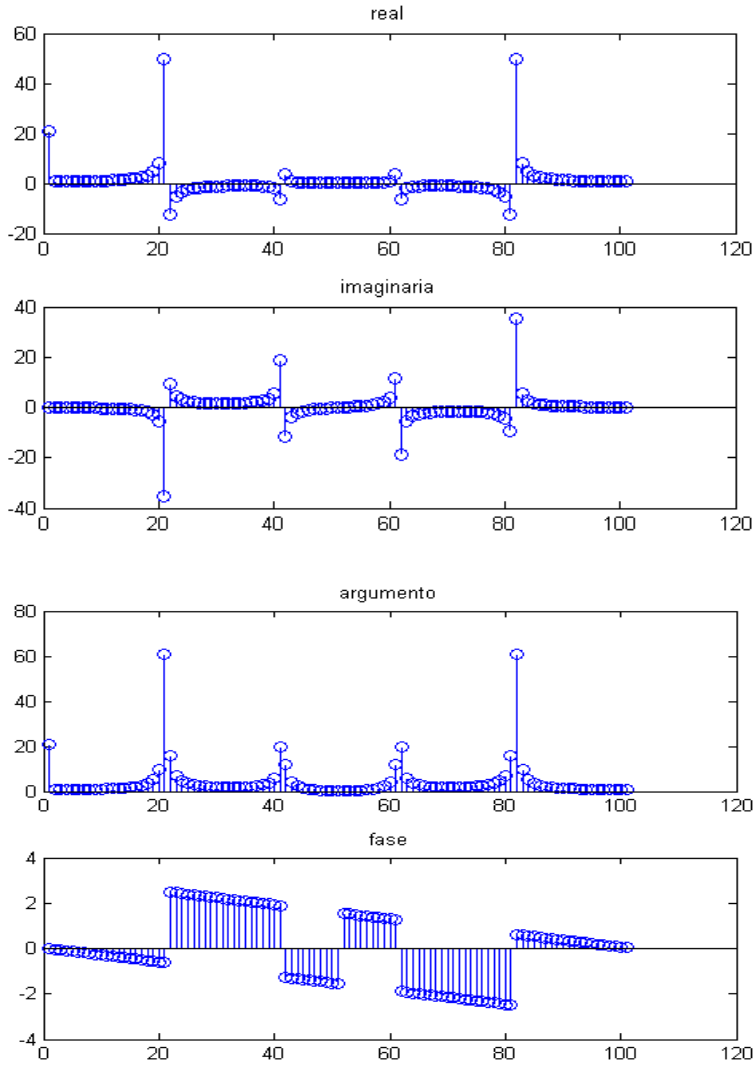
2.7.2 Obtenga, mediante la función fft el espectro de frecuencias de las señales senoidal (sin), triangular (tripuls) y cuadrada (square). Tomar como frecuencia de la señal de entrada fo=20Hz y frecuencia de muestreo fs=100Hz. Grafique mediante la función stem la parte real (función real()) e imaginaria (función imag()) del vector resultante. Utilice la función subplot para que ambos gráficos se realicen en la misma figura. Grafique a continuación la magnitud (función abs) y la fase (función angle) de la función en el dominio de la frecuencia. Exprese el eje de las abscisas en frecuencia en Hz, tenga en cuenta que para n=N tendremos la frecuencia de muestreo fs.

Resolución:

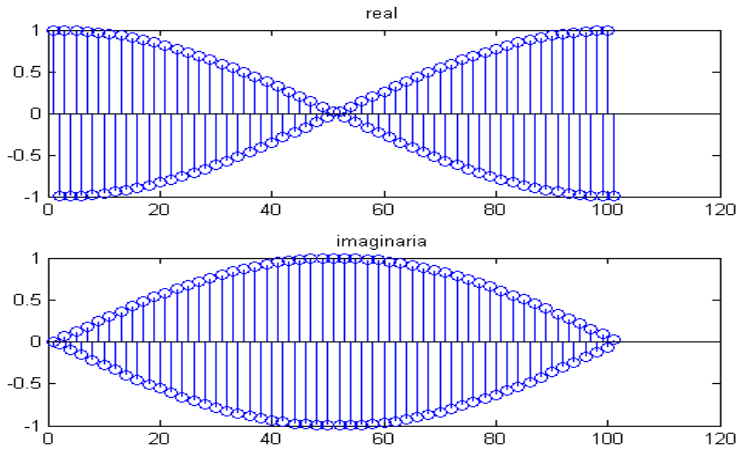

real
30
20
10
0
-10
0 10 20 30 40 50 60 70 80 90 100
imaginario
40
20
0
-20
-40
0 10 20 30 40 50 60 70 80 90 100

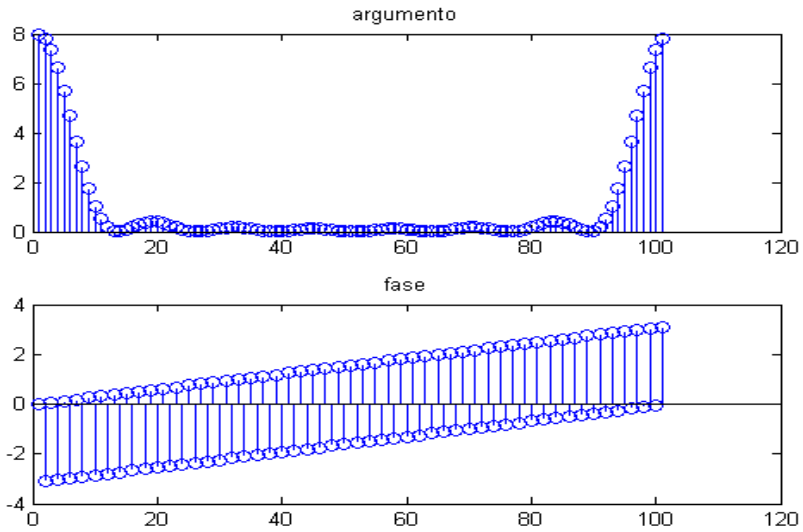

magnitud
50
40
30
20
10
0
0 20 40 60 80 100 120
Hz
fase
4
2
0
-2
-4
rad
0 20 40 60 80 100 120
Hz

Para la onda cuadrada:



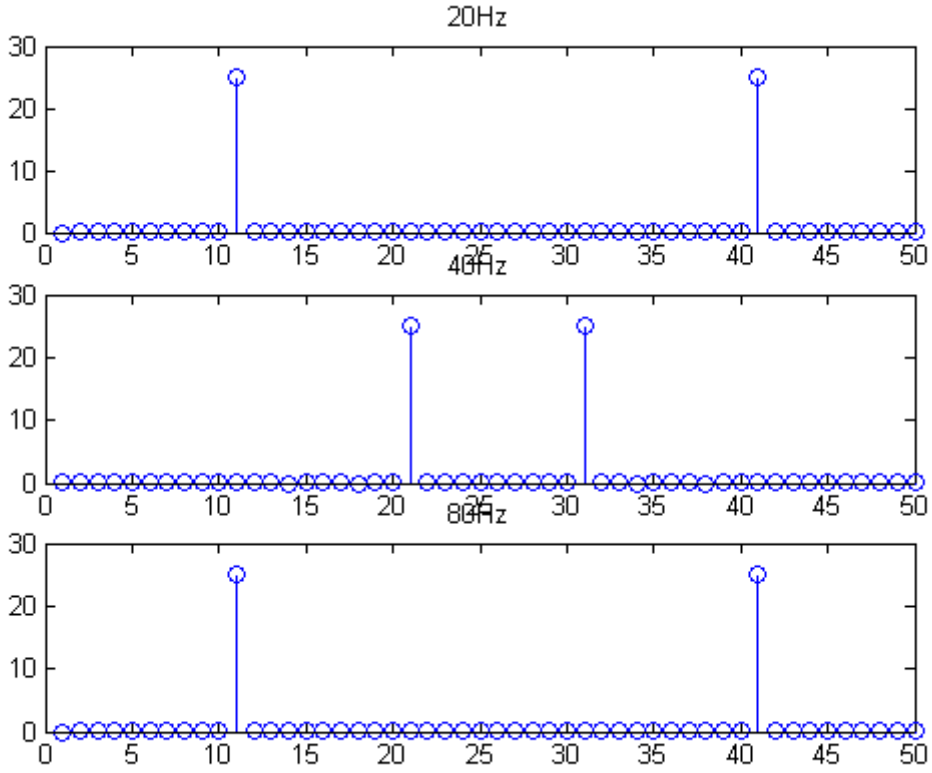
Para la señal triangular (notar que la señal es solo un pulso, no es periodica):



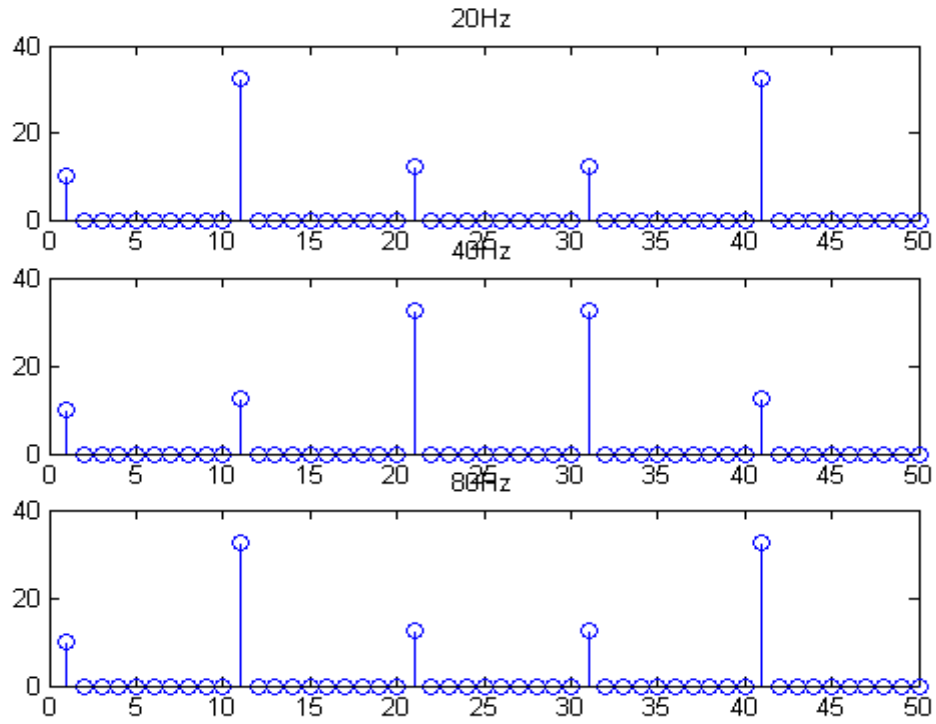


2.7.3 Varíe la frecuencia de la señal de entrada de modo que fo sea 20Hz, 40Hz y 80Hz, manteniendo fs=100Hz. Obtenga las transformadas de FOURIER para distintas frecuencias, a partir de que frecuencia comienza a aparecer el solapamiento?

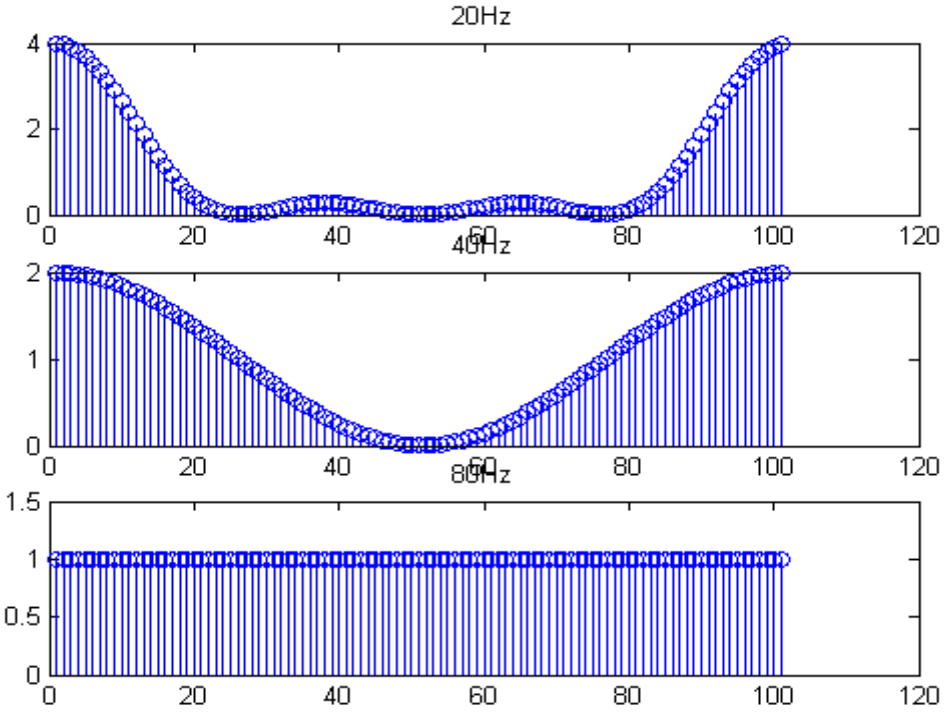
Señal senoidal:



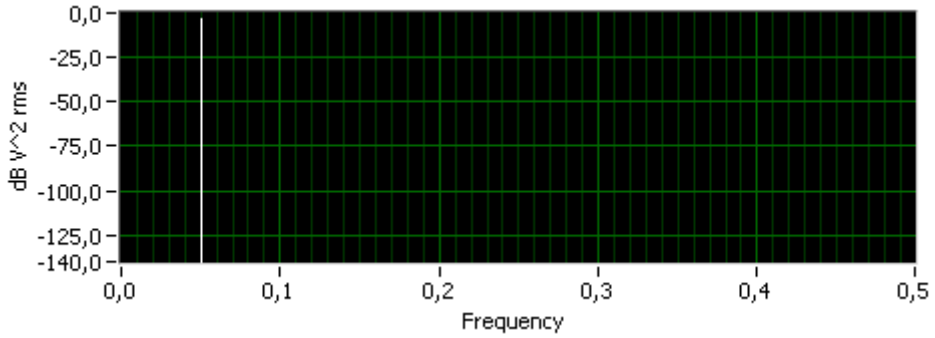
Señal cuadrada:

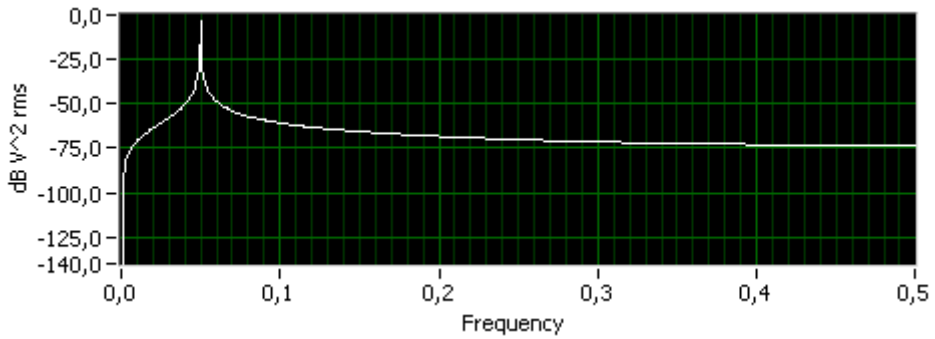


Señal triangular:



2.7.4 Desarrolle un instrumento virtual que permita realizar un análisis espectral de la señal inyectada. Mediante el generador desarrollado en el TP1 inyecte una señal senoidal y verifique que sucede al modificar el número de muestras de modo que el número de ciclos de la señal generada sea entero, luego pruebe generando una cantidad de muestras de modo que el número de ciclos de la señal generada no sea un entero. ¿Cómo se denomina el efecto causado?



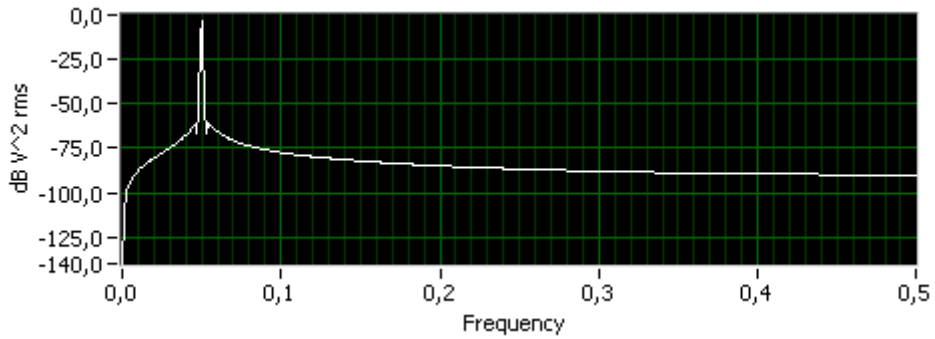


El efecto observado es el llamado "fuga espectral", es producido debido a que la señal no es periódica, porque esta muestreada, y cuando las muestras que tomamos no caen en un periodo entero se produce este efecto.

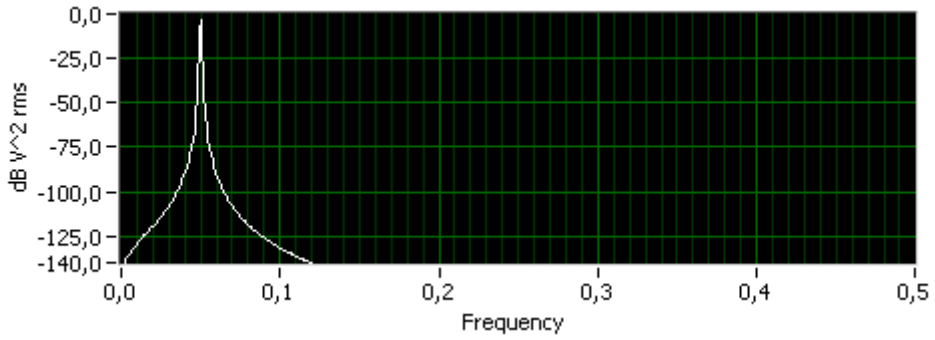
2.7.5 Modifique el instrumento virtual creado en el punto 2.7.4 para demostrar que el efecto de fuga o drenaje espectral (spectral leakage) es suavizado agregando distintas ventanas (Hanning, Hamming, Triangular, etc) antes de pasar la señal del dominio del tiempo al dominio de la frecuencia.

Los resultados obtenidos con ventanas :

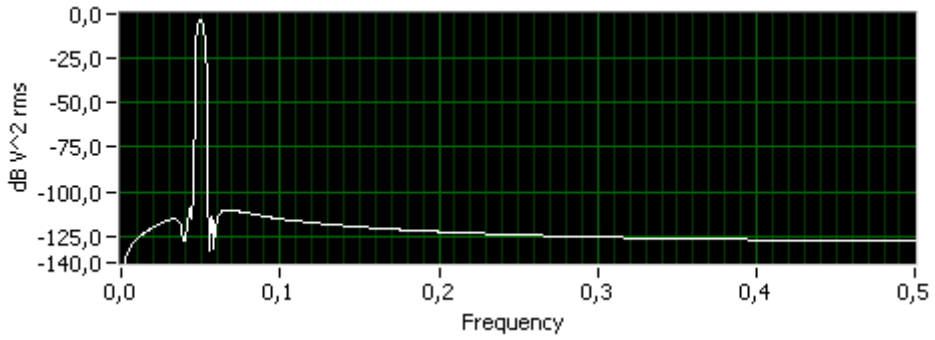
Hamming:



Hanning:

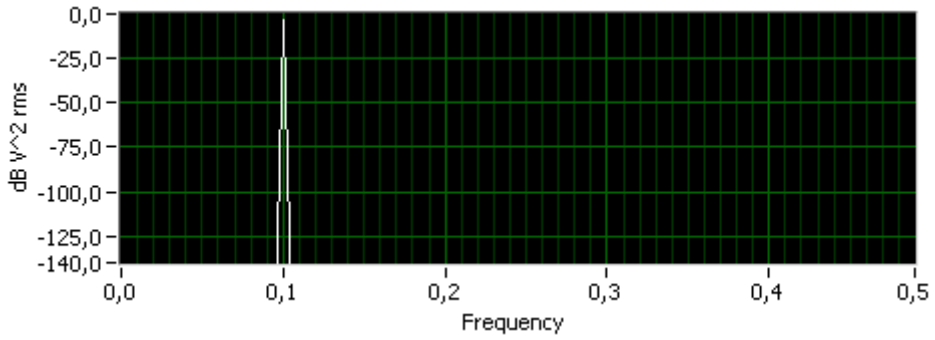


Flat-Top:

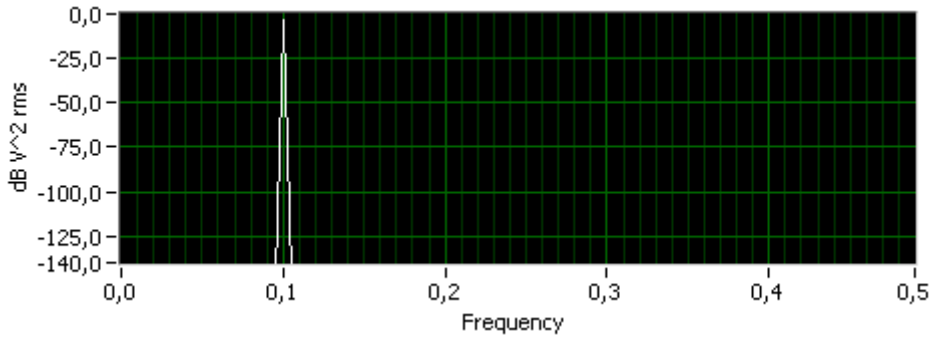


2.7.6 Mediante el instrumento virtual creado en el punto 2.7.4 demuestre el efecto ALIASING en el dominio de la frecuencia. Primero genere una señal senoidal de 10Hz muestreada a 100 muestras por segundo, luego ajuste la frecuencia de la señal a 90 Hz manteniendo la frecuencia de muestreo.

Señal de 10 Hz:

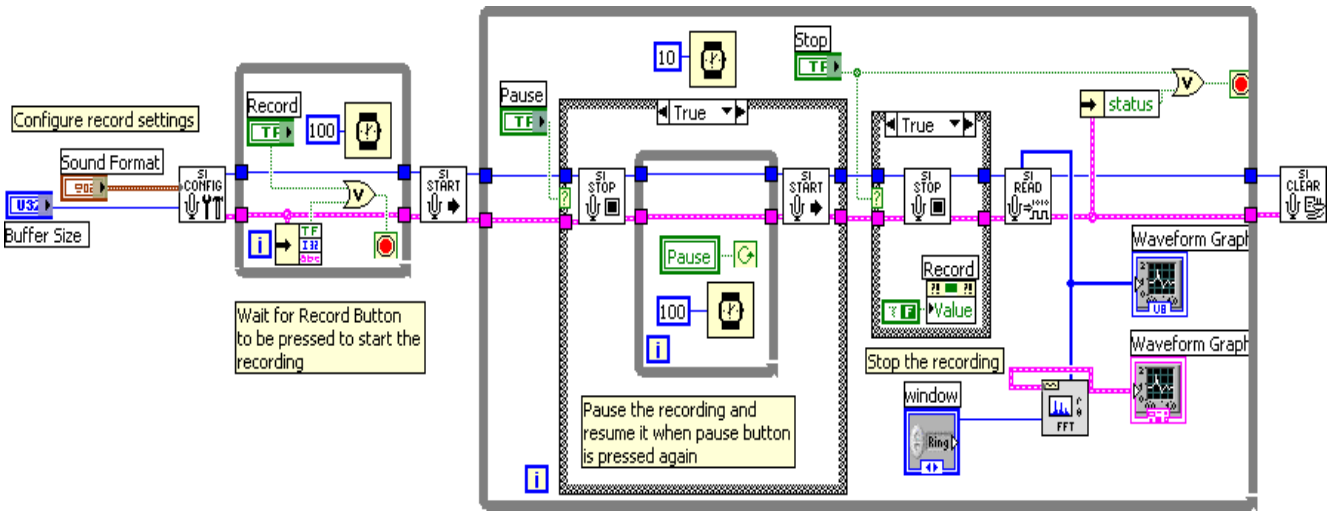


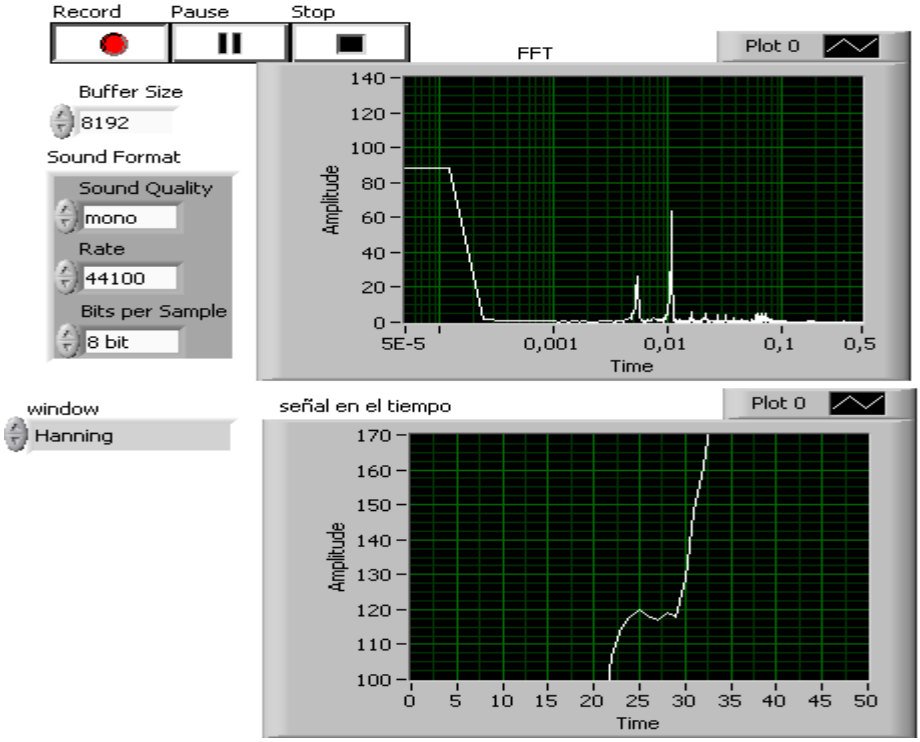
Señal de 90Hz:



Podemos observar que la señal de 90Hz aparece como si fuera de 10Hz debido al aliasing, (este efecto es utilizado por algunos osciloscopios especiales).

2.7.7 Realice un instrumento virtual que permita adquirir señales mediante la placa de sonido de la PC y muestre en pantalla la señal adquirida y el especto en frecuencia de dicha señal.




Record
Pause
Stop
Buffer Size
8192
Sound Format
Sound Quality
mono
Rate
44100
Bits per Sample
8 bit
FFT
Plot 0
Amplitude
Time
window
Hanning
señal en el tiempo
Plot 0
Amplitude
Time