

**DEPARTAMENTO DE ELECTRÓNICA Y AUTOMÁTICA**

**FACULTAD DE INGENIERÍA – UNIVERSIDAD NACIONAL DE SAN JUAN**

Informe de Practica Nº1

Introducción a MatLab y señales discretas

**Asignatura:** Procesamiento Digital de Señales

**Ingeniería Electrónica**

***Autor:***

*Avila, Juan Agustin – Registro 26076*

**1º Semestre**

**Año 2020**

# Introducción.

En esta práctica se desarrollarán los siguientes contenidos:

* Introducción a la graficación en MATLAB®.
* Gráfica de Funciones de Tiempo Discreto (DT).
* Periodicidad de Señales DT.
* Operaciones básicas sobre señales DT

# Actividades.

## Actividad 1.

Dada las siguientes señales de tiempo continuo, representarlas en el dominio continuo (CT) y en el dominio discreto (DT), empleando los comandos correspondientes. Tener en cuenta que el intervalo de tiempo seleccionado sea el apropiado. En el caso de las representaciones DT, el paso entre muestras debe ser 0,25. Paras las gráficas CT, emplear un paso entre datos igual o inferior a 0,01

a) x1(t)=6·sen(2𝜋𝑡)

b) x2(t)=5\*u(t-3)

c) x3(t)= 4·tri[(t+2)/4]

d) x4(t)= x1(t)+ x2(t)+ x3(t)

Encontrar el máximo y el tiempo del máximo de x4(n) y de x4(t). Compare los resultados obtenidos y saque conclusiones.

Para obtener las funciones y graficarlas se utilizó el siguiente código de matlab:

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Punto 1\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

%Funciones continuas

t=-6:0.01:10; %base temporal de señales continuas

x1t=6\*sin(2\*pi\*t);

x2t=5\*ustep(t-3);

x3t=4\*tri((t+2)/4);

x4t=x1t+x2t+x3t;

%Graficacion de funciones continuas

figure('Name','Grafica de funciones continuas');

subplot(2,2,1); plot(t,x1t);

title('x1(t)=6\*sen(2\*pi\*t)');

grid on; hold on;

subplot(2,2,2); plot(t,x2t);

title('x2(t)=5\*u(t-3)');

grid on;

subplot(2,2,3); plot(t,x3t);

title('x3(t)=4\*tri[(t+2)/4]');

grid on;

subplot(2,2,4); plot(t,x4t);

title('x4(t)=x1(t)+x2(t)+x3(t)');

grid on;

%Funciones discretas

n=-6:0.25:10;

x1n=6\*sin(2\*pi\*n);

x2n=5\*ustep(n-3);

x3n=4\*tri((n+2)/4);

x4n=x1n+x2n+x3n;

%Graficacion de funciones discretas

figure('Name','Grafica de funciones discretas');

subplot(2,2,1); dtplot(n,x1n);

title('x1(n)=6\*sen(2\*pi\*n)');

grid on;hold on;

subplot(2,2,2); dtplot(n,x2n);

title('x2(n)=5\*u(n-3)');

grid on;

subplot(2,2,3); dtplot(n,x3n);

title('x3(n)=4\*tri[(n+2)/4]');

grid on;

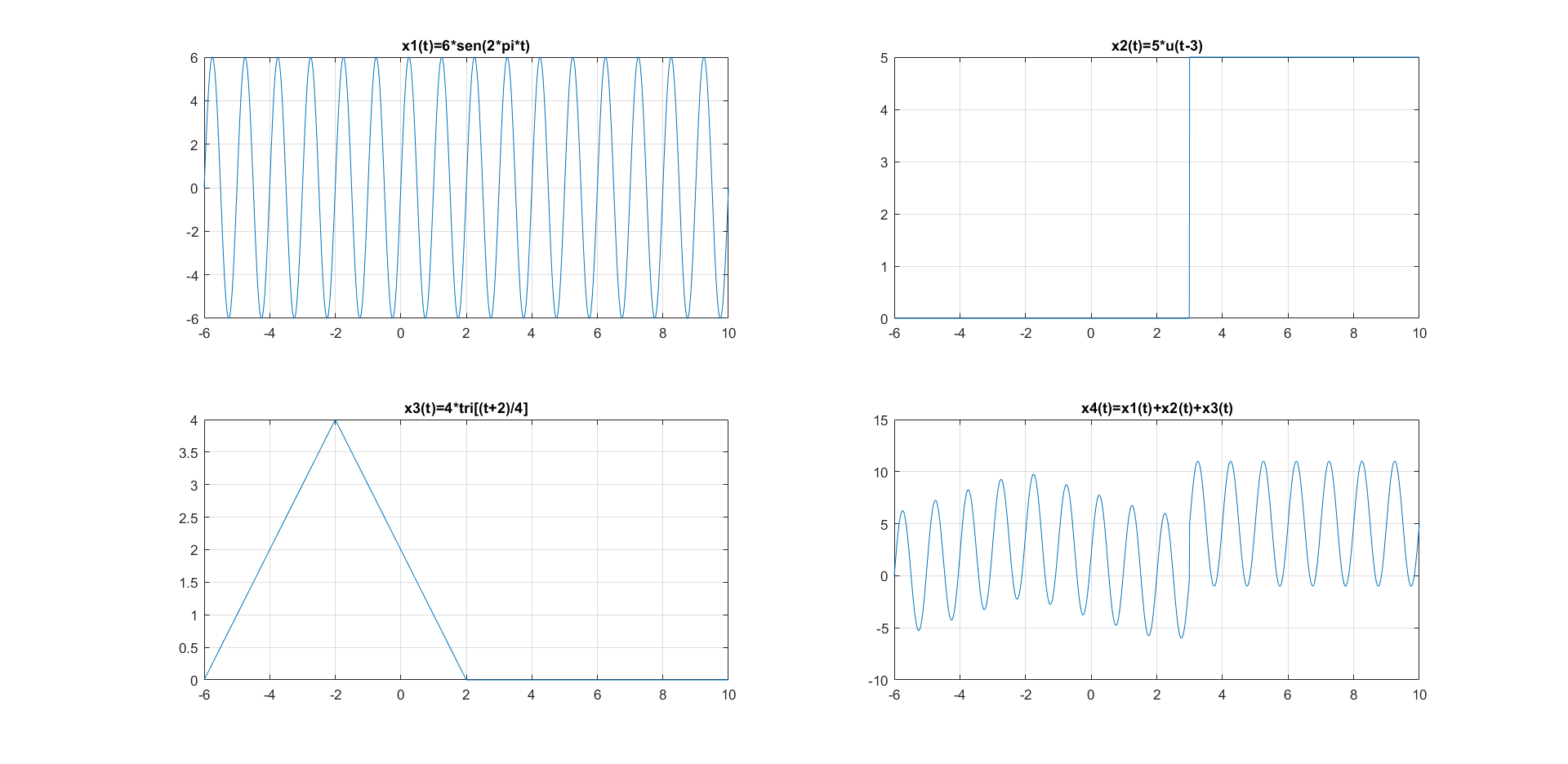
subplot(2,2,4); dtplot(n,x4n);

title('x4(n)=x1(n)+x2(n)+x3(n)');

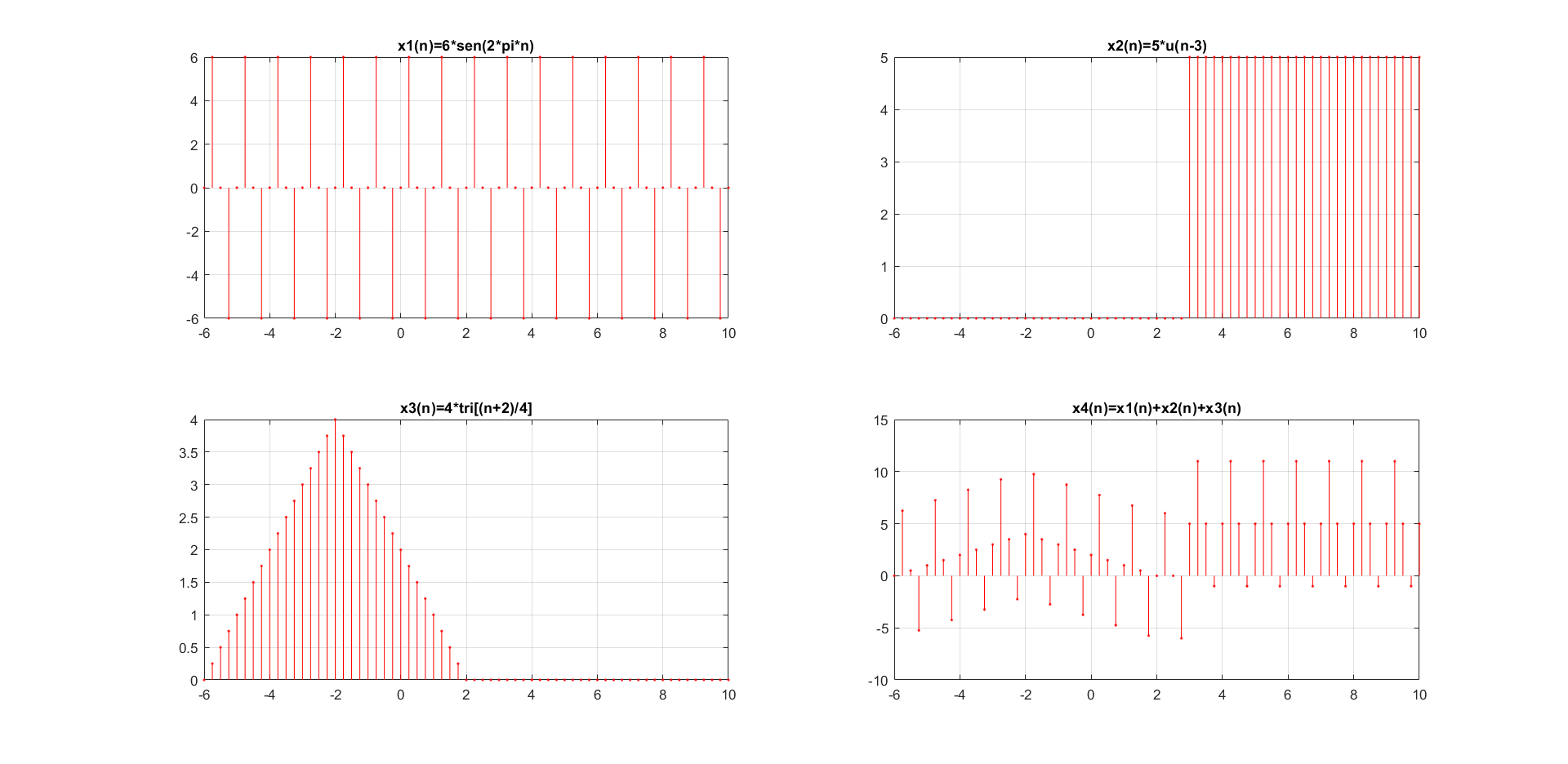
grid on;

A continuación se adjuntan las graficas con las funciones:

1 - Gráfica de funciones contínuas



2 - Gráfica de funciones discretas



Para obtener los máximos de x4(n) y x4(t) y sus respectivos tiempos se utilizaron los siguientes comandos:

M4c=max(x4t)

M4d=max(x4n)

iM4c=find(x4t==M4c);

iM4d=find(x4n==M4d);

tM4c=t(iM4c)

tM4d=n(iM4d)

Y se obtuvo los siguientes resultados:

M4c = 11

M4d = 11

tM4c = 3.2500 4.2500 5.2500 6.2500 7.2500 8.2500 9.2500

tM4d = 3.2500 4.2500 5.2500 6.2500 7.2500 8.2500 9.2500

Se observa que el valor máximo en ambas señales coincide en 11, y ese valor se repite periódicamente a partir de 3.25s cada 1s.

## Punto 2

Use la función dtplot o stem para graficar las siguientes funciones en el rango − 30 ≤ n ≤ 30 con paso unitario. Empleando el comando hold, grafique de manera superpuesta la envolvente continua de la señal usando el comando plot. Para la graficación de la envolvente, genere una nueva base de tiempo con paso t=0.01.

|  |
| --- |
| 1. 𝑦1(𝑛) = 2 cos(𝜋/2 𝑛) sen (𝜋/6 𝑛) |
| 1. 𝑦2(𝑛) = 2 cos(𝜋/2 𝑛) + 5 sen (𝜋/5 𝑛) |
| 1. 𝑦3(𝑛) = 5 sen(𝜋/8 𝑛 + 𝜋/4) + 5 cos (𝜋/5 𝑛 – 𝜋/4) |
| d) 𝑦4(𝑛) = cos(0.8𝑛) |

Para calcular las funciones y graficarlas se utilizó el siguiente script:

%\*\*\*\*\*\*\*\*\*\*\*\*\*\* Punto 2 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

n=-30:1:30; %paso unitario

t=-30:0.01:30; %paso de 0.01

y1n=2\*cos((pi/2)\*n).\*sin((pi/6)\*n);

y2n=2\*cos((pi/2)\*n)+cos((pi/5)\*n);

y3n=5\*sin((pi/8)\*n+pi/4)+5\*cos((pi\*n)/5-pi/4);

y4n=cos(0.8\*n);

y1t=2\*cos((pi/2)\*t).\*sin((pi/6)\*t);

y2t=2\*cos((pi/2)\*t)+cos((pi/5)\*t);

y3t=5\*sin(((pi\*t)/8)+pi/4)+5\*cos((pi\*t)/5-pi/4);

y4t=cos(0.8\*t);

%graficacion de funciones

figure('Name','y1(n), y1(t)');

dtplot(n,y1n); hold on; grid on; plot(t,y1t);

figure('Name','y2(n), y2(t)');

dtplot(n,y2n); hold on; grid on; plot(t,y2t);

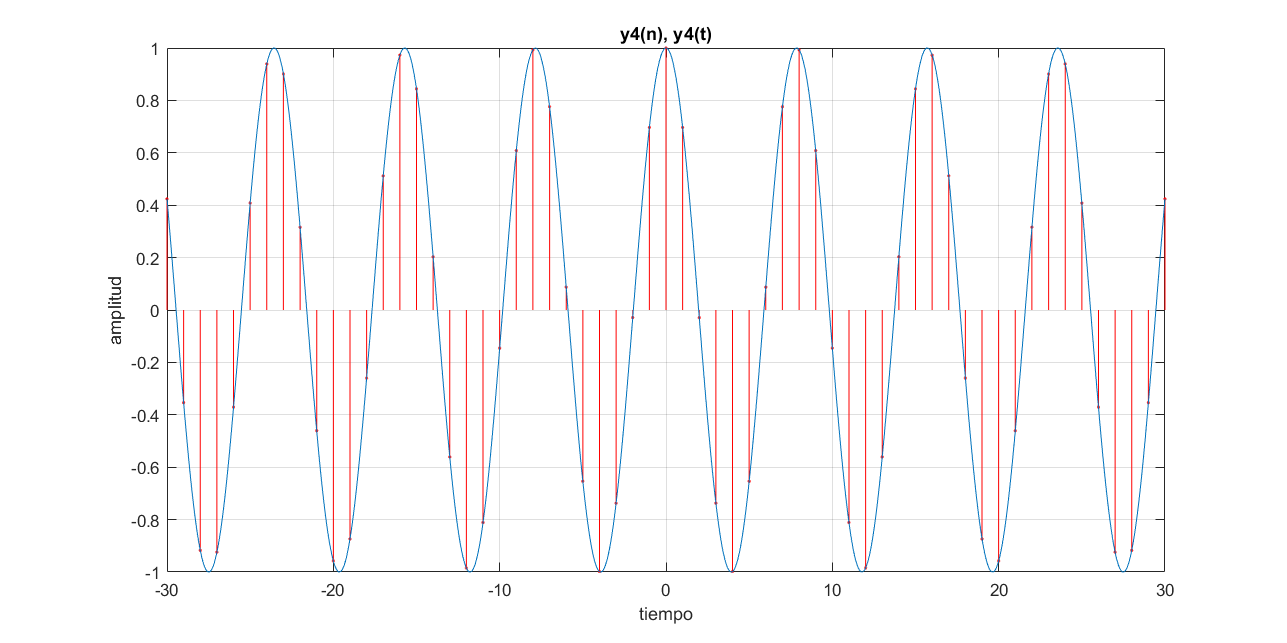
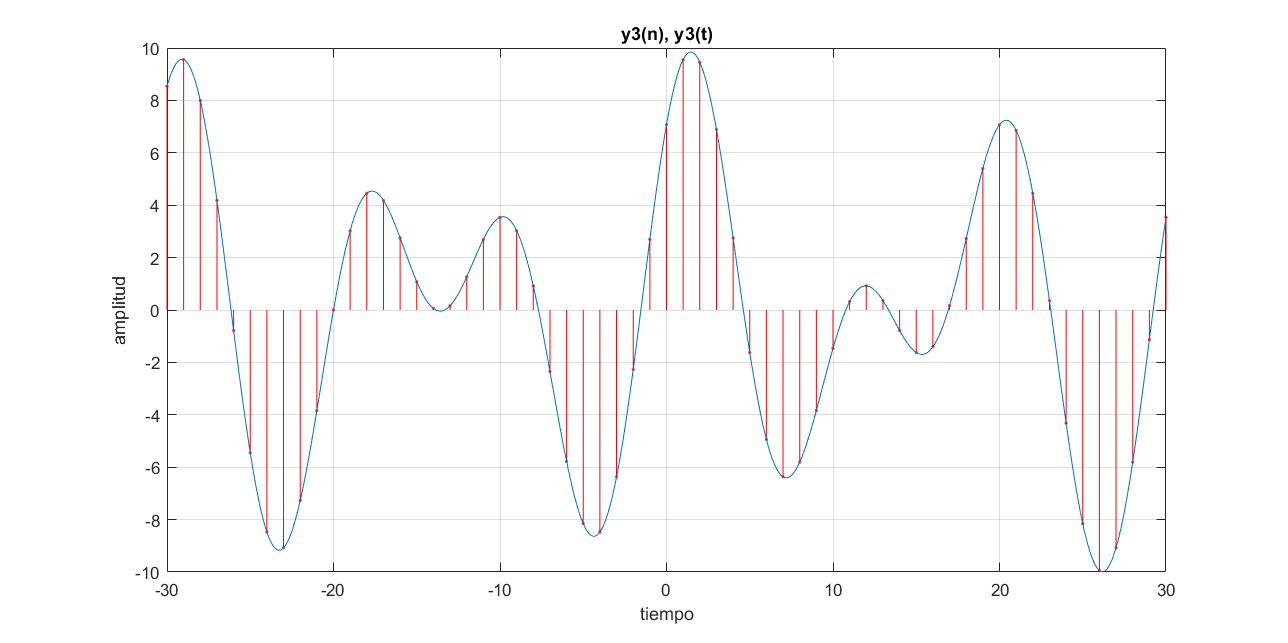
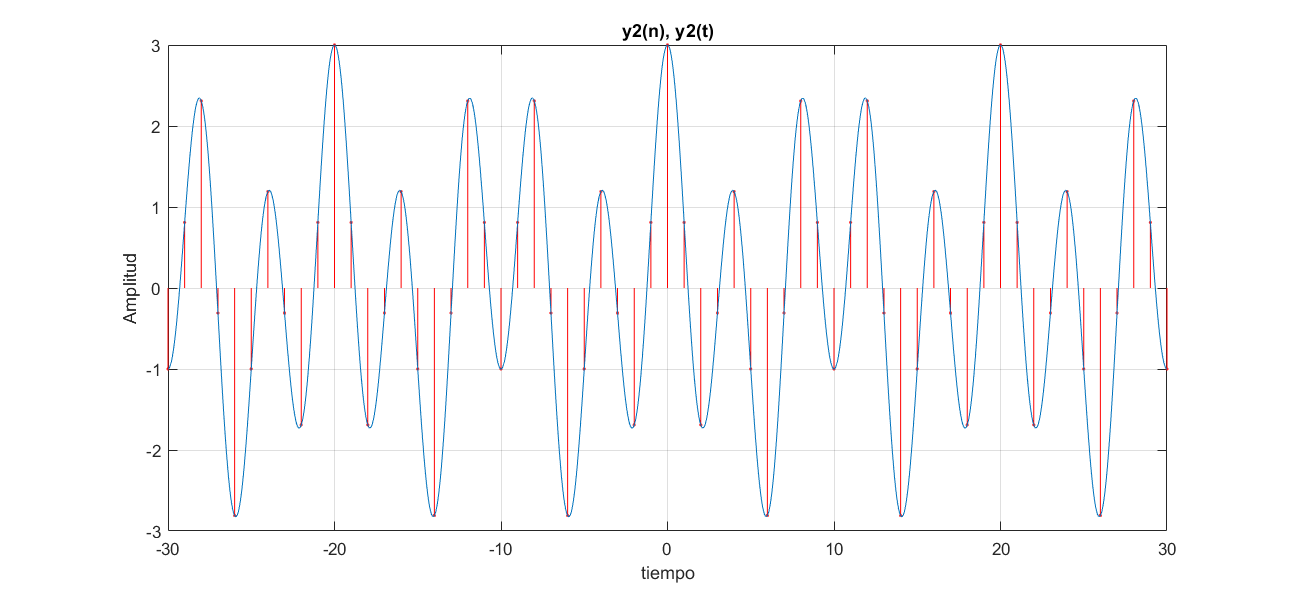
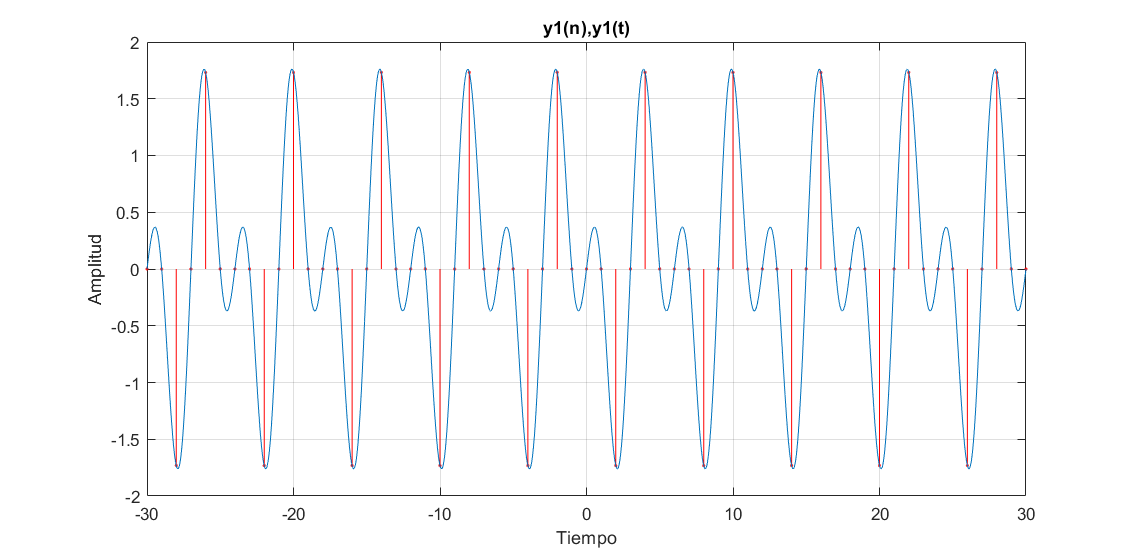
figure('Name','y3(n), y3(t)');

dtplot(n,y3n); hold on; grid on; plot(t,y3t);

figure('Name','y4(n), y4(t)');

dtplot(n,y4n); hold on; grid on; plot(t,y4t);

Y a continuación se adjuntan los gráficos correspondientes:



A continuación, se determinó si las señales mantienen la periodicidad:

Por lo tanto, F1 tiene un periodo de 4 y F2 un periodo de 12. Se busca el periodo de ambas señales en conjunto con el método del mínimo común múltiplo:

N1=lcm(4,12)

N1 = 12, por lo tanto el periodo de y1 es de 12.

Por lo tanto, F1 tiene un periodo de 4 y F2 un periodo de 10. Se busca el periodo de ambas señales en conjunto con el método del mínimo común múltiplo:

N2=lcm(4,10)

N2 = 20, por lo tanto el periodo de y2 es de 20.

Por lo tanto, F1 tiene un periodo de 16 y F2 un periodo de 10. Se busca el periodo de ambas señales en conjunto con el método del mínimo común múltiplo.

N3=lcm(16,10)

N3 = 80, por lo tanto el periodo de y3 es de 80.

Esta relación da como resultado una frecuencia digital irracional, por lo tanto no es una función periódica.

## Punto 3

Implemente el código que se le propone a continuación como guía, en el cual se realizan operaciones básicas sobre una señal de audio. Usted podrá adquirir una señal de audio a través de un micrófono o importar un archivo de audio usted mismo. En este último caso deberá verificar que el formato a importar sea el adecuado para su tratamiento bajo el entorno de Matlab.

Puede introducir cambios en los valores del código a fin de experimentar. Para poder verificar las operaciones realizadas, conecte unos auriculares a la salida de audio de su computadora.

Para que el script fuera compatible con la versión de matlab R2017b, se tuvo que modificar la adquisición de audio ya que a partir de la versión R2015a ya no incluía la herramienta “wavrecord()”. A su vez, el script contaba con un error en la variable “y”(al realizar el eco). Una variable era un vector fila y la otra era un vector columna, por lo cual generaba una matriz de 33k x 33k (en 3 segundos de audio), parando la ejecución del script por falta de memoria. A continuación se transcribe el código modificado:

%Laboratorio Nº 1 - Ejercicio 3

%Operaciones Básicas con Señales Discretas

%---------------------------------------------------------------

% Inicialización

clear all;

close all;

% Adquisición de audio con Fs

Fs=11025; % Este valor puede ser modificado para experimentar

r=audiorecorder(Fs,8,1);%estas tres lineas

recordblocking(r,3) %reemplazan al comando

x=getaudiodata(r); %wavrecord

soundsc(x, Fs);

input('sonido original');

N = length(x);

%---------------------------------------------------------------

% Generación de eco mediante la suma de la versión retrasada de la señal.

delay = 0.4 % Este valor puede ser modificado para experimentar

atten = 0.5 % Este valor puede ser modificado para experimentar

n0 = round(delay.\*Fs)

n\_ext = [1:1:N+n0]';

x\_delay(n0+1:N+n0) = x(n\_ext(n0+1:N+n0)-n0);

x\_delay=x\_delay(:); %se transpolo este vector para evitar errores

x\_extend = x;

x\_extend(N+1:N+n0) = zeros(1,n0);

y = x\_extend + atten.\*x\_delay; %aqui generaba una matriz enorme

t\_extend = (n\_ext-1)./Fs;

subplot(2,1,1), plot(t\_extend,x\_extend);

xlabel('t seg');

title('Sonido Original');

soundsc(x,Fs);

input('Sonido con Eco')

subplot(2,1,2), plot(t\_extend,y);

xlabel('t seg');

title('Sonido con Eco');

soundsc(y,Fs);

%---------------------------------------------------------------

% Reflexión

n = 1:1:N;

minus\_n = N+1 - n;

z = x(minus\_n);

t = (n-1)./Fs;

input('Sonido Original')

figure;

subplot(2,1,1), plot(t, x);

xlabel('t seg');

title('Sonido Original');

soundsc(x,Fs);

input('Sonido Reflejado')

subplot(2,1,2), plot(t, z);

xlabel('t seg');

title('Sonido Reflejado');

soundsc(z,Fs);

%---------------------------------------------------------------

% Submuestreo por 2 - Decimación

%Procesamiento Digital de Señales Ciclo 2020 Laboratorio 1

z = zeros(1,N);

z(1:ceil(N/2)) = x(1:2:N);

z(ceil(N/2)+1:N) = zeros(1,N-ceil(N/2));

input('Sonido Original')

figure;

subplot(2,1,1), plot(t,x);

xlabel('t seg');

title('Sonido Original');

soundsc(x,Fs);

input('Sonido Submuestreado por 2')

subplot(2,1,2), plot(t,z);

xlabel('t sec');

title('Sonido Submuestreado por 2');

soundsc (z,Fs);

%---------------------------------------------------------------

% Sobremuestreo por 2

z = zeros(1,2.\*N);

z(1:2:2.\*N) = x(1:N);

x\_extend(N+1:2.\*N) = zeros(1,N);

n\_extend = 1:1:2.\*N;

t\_extend = n\_extend./Fs;

input('Sonido Original')

figure;

subplot(2,1,1), plot(t\_extend,x\_extend);

xlabel('t seg');

title('Sonido Original');

soundsc(x,Fs);

input('Sonido Sobremuestreado por 2')

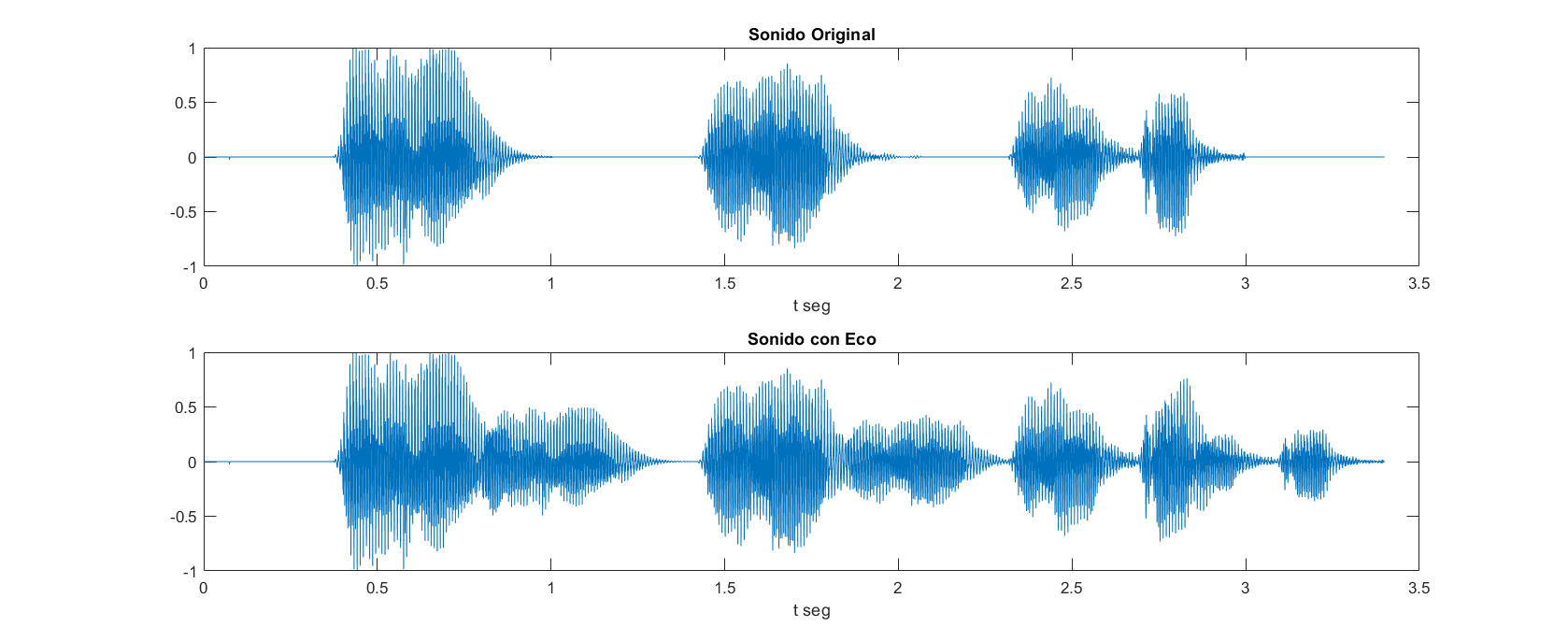
subplot(2,1,2), plot(t\_extend,z);

xlabel('t seg');

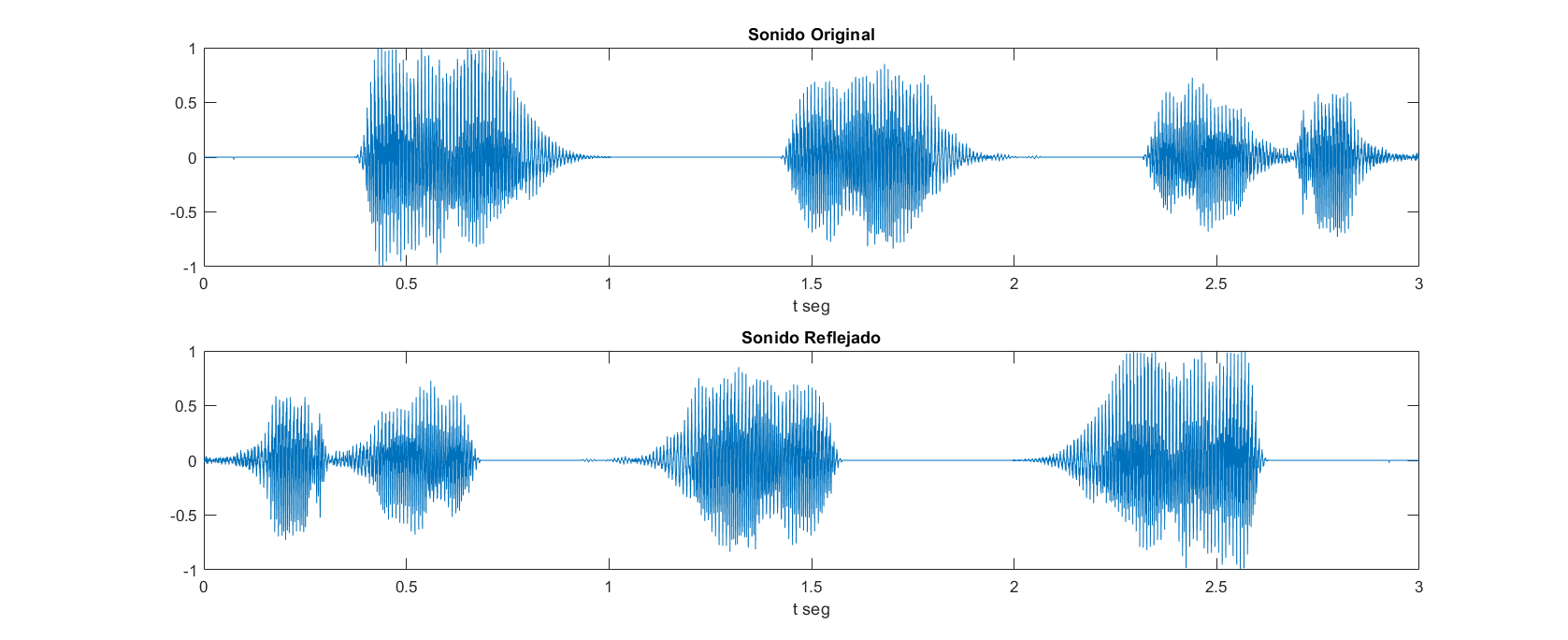
title('Sonido Sobremuestreado por 2');

soundsc(z,Fs);

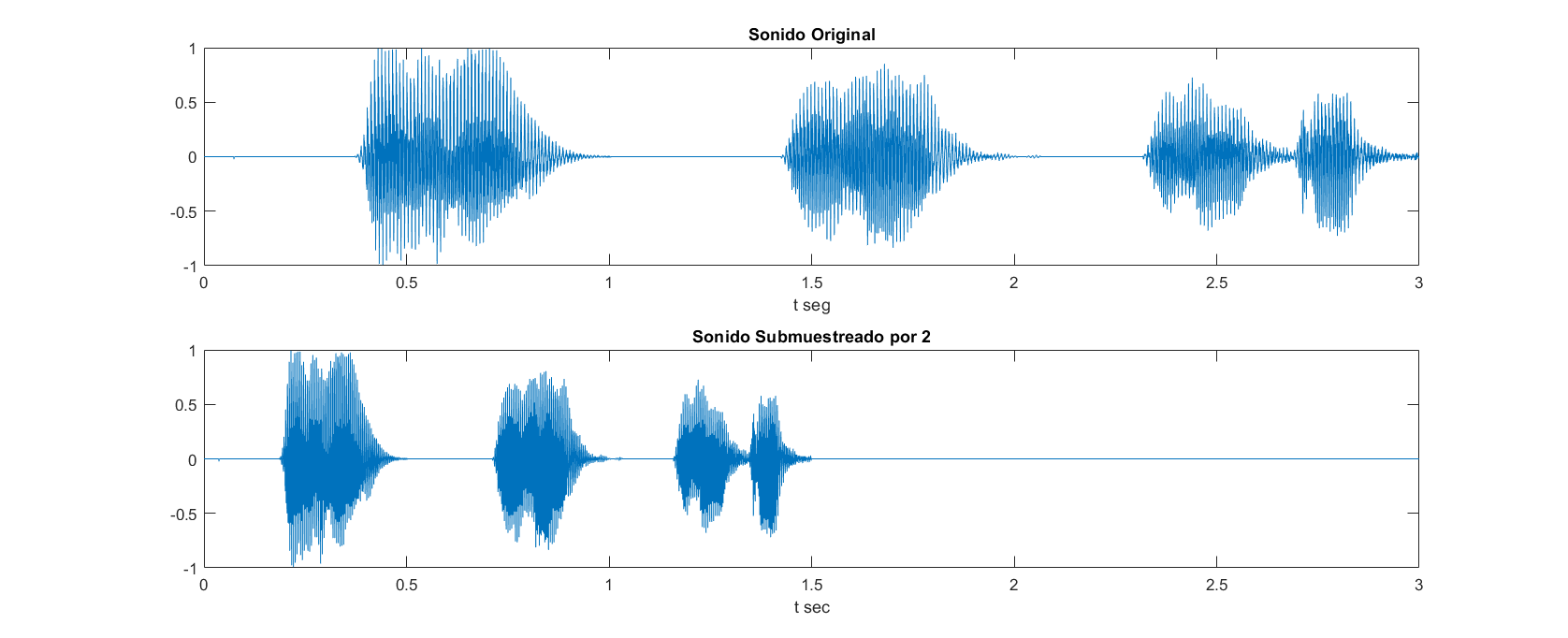
Y se obtuvieron los siguientes resultados:



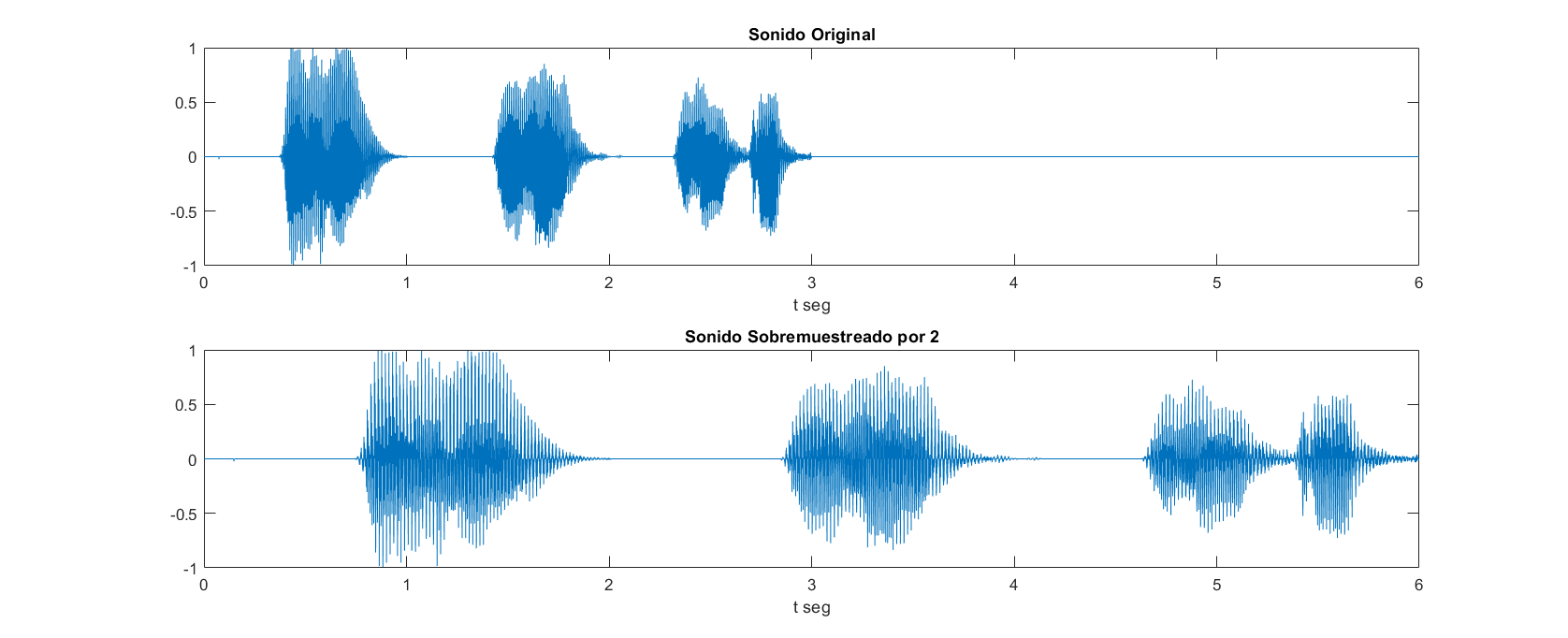
En este caso se observa cómo se repite la señal original con un delay y una atenuación, generando así un efecto de “eco”.



En el segundo caso, se observó una reflexión de la señal, es decir que se reprodujo de atrás hacia adelante, sin otra modificación a la señal original.



En este caso se submuestreó la señal original por un factor de 2, produciendo que la onda modificada dure la mitad de la original. Lo cual provoca que se escuche el doble de rápido y que todos los tonos se escuchen más agudos.



En este caso se sobremuestreó la señal original por un factor de 2, generando que la señal modificada duplique su duración. Al reproducirse, se escucha más lento que el original, con los tonos escuchándose más graves. Además, se escucha un “silbido”, que se asume que se debe a que los espacios vacíos fueron rellenados con ceros.