Instituto Politécnico Nacional

**Unidad Profesional Interdisciplinaria de Biotecnología**

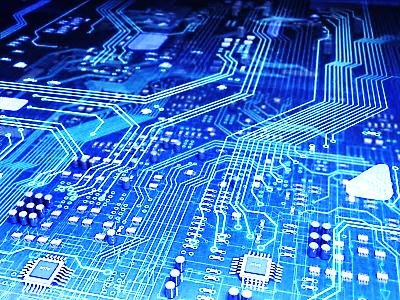


Materia:  
Procesamiento Digital De Bioseñales e Imágenes.

Practica 2

Generación y Muestreo de Señales con Matlab y Simulink

Profesores:

Venegas Anaya Darinel

|  |  |
| --- | --- |
| **González Alvarado Alejandro**  **Carlos**  **Iturbe Gil Carlos**  5MV2 |  |

México, DF. a 6 de Agosto del 2017

**Objetivo General**

Representar y reconstruir señales de tiempo continuo y tiempo discreto a través del muestreo de una señal y procesar las señales como funciones matemáticas.

**Objetivos Específicos**

Representar señales o funciones de tiempo continuo y discreto mediante la herramienta de cómputo Matlab®.

Realizar la discretización de una señal mediante la Frecuencia de Muestreo

Reconstruir funciones discontinuas como función pulso o triangular a partir de funciones especiales como función escalón y función rampa.

Reconstruir funciones definidas por intervalos mediante el proceso de ventana

Utilizar la herramienta Simulink® de Matlab® para la reconstrucción y representación de señales

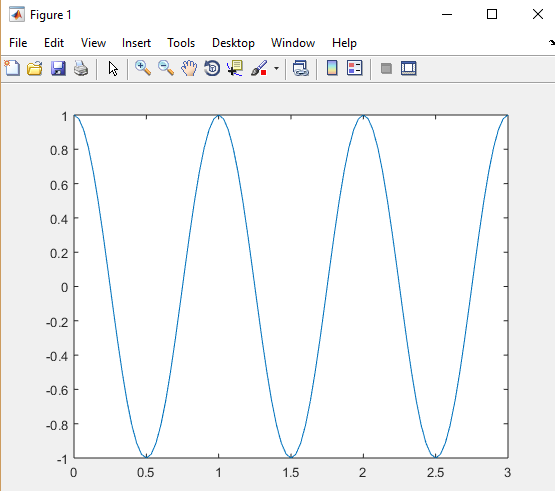
Implementar un programa de Matlab para la generación de gráficas de la representación de una señal de tiempo continuo o tiempo discreto

**Desarrollo**

**1.1. Señal continúa**

En la primer parte se generaron las siguientes ecuaciones con una frecuencia de muestreo de 30[Hz] y 3 ciclos como se muestra en el siguiente orden, a cada gráfica le correspondían 3 ciclos.

**Figura1. X1 (t) = cos(2πf)**

****

**F=1 T=1 Wo=2π**

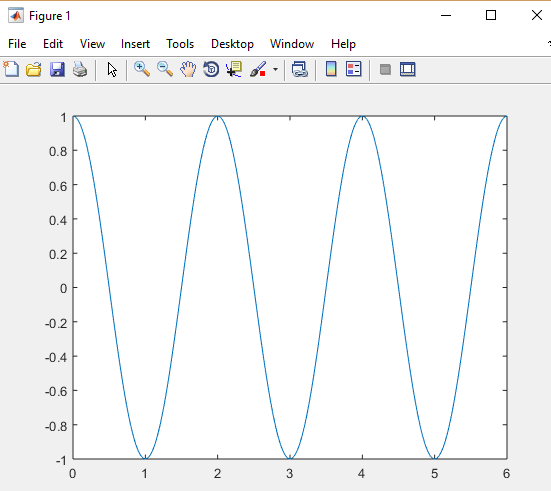
fs=30

x=0:1/fs:2;

y=cos(2\*pi\*x);

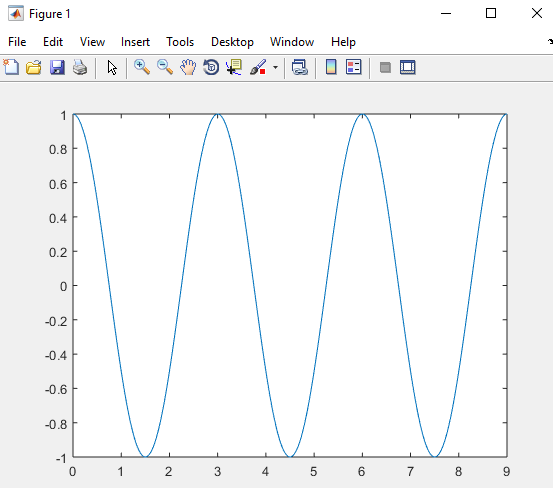
plot(x,y)

**Figura2. X2 (t) = cos(2π 2 f)**

****

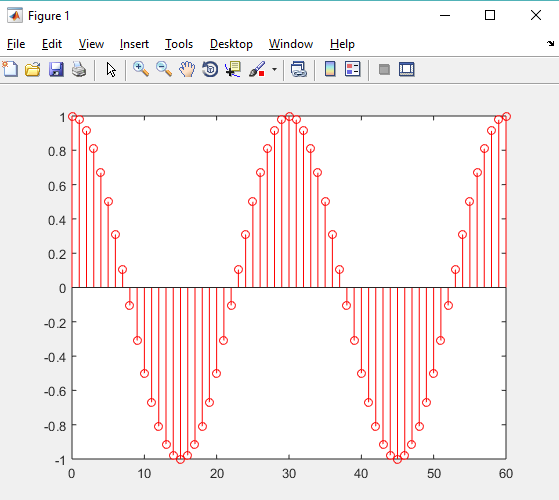
**F=1/2 T=2 Wo=2π/2**

**Figura 3. X3 (t) = cos(2π 3 t)**

****

**F=1/3 T=3 Wo=2π/3**

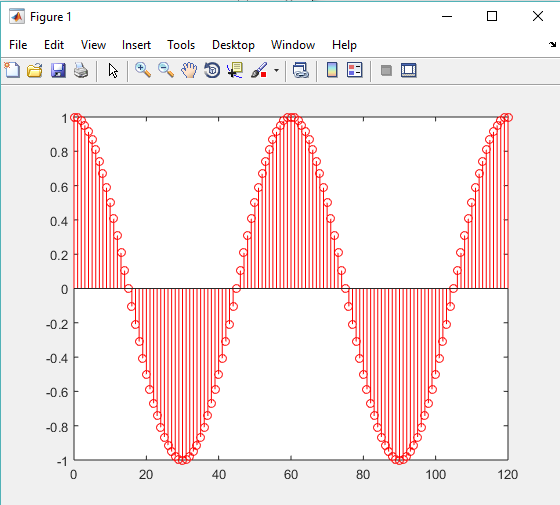
**1.2. Señal discreta**



Dado que X1=cos((2\*pi\*t)

X1=cos((2\*pi\*n)/30);

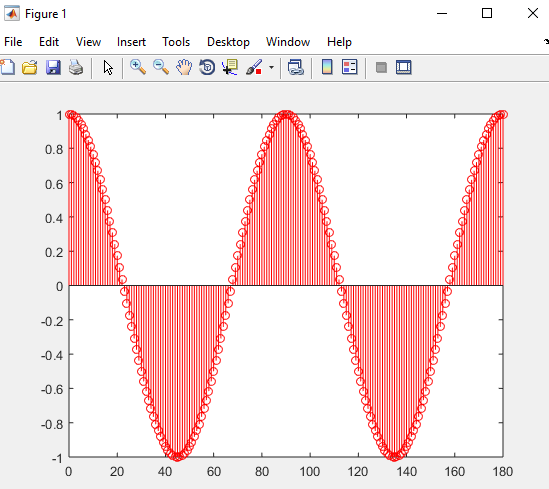
Td= 30 Fd(Wod)==2π/30 fd=1/30



Dado que X2=cos((2\*pi\*t/2)

X2=cos((2\*pi\*n)/60);

Td=6 0 Fd(Wod)==2π/60 fd=1/60



Dado que X3=cos((2\*pi\*t/3)

X3=cos((2\*pi\*n)/90);

Td=90 Fd(Wod)=2π/60 fd=1/90

n=0:60;

x1=cos(2\*pi/30\*n);

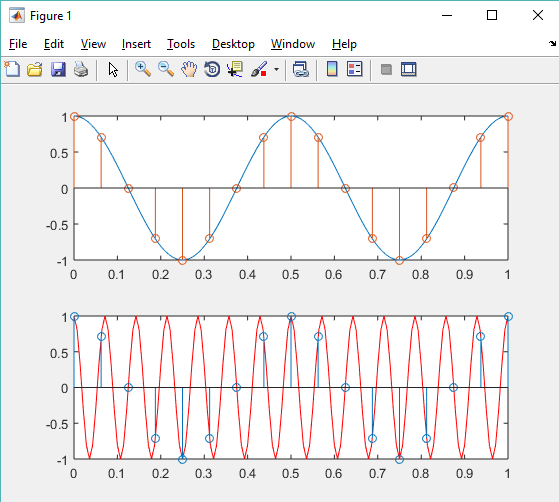
stem(n,x1,'r')

**1.3. Teorema de muestreo**

Considerando las siguientes señales continúas, se obtuvo lo siguiente:

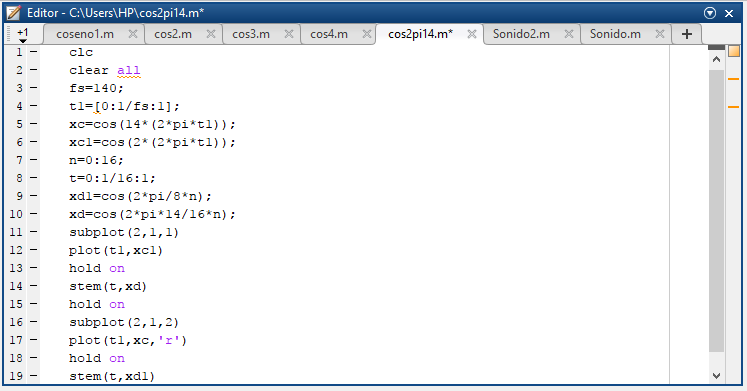
x4 (t) = cos(2π2t)

x5 (t) = cos(2π14t)



En esta grafica se puede apreciar un efecto de Aliasing debido a una mala determinación de frecuencia de muestreo.

El Teorema de Nyquist indica que la frecuencia de muestreo mínima que tenemos que utilizar debe ser mayor que *2·fmax*, donde *fmax* es la frecuencia máxima de la señal compleja. Si utilizamos esa frecuencia de muestreo, podremos reproducir posteriormente la señal a partir de las muestras tomadas



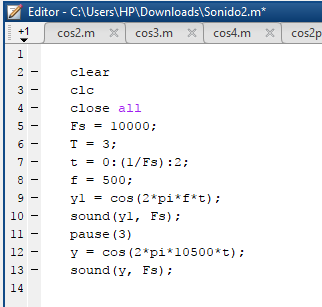
**1.4. Audio**

Generando dos señales cosenoidales que pueda reproducirse como una señal de audio, las cuales deben tener las siguientes características:

x5 (t) = cos(2πfAUDIO ∗t)

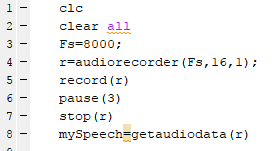
x6 (t) = cos(2π [Fs−fAUDIO]∗t)

A continuación se mostrara el código, al compilar y ejecutar este generara 2 sonidos en los cuales podemos apreciar el efecto de splicing



Con la ayuda de la entrada de audio

1. Obtenga una señal de audio a una frecuencia de muestreo adecuada Fs.



b) Reproduzca la señal anterior con frecuencias de muestreo mayor y menor a la frecuencia de muestreo del inciso anterior.

