Tecnológico de Costa Rica
Escuela de Ingeniería Electrónica
Procesamiento Digital de Señales
Kendall González
Carlos Peralta

Tarea #1

Ejercicio 1

Plot

Esta función es parte de la biblioteca estándar de GNU/Octave, es decir, no ocupa instalar complementos o extensiones para su funcionamiento. La misma genera gráficos en dos dimensiones. Esta función ha sido sobrecargada, es decir, posee el mismo nombre y diferente posible combinación de parámetros. El funcionamiento está dado por la siguiente sintaxis:

```
>> plot(Y);
>> plot(X, Y);
>> % Propiedad - Valor se puede repetir muchas veces, pero siempre vienen en parejas.
>> plot(X, Y, propiedad, valor, … );
>> plot (X, Y, FMT, ...)
```

Donde Y es un arreglo de coordenadas (x,y) o también un arreglo en cascada de alguna función que genera dicho arreglo de coordenadas (vectores), por ejemplo, para gráficar en su forma más simple la función seno dentro del dominio [0, 2pi], se puede realizar de la siguiente forma:

```
>> x = 0:0.1:2*pi;
>> plot(sin(x));
```

El arreglo de datos ‘x’ se asigna mediante la sintaxis x = 0:0.1:2*pi, lo cual se interpreta como, “asigne todos los números que estén dentro de (inclusive) 0 a 2pi con un step de 0.1”.

Las propiedades que se pueden específicar en los parámetros son los siguientes, con sus respectivos valores posibles:

- "linestyle”
  - '-' Para línea sólida.
  - '--' Para línea segmentada.
  - ':' Para línea punteada.
  - '-.' Para línea segmentada-punteada.
- "linewidth"
  - Número para indicar el ancho de la línea.

- "color"
  - 'k' Negro.
  - 'r' Rojo.
  - 'g' Verde.
  - 'b' Azul.
  - 'y' Amarillo.
  - 'm' Magenta.
  - 'c' Cyan.
  - 'w' Blanco.
- "marker"
  - '+' Mira.
  - 'o' Circulo.
  - '*' Estrella.
  - '.' Punto.
  - 'x' Cruz.
  - 's' Cuadrado.
  - 'd' Rombo.
  - '^' Triangulo hacia arriba.
  - ‘v' Triangulo hacia abajo.
  - '>' Triangulo hacia la derecha.
  - '<' Triangulo hacia la izquierda.
  - 'p' Pentagrama.
  - 'h' Hexagrama.
- "markersize":
  - Número para indicar el tamaño del marcador.
- "markeredgecolor"
  - Ver color.
- "markerfacecolor"
  - Ver color.

El el achivo Exercise1.m encontrará varias funciones, una de ella es grapher, que recibe como parámetro un vector con valores independientes a evaluar en la función seno. Su gráfica es la siguiente:

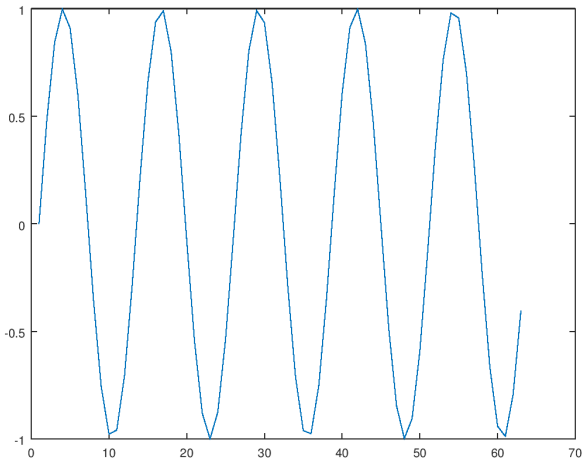


Gráfico 1. Función seno con dominio [0, 10pi], con un intervalo de 0,5.

Stem

Esta función posee una sintaxis similar a la de plot, sin embargo, stem gráfica la función de una forma discreta, no continua como plot. En el archivo Exercise1.m encontrará otra función denominada grapherDiscrete, recibe los mismos parámetros que la función anterior. Su gráfica es la siguientes

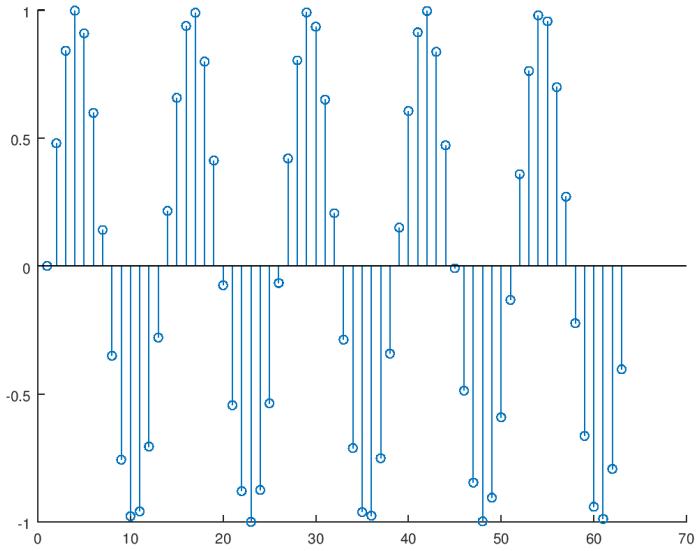


Gráfico 2. Función seno con dominio [0, 10pi], con un intervalo de 0,5 discreto.

Hold

Este comando permite indicarle al motor de gráficos si el resultado de plot y stem debe generarse en nuevas ventanas, o generar una superposición de estas.

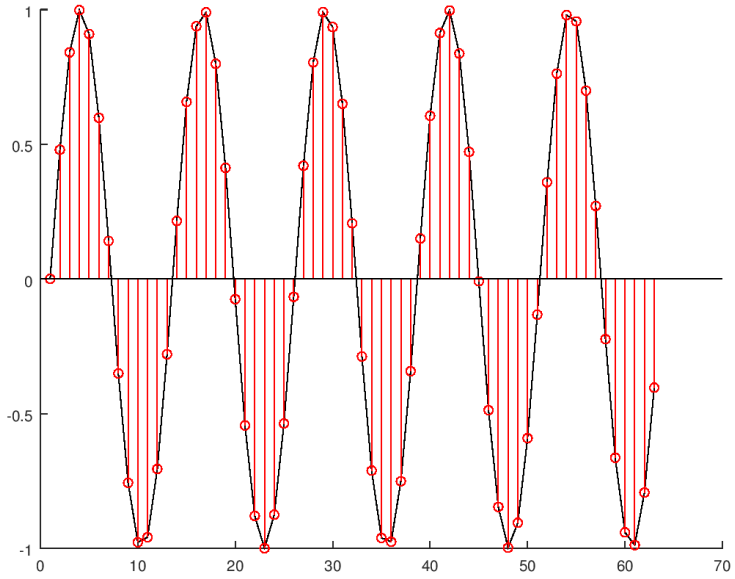


Gráfico 3. Gráficos 1 y 2 superpuestos gracias a hold.

Figure

Este comando permite generar nuevas ventanas para generar gráficos en ellas.

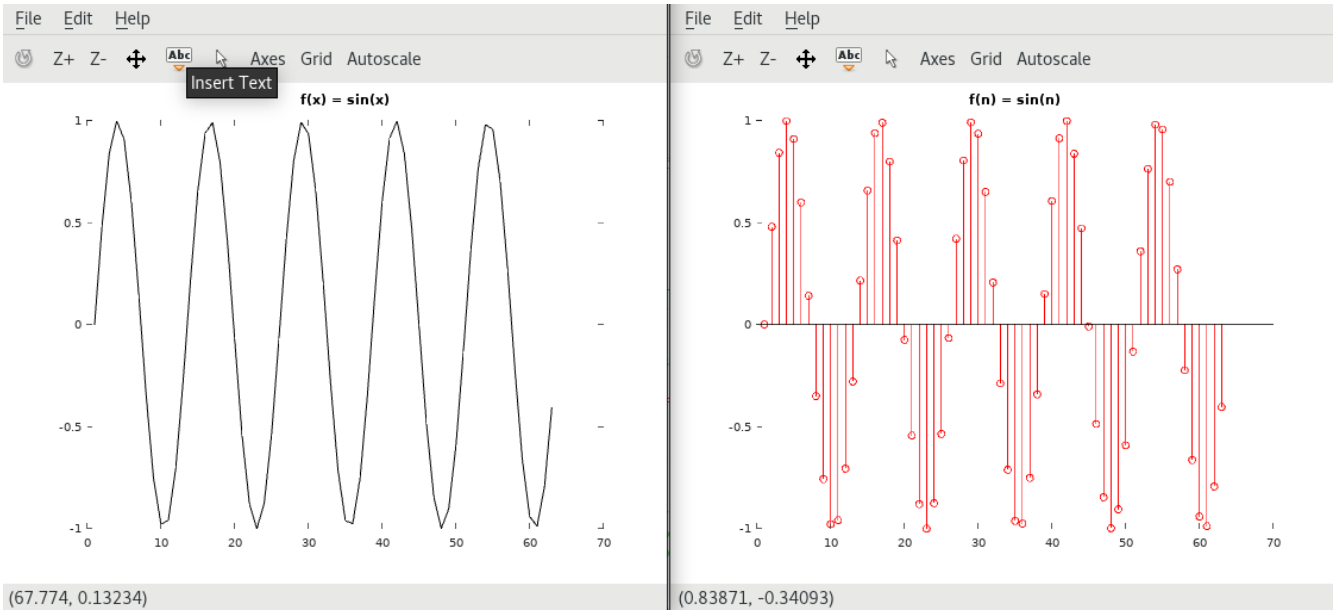


Imagen 1. Gráficos 1 y 2 en ventanas independientes.

Subplot

Este comando permite generar gráficos en una misma ventana, generando una grilla.

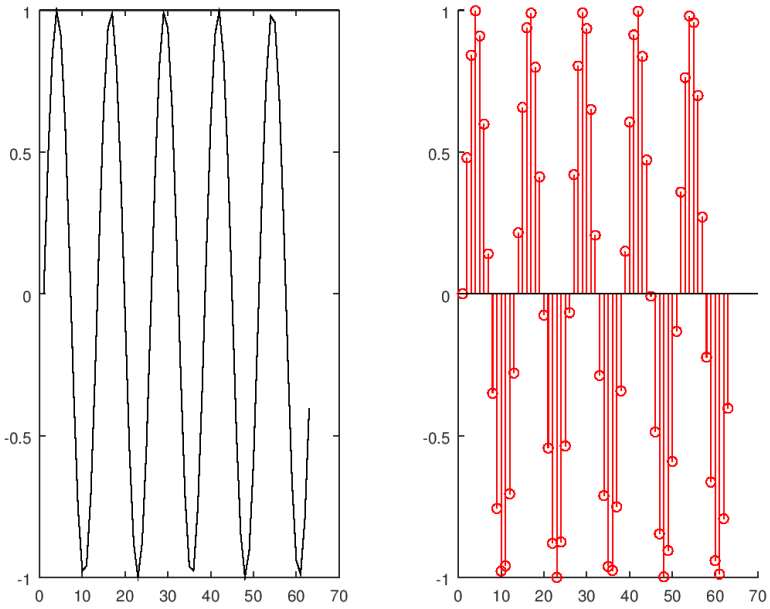


Imagen 2. Gráficos 1 y 2 en una misma ventana.

Ejercicio 2

Gráficas

Se pide graficar y(n) = sen(πfn), con n de 0 a 100, utilizando las siguientes frecuencias:

- f = 0,01
- f = 0,02
- f = 0,05
- f = 0,1

Se puede encontrar en el archivo Exercise2.m el programa en Octave encargado de realizar las gráficas, de las cuales se obtuvieron las siguientes imágenes:

Ventana Propia

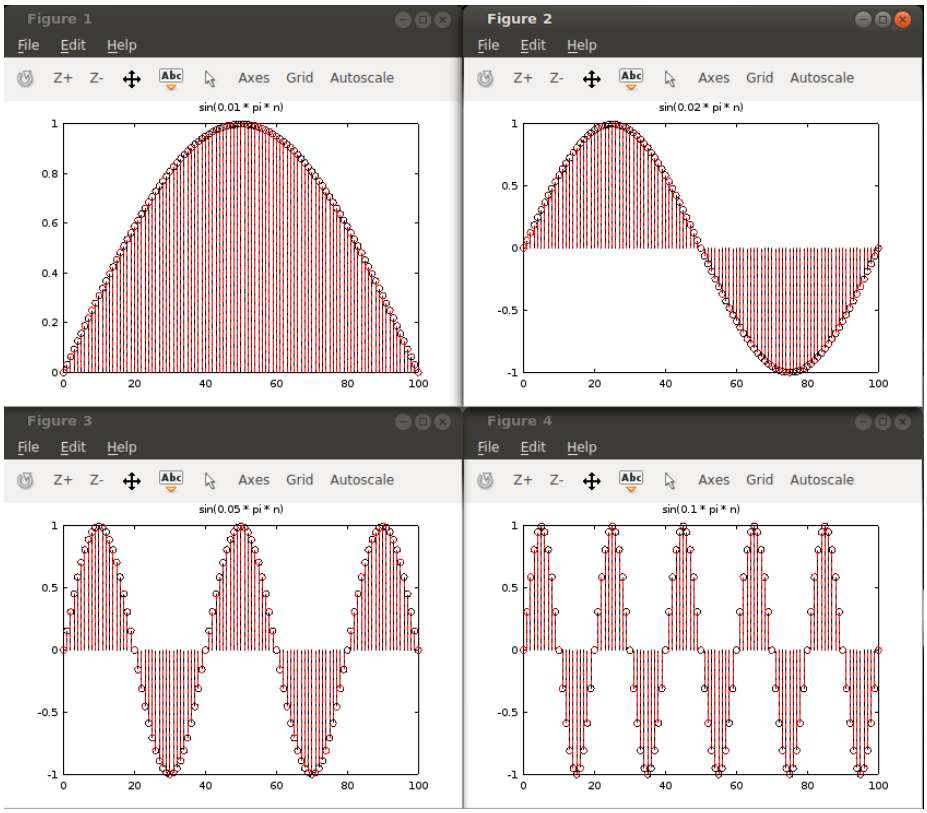


Imagen 3. Gráficas en ventanas separadas.

Misma ventana

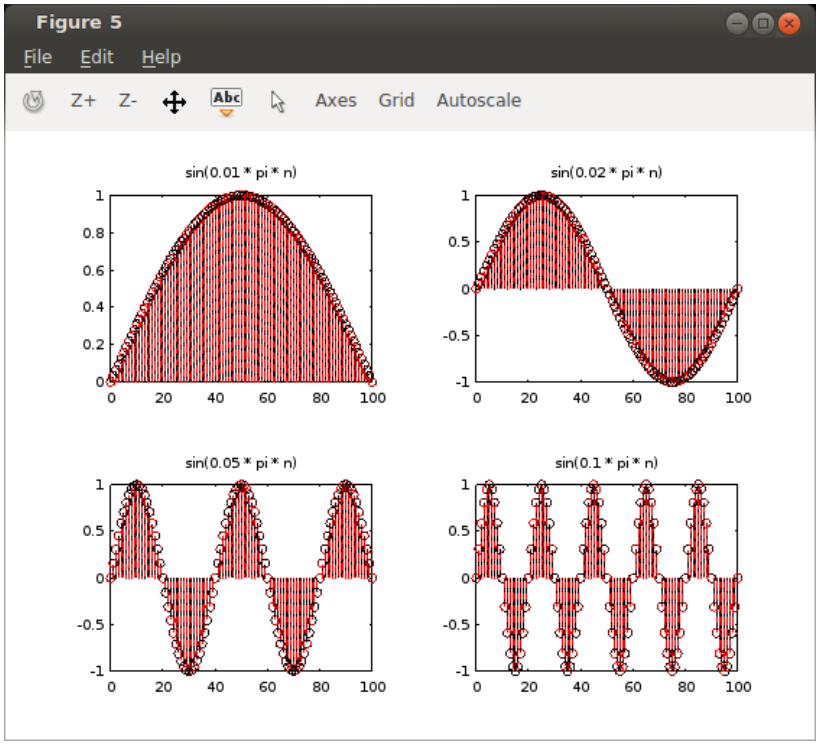


Imagen 4. Gráficas separadas en la misma ventana.

Traslapadas

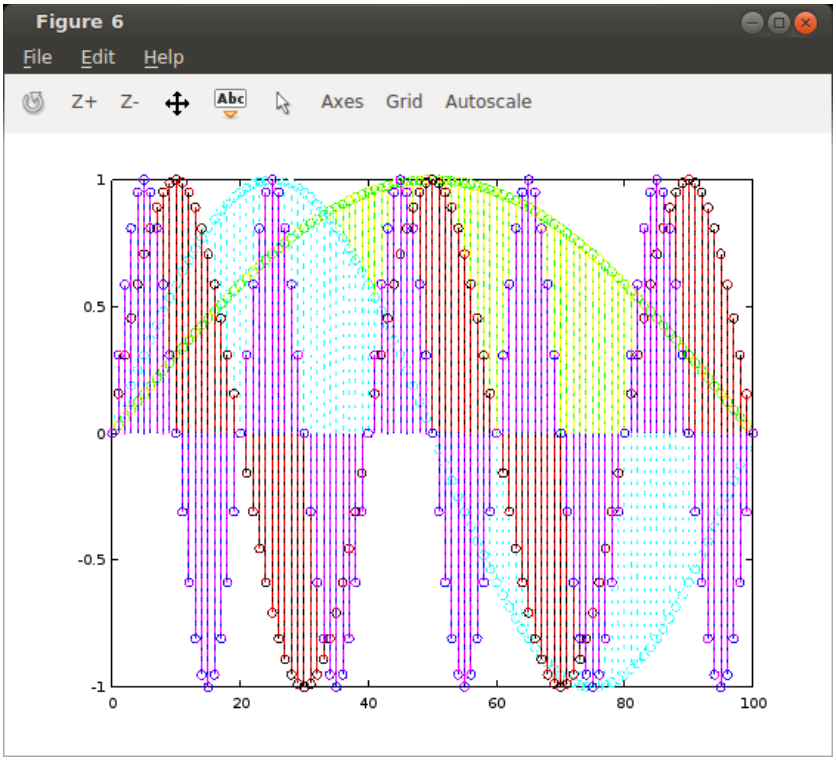


Imagen 5. Gráficas traslapadas.

Stem y Plot

La principal diferencia entre ambos comandos es que plot genera gráficas continuas de la función ingresada, mientras que stem genera gráficas discretas. Se puede observar en el gráfico 1 el resultado de plot y en el gráfico 2 el resultado de stem, para notar más la diferencia.

Cantidad de muestras

Para calcular la cantidad de muestras *N* en el periodo de cada función se utiliza la siguiente ecuación:

Como: $y(n) = sen(\pi * f * n)$

(1) $N = \frac{2}{f}$

- f = 0,01 => N = 200
- f = 0,02 => N = 100
- f = 0,05 => N = 40
- f = 0,1 => N = 20

Ejercicio 3

Para resolver este problema primero se debe establecer el concepto de Periodicidad de una función, una función $x(t) = \sin(\omega t + \theta)$ es periódica si cumple lo siguiente:

$$x(t+N) = x(t), \forall N \in \mathbb{Z}$$

Considérese de nuevo la señal sinusoidal, cuyo argumento es πf . La función será periódica siempre y cuando se cumpla lo siguiente:

$$\pi f_0 + 2\pi = \pi F \Rightarrow \pi(f_0 + 2) = \pi F \Rightarrow f_0 + 2 = F$$

Por lo tanto, la función de Octave/Matlab encargada de calcular la siguiente frecuencia alias realizará el siguiente cálculo

$$F = f_0 + 2$$

Donde f_0 es la frecuencia que se recibirá por parámetro y F la frecuencia alias siguiente. Es importante el hecho de que en este resultado se suma 2 y no 1 , según vimos en clase, esto radica por el hecho de que la función sinusoidal tiene la forma de $y(n) = \sin(\pi n f)$, donde su frecuencia angular no posee la constante 2 al frente. Por otro lado, es importante recalcar que para que este

concepto funcione, la frecuencia debe poder ser expresada como una fracción de números primos relativos.

Aplicando los resultados anteriores, se tiene las siguientes gráficas:

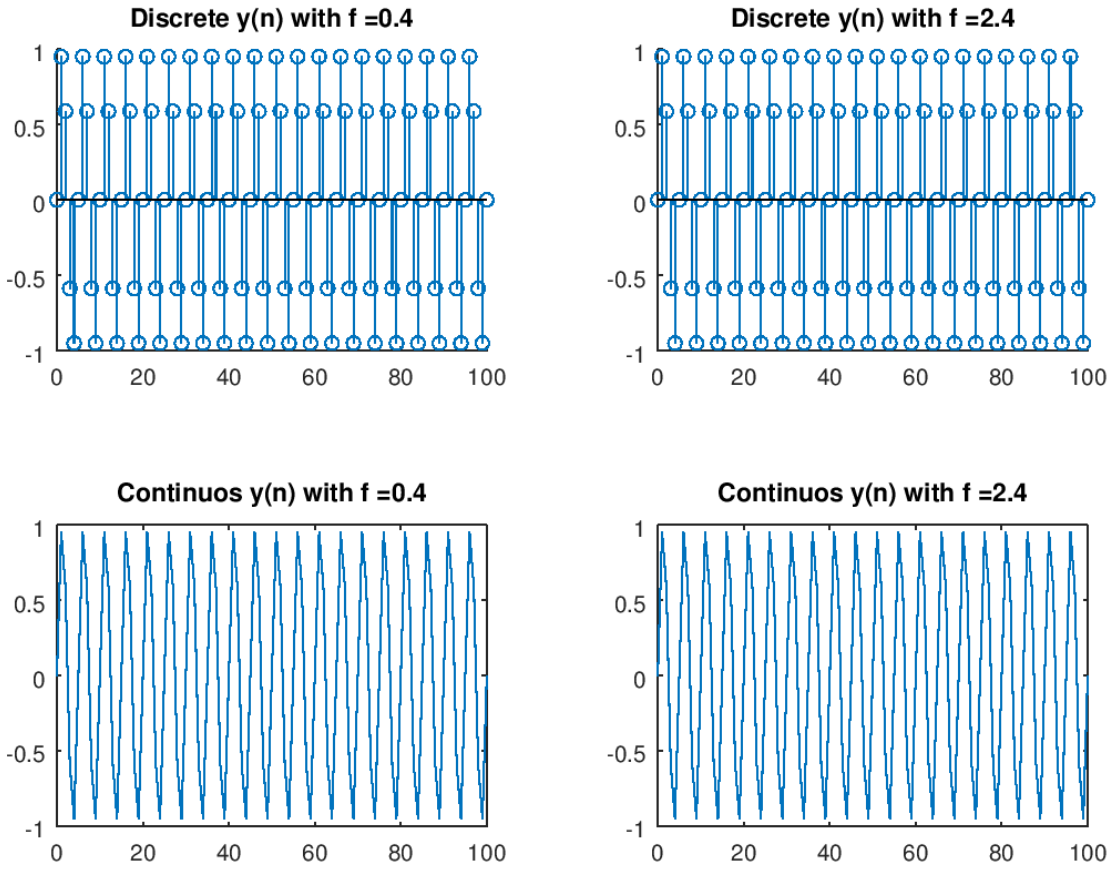


Imagen 3. Demostración frecuencias alias.

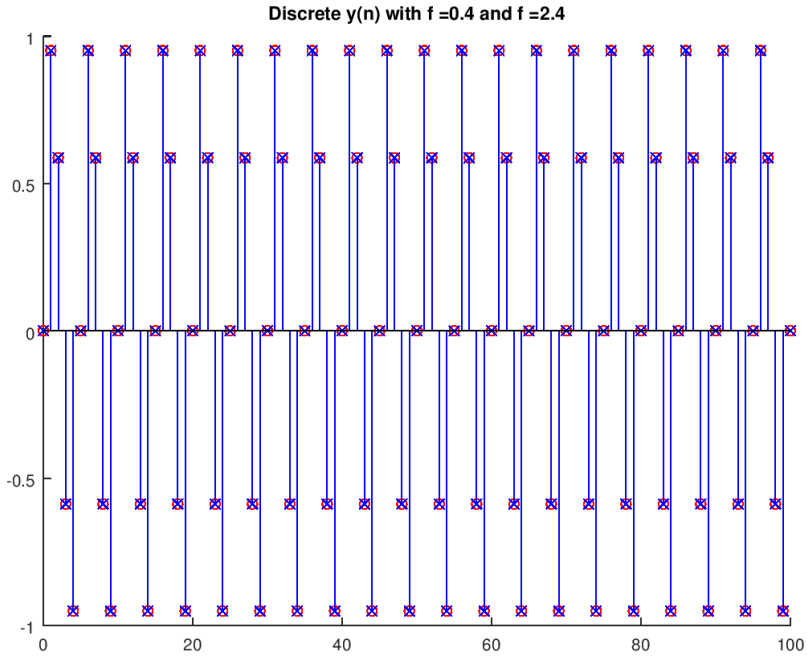


Gráfico 4. Función con frecuencia original y frecuencia alias superpuestas.

Ejercicio 4

Leer señales de audio

>> y = wavread (filename)

>> [y, fs, nbits] = wavread (filename)

>> […] = wavread (filename, n)

>> […] = wavread (filename, [n1 n2])

>> […] = wavread (…, datatype)

>> sz = wavread (filename, "size")

>> [n_samp, n_chan] = wavread (filename, "size")

"y" se convierte en el archivo indicado con "filename", "n" implica la cantidad de muestras, o "n1" y "n2" el inicio y final de las muestras. Si se envía el parámetro "size" se devuelve el tamaño del archivo y no las muestras. "fs" indica la frecuencia de muestreo del archivo de audio, y "nbits" indica la cantidad de bits por muestra"

Guardar una señal de audio

>> wavwrite (y, filename)

>> wavwrite (y, fs, filename)

>> wavwrite (y, fs, nbits, filename)

Escribe la señal de audio "y" en el archivo "filename". "fs" indica la frecuencia de muestreo del archivo de audio, y "nbits" indica la cantidad de bits por muestra". Los valores por defecto son: "fs" = 8000 Hz y "nbits" = 16 bits por muestra.

Reproducir una señal de audio

>> sound (y)

>> sound (y, fs)

>> sound (y, fs, nbits)

Reproduce la señal de audio "y" en el dispositivo de audio por defecto.

"fs" indica la frecuencia de muestreo del archivo de audio, y "nbits" indica la cantidad de bits por muestra". Los valores por defecto son: "fs" = 8000 Hz y "nbits" = 8 bits por muestra.

Ejemplo

>> y = wavread("bongos1.wav")

>> sound(y, 44100)

>> plot(y)

En este ejemplo se lee el archivo “bongos1.wav”, que está incluido en la solución, y se envía a reproducir a una frecuencia de muestreo de 44100. En caso de que no se indique la frecuencia, el sonido se reproduce pero suena “más lento”. En la figura xx podemos ver el resultado de mostrar la señal de audio de bongos1.wav.

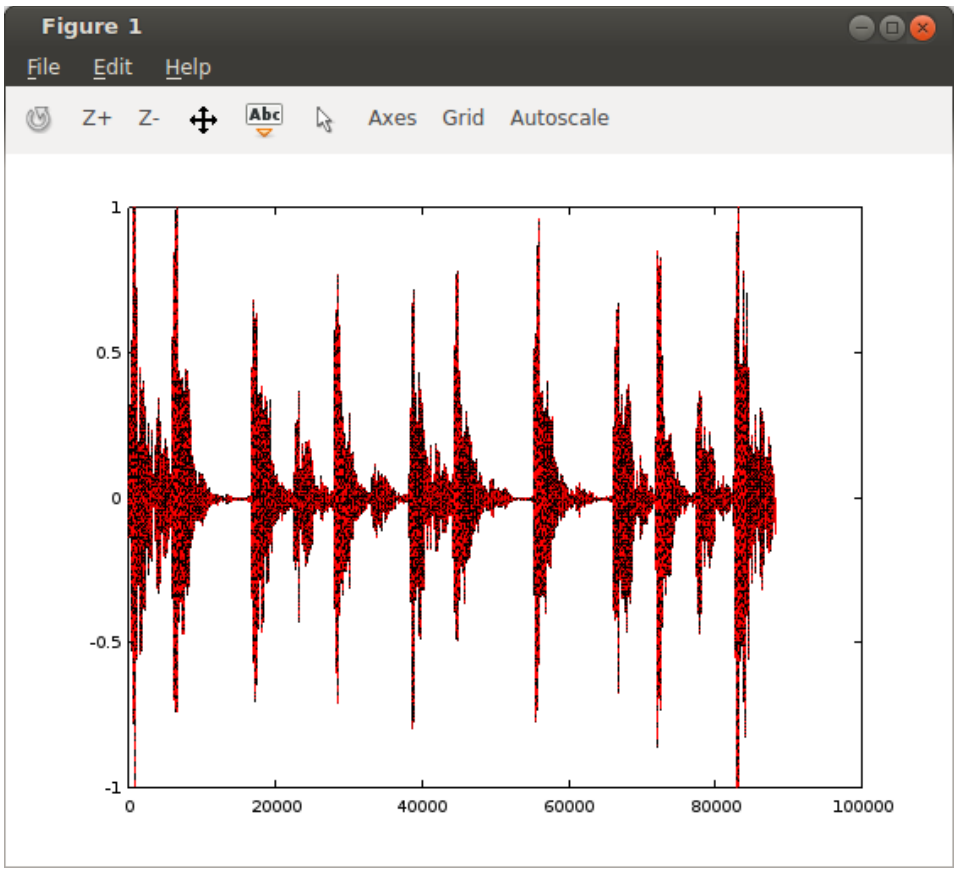


Imagen xx. Señal de audio “bongos1.wav”.

Ejercicio 5

Para la solución de este ejercicio se definirá la frecuencia de muestreo del convertidor analógico-digital y digital-analógico, para efectos de este documento se establecerá en $44{,}1\,KHz$ (44100 muestras por segundo).

Posteriormente definiremos el vector que contendrá los elementos del dominio de la función, este estará dado por todos los números enteros desde 0 hasta la frecuencia de muestreo multiplicado por la cantidad de segundos de reproducción que se desea, ya que la cantidad de muestras por segundo está definido por la frecuencia de muestreo.

Para definir la frecuencia de la señal contruida, se utilizará el siguiente resultado (notas de clase, “Equivalencia de frecuencias entre sinusoides continuos y discretos”, página 24):

$$f = \frac{F}{F_s}$$

donde f se conoce como frecuencia normalizada. Luego, se construirá la función senoidal como sigue

$$y(n) = \sin(2\pi f n)$$

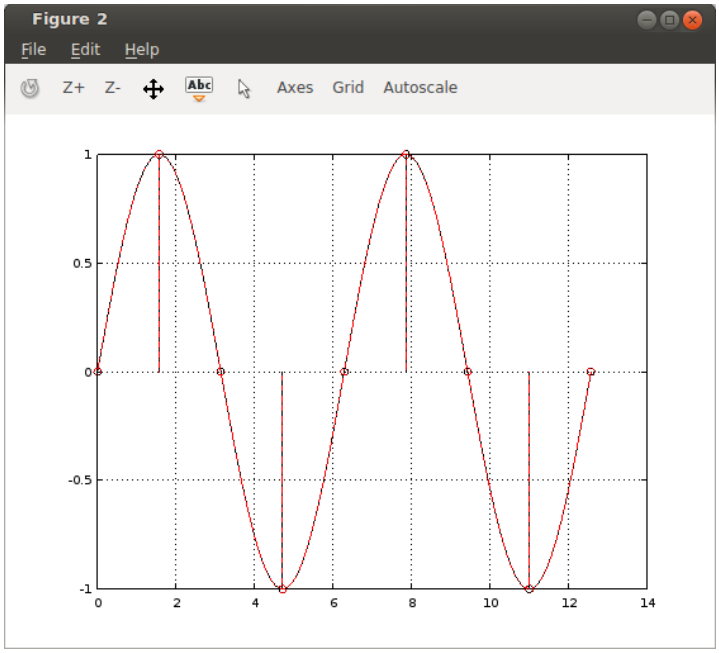
donde f es la frecuencia normalizada calculada anteriormente. Ahora se utilizará la función estudiada en el ejercicio anterior para reproducir la señal sintetizada.

Ejercicio 6

Para este ejercicio, se creó una función llamada show, que recibe la frecuencia y la cantidad de muestras por periodo. La función tiene como salida una gráfica que muestra dos periodos de la señal senoidal a la frecuencia ingresada, y las muestras tomadas.

Ejemplo

>> show(1,4)



>> show(10,5)

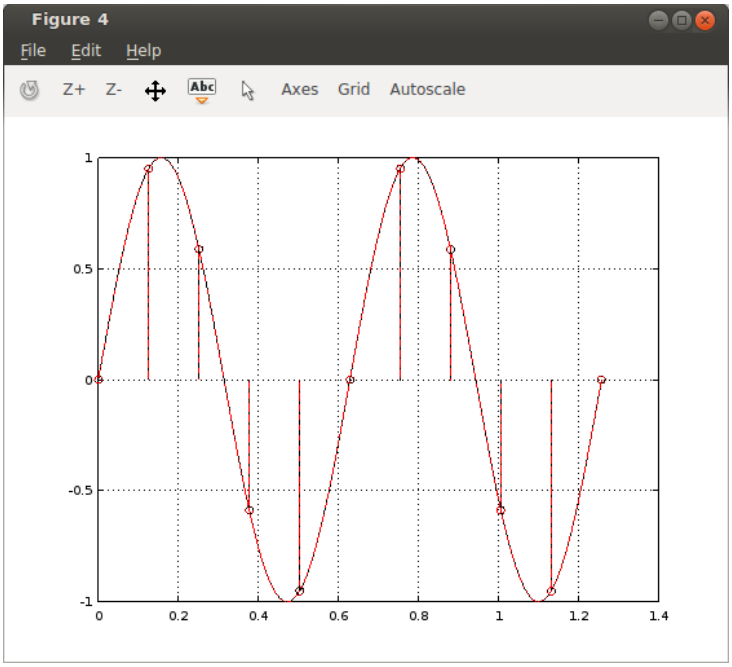


Figure 4
File Edit Help
Z+ Z- Axes Grid Autoscale
1
0.5
0
-0.5
-1
0 0.2 0.4 0.6 0.8 1 1.2 1.4

>> show (30,10)

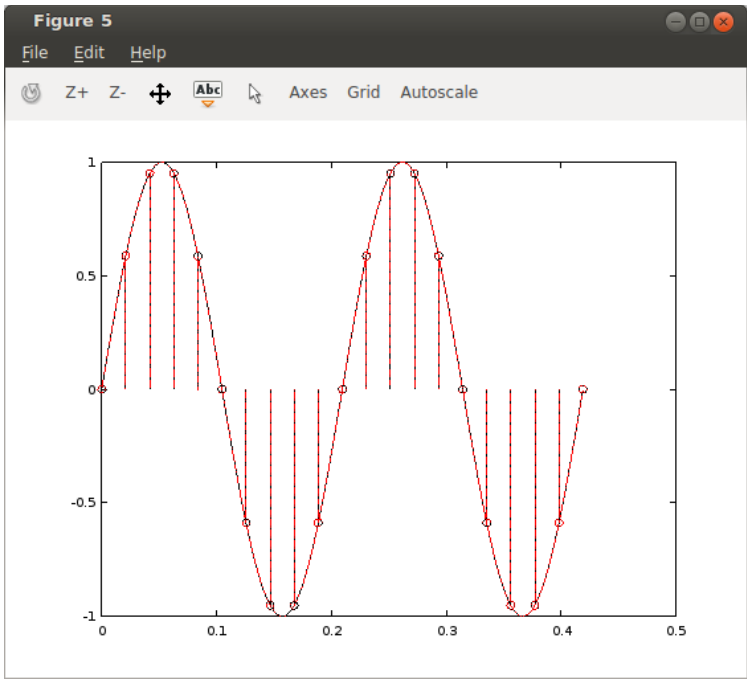


Figure 5
File Edit Help
Z+ Z- Axes Grid Autoscale
1
0.5
0
-0.5
-1
0 0.1 0.2 0.3 0.4 0.5

Ejercicio #7

En este ejericio se generará primero la señal de prueba, para el demo la señal será la misma del documento de notas del curso: $x(n)=\{0,1,2\}$. Luego, se llamará a la función main con la señal de prueba como parámetro.

El algoritmo para resolver el problema es el siguiente:

- Calcular el tamaño de la señal (Vector, una fila, n columnas).
- Generar un dominio tentativo, el cual será una secuencia de números enteros desde 0 hasta n – 1 . (Por ejemplo, para la señal $x(n)=\{0,1,2\}$, el dominio es $\{0,1,2\}$, con $n=3$.
- Generar una “señal normalizada”, consiste en dos secuencias unidas, la primera es un vector de ceros, de tamaño n – 1 y luego la señal original. Esto para preparar el camino a la inversión.
- Invertir la señal, para esto se utilizó la función fliplr, luego se añaden ceros (de la misma forma que el punto anterior) pero después de la señal, no antes.
- Generar el dominio definitivo, el cual es $D=[-(n-1),n-1]$, donde todo $k\in D, k\in\mathbb{Z}$.
- Teniendo la señal original, la señal invertida y el dominio, se procede a calcular las componentes par e impar.
- Se graficará lo requerido:
  - Señal original
  - Suma de las componentes par e impar.
  - Componente Par
  - Componente Impar

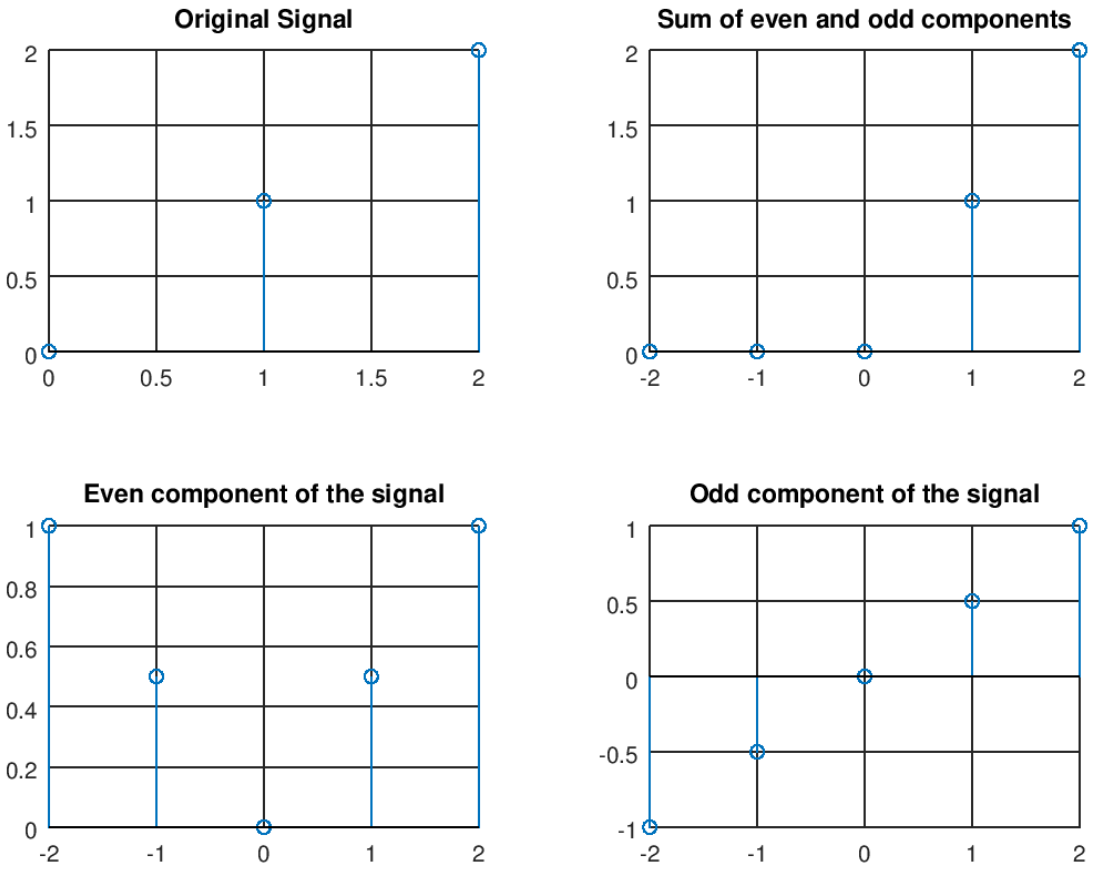


Imagen 4. Gráficos Señal original, Suma de las componentes par e impar, Componente Par, Componente Impar

Ecuación composición de señales: $x(n)=x_p(n)+x_i(n)$.

Ecuación componente par $x_p(n)=\frac{x(n)+x(-n)}{2}$.

Ecuación componente impar $x_i(n)=\frac{x(n)-x(-n)}{2}$.