UNIVERSIDAD MICHOACANA DE SAN NICOLAS DE HIDALGO
Facultad de Ingeniería Eléctrica

Laboratorio de Procesamiento Digital de Señales

PRÁCTICA 1

"Graficación de Señales Sinusoidales, y el Problema de Aliasing "

Objetivo

Aprender a graficar señales senoidales con las escalas horizontal y vertical adecuadas y observar la importancia del número de muestras por periodo y el fenómeno de traslape o aliasing.

ANTECEDENTES

Señales

Una señal es una función de una o varias variables. Si se clasifican según los valores que pueden tomar dichas variables (*dominio*); las señales se clasifican en dos grupos: señales analógicas y señales discretas.

Las señales analógicas son función de una o varias variables reales (continuas), y las señales discretas o *secuencias* son función de una o varias variables que únicamente puede tomar valores enteros.

Ejemplos de señales de dominio continuo son la señal de voz (presión en función del tiempo), imagen (brillo en función de la posición x,y), o la temperatura en función de la altura. Ejemplos de señales de dominio discreto son el stock de un almacén (cantidad en función del elemento), el índice de la bolsa (valor en función del día) y todas las señales que son resultado de tomar muestras o valores de señales de dominio continuo en instantes determinados.

El *rango* de una señal es el conjunto de valores que puede tomar la señal en su dominio. El rango es continuo si está formado por uno o varios intervalos reales, o discreto cuando la señal toma un valor de entre los de un conjunto previamente establecido. Ejemplos de

señales de rango continuo son la temperatura, la tensión o el brillo de una imagen; ejemplos de señales de rango discreto son la población, el número de coches (sólo pueden tomar valores enteros) y las que se obtienen como resultado de un proceso de cuantificación (representación de una señal por medio de N valores distintos): una imagen donde sólo se permite una escala de grises determinada para su representación, una señal almacenada en una memoria con un número de bits determinado, etc.
En la figura se ofrece una muestra de cada uno de los cuatro tipos de señales que se distinguen en el conjunto de señales de acuerdo con su rango y su dominio.

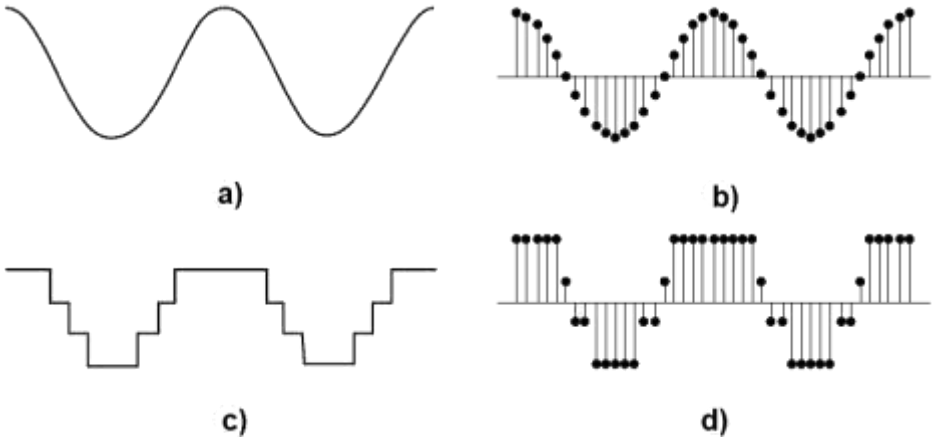


Figura 1: Tipos de señales. a) Amplitud y tiempo continuos. b) Amplitud continua, tiempo discreto. c) Amplitud discreta tiempo continuo. d) Amplitud y tiempo discretos.

Secuencias

Una secuencia es una señal que depende de una variable discreta. También puede definirse como un conjunto ordenado de números o valores. Una secuencia *"x[n]"* se representa en función de una variable "*n*" que únicamente toma valores enteros. Una forma habitual de definir una secuencia es la enumeración de los valores de sus elementos como se ejemplifica a continuación:

$$x[n] = \{0,1,4,4,3,2,0,-1,-2,-4,0,...,0\} \quad (0.1)$$

Una forma de generar una secuencia, es tomar muestras equiespaciadas de una señal analógica dependiente del tiempo; por lo que a los valores de la secuencia se denominen comúnmente como *muestras*. Independientemente de si la secuencia ha sido generada por medio de un muestreo o no, el eje de abscisas se conoce como "eje temporal o de tiempo" por lo que los valores concretos de la variable "*n*" reciben el nombre de "instantes". Se define "*Longitud o Duración de una secuencia*" al número de muestras contenidas en el intervalo que recoge todas las muestras, y normalmente se representa por una letra "N" mayúscula. Cuando el número de muestras no nulas de la secuencia es finito, se dice que la secuencia es de duración finita; en caso contrario, diremos que la duración de la secuencia es infinita.

DESARROLLO

1.- Usando SciLab, Octave o MATLAB, genere un vector de tiempo *"t"* que vaya de 0 a 0.5 en incrementos de 0.01.

2.- Calcule un vector *"y"* utilizando la expresión $y(t) = \sin(\omega t)$ donde $\omega = 2\pi F$ para cada una de las siguientes frecuencias:

   a) F_1= 1Hz.

   b) F_2 = 2Hz.

   c) F_3 = 4.5Hz.

   d) F_4 = 10Hz.

   e) F_5 = 50Hz.

   f) F_6 = 104.5Hz

3.- Grafique de forma independiente cada uno de los vectores resultantes asegurándose que la escala mostrada en el eje de las abscisas (eje x) este graduado conforme al vector *"t"* calculado en el inciso 1.

4.- Observe y asegúrese que las señales representadas se correspondan con sus respectivas frecuencias. En caso de no ser así, explique con sus propias palabras la posible causa.

5.- Grafique las señales de frecuencia *f*3 y *f*6 juntas, y compárelas visualmente. Escriba sus observaciones y trate de explicar con sus propias palabras el fenómeno observado.

6.- Compare numéricamente las dos señales anteriores y determine si se tratan de la misma señal, y explique el porqué.

Reportar

- Gráficas y códigos utilizados para obtener las gráficas. El código deberá estar debidamente comentado.
- Observaciones y explicaciones en sus propias palabras de todos los fenómenos observados en cada punto.
- Conclusiones.

ANEXO 1

Manejo de Secuencias en la PC (Uso de MATLAB)

MATLAB es el nombre abreviado de "MATrix LABoratory". MATLAB es un programa para realizar cálculos numéricos con vectores y matrices. Como caso particular puede también trabajar con números escalares (tanto reales como complejos), con cadenas de caracteres y con otras estructuras de información más complejas. Una de las capacidades más atractivas es la de realizar una amplia variedad de gráficos en dos y tres dimensiones. MATLAB tiene también un lenguaje de programación propio.

Definición de Vectores y Matrices.

Como en casi todos los lenguajes de programación, en MATLAB las matrices y vectores son variable*s* que tienen nombres, y normalmente se sugiere que se utilicen letras mayúsculas para matrices y letras minúsculas para vectores y escalares (MATLAB no exige esto, pero puede resultar útil). Para definir una matriz no hace falta declararlas o establecer de antemano su tamaño (de hecho, se puede definir un tamaño y cambiarlo posteriormente). MATLAB determina el número de filas y de columnas en función del número de elementos que se proporcionan (o se utilizan). Las matrices se definen o introducen por filas; los elementos de una misma fila están separados por espacios o comas, mientras que las filas están separadas por pulsaciones de la tecla *intro* o por caracteres punto y coma. Por ejemplo los comandos siguientes definen una misma matriz de 3 x 2:

```
>> A = [1   2 ; 2   3 ; 4   5]
```

ó

```
>>  A  = [1, 2 ; 2, 3 ; 4, 5]
```

La respuesta del programa será:

```
A  =
      1   2
      2   3
      4   5
```

A partir de este momento la matriz A está disponible para hacer cualquier tipo de operación con ella (además de valores numéricos, en la definición de una matriz o vector se pueden utilizar expresiones y funciones matemáticas). En MATLAB se accede a los elementos de un vector poniendo el índice entre paréntesis (por ejemplo x(3) ó x(i)). Los elementos de las matrices se acceden poniendo los dos índices entre paréntesis, separados por una coma (por ejemplo A(1,2) ó A(i,j)). Las matrices se almacenan por columnas (aunque se introduzcan por filas), y teniendo en cuenta esto puede accederse a cualquier elemento de una matriz con un sólo subíndice. Por ejemplo, si B es una matriz (3×3) se obtiene el mismo valor escribiendo B(1,2) que escribiendo B(4).

De forma análoga a las matrices, es posible definir un vector fila x en la forma siguiente (si los tres números están separados por espacios o comas, el resultado será un vector fila):

```
>>   x=[10 20 30]     % vector fila
x =
       10   20   30
```

Por el contrario, si los números están separados por intros o puntos y coma (;) se obtendrá un vector columna:

```
>> y  =  [11; 12; 13]      % vector columna
y =
          11
          12
          13
```

MATLAB tiene en cuenta la diferencia entre vectores fila y vectores columna. Por ejemplo, si se intenta sumar los vectores x e y se obtendrá el siguiente mensaje de error:

```
>>  x+y
      ??? Error using ==> +
      Matrix dimensions must agree.
```

El problema se corrige si se suma x con el vector transpuesto de y:

```
>>  x + y'
ans =
          21   32   43
```

MATLAB considera vectores fila por defecto, como se ve en el ejemplo siguiente:

```
>> x(1)=1,  x(2)=2
x =
     1
x =
     1 2
```

Operadores Aritméticos

MATLAB puede operar con matrices y vectores por medio de operadores y por medio de funciones. Los operadores matriciales de MATLAB son los siguientes:

+	adición o suma
–	sustracción o resta
*	multiplicación
'	traspuesta
^	potenciación
\	división-izquierda
/	división-derecha
.*	producto elemento a elemento
./ y .\	división elemento a elemento
.^	elevar a una potencia elemento a elemento

Estos operadores se aplican también a las variables o valores escalares, aunque con algunas diferencias. Todos estos operadores son coherentes con las correspondientes

operaciones matriciales: no se puede por ejemplo sumar matrices que no sean del mismo tamaño. Si los operadores no se usan de modo correcto se obtiene un mensaje de error. Los operadores anteriores se pueden aplicar también de modo mixto, es decir con un operando escalar y otro matricial. En este caso la operación con el escalar se aplica a cada uno de los elementos de la matriz.

Operador Dos Puntos (:)

Este operador es muy importante en MATLAB y puede usarse de varias formas. Para empezar, defínase un vector x con el siguiente comando:

```
>> x=1:10
X  = 1    2    3    4    5    6    7    8    9    10
```

Se podría decir que el operador ":" representa un rango, en este caso, los números enteros entre el 1 y el 10. Por defecto el incremento es 1, pero este operador puede también utilizarse con otros valores enteros y reales, positivos o negativos. En este caso el incremento va entre el valor inferior y el superior, en las formas que se muestran a continuación:

```
>> x = 1 : 2 : 10
x   =  1    3    5    7    9
>> t = 0:0.2:1
t =  0    0.2000    0.4000    0.6000    0.8000    1.0000

>> x =  5 : -1 : 1
x =  5    4    3    2    1
```

Puede verse que, por defecto, este operador produce vectores fila. Si se desea obtener un vector columna, basta trasponer el resultado.

El operador dos puntos ":" es aún más útil y potente (y también más complicado). Analícese el siguiente ejemplo:

```
>> A=magic(5)
A =
   17    24     1     8    15
   23     5     7    14    16
    4     6    13    20    22
   10    12    19    21     3
   11    18    25     2     9
```

Recuérdese que MATLAB accede a los elementos de una matriz por medio de los índices de fila y de columna encerrados entre paréntesis y separados por una coma. Por ejemplo:

```
>> A(3,4)
ans  =  20
```

El siguiente comando extrae los últimos 3 elementos de la 1ª fila:

```
>> A(1, 3:5)
ans  =     1     8    15
```

Los dos puntos aislados representan "todos los elementos". Por ejemplo, el siguiente comando extrae todos los elementos de la 4ª fila:

```
>> A(4 , : )
ans =   10   12   19   21    3
```

Para acceder a la última fila o columna puede utilizarse la palabra *"end"*, en lugar del número correspondiente. Por ejemplo, para extraer la quinta fila (la última) de la matriz:

```
>> A (end,  : )
ans =   11   18   25    2    9
```

El siguiente comando extrae todos los elementos de las filas 3, 4 y 5:

```
>> A(3:5,:)
ans =
    4    6   13   20   22
   10   12   19   21    3
   11   18   25    2    9
```

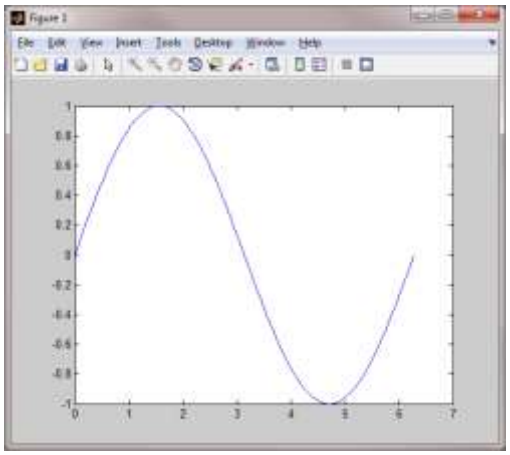
Funciones en Matlab

MATLAB tiene un gran número de funciones incorporadas. Algunas son funciones intrínsecas, esto es, funciones incorporadas en el propio código ejecutable del programa. Estas funciones son particularmente rápidas y eficientes, además existen funciones definidas en ficheros "*.m" y "*.mex" que vienen con el propio programa o que han sido aportadas por usuarios del mismo. Estas funciones extienden en gran manera la potencia y flexibilidad del programa.

Funciones matemáticas elementales que operan de modo escalar

Comprenden las funciones matemáticas trascendentales y otras funciones básicas, cuando se aplican a una matriz actúan sobre cada elemento de la matriz como si se tratase de un escalar. Por tanto, se aplican de la misma forma a escalares, vectores y matrices. Algunas de las funciones de este grupo son las siguientes: sin(x) seno, cos(x) coseno, tan(x) tangente, asin(x) arco seno, acos(x) arco coseno, atan(x) arco tangente (devuelve un ángulo entre -π/2 y +π/2)…… Para una lista completa de funciones, consúltese la ayuda en Matlab. Ejecutando los siguientes comandos podemos observar el uso de la función seno, y la función plot.

```
>> x  =  0 :  pi/30  : 2*pi;
>> y  =  sin(x);
>> plot(x,y)
```



Gráficos Bidimensionales

Los gráficos 2-D de MATLAB estén fundamentalmente orientados a la representación gráfica de vectores (y matrices). En el caso más sencillo los argumentos básicos de la función plot van a ser vectores. Cuando una matriz aparezca como argumento, se considerará como un conjunto de vectores columna (en algunos casos también de vectores fila). MATLAB utiliza un tipo especial de ventanas para realizar las operaciones gráficas. Ciertos comandos abren una ventana nueva y otros dibujan sobre la ventana activa, bien sustituyendo lo que hubiera en ella, bien añadiendo nuevos elementos gráficos a un dibujo anterior.

Funciones gráficas 2D elementales

MATLAB dispone de cinco funciones básicas para crear gráficos 2-D. Estas funciones se diferencian principalmente por el tipo de escala que utilizan en los ejes de abscisas y de ordenadas. Estas cuatro funciones son las siguientes:

plot()	Crea un gráfico a partir de vectores y/o columnas de matrices, con escalas linealessobre ambos ejes
plotyy()	Dibuja dos funciones con dos escalas diferentes para las ordenadas, una a la derechay otra a la izquierda de la figura.
loglog()	Similar al anterior pero con escala logarítmica en ambos ejes.
semilogx()	Similar al anterior pero con escala lineal en el eje de ordenadas y logarítmica en el eje de abscisas.
semilogy()	Similar al anterior pero con escala lineal en el eje de abscisas y logarítmica en el eje de ordenadas.

Existen además otras funciones orientadas a añadir títulos al gráfico, a cada uno de los ejes, a dibujar una cuadrícula auxiliar, a introducir texto, etc. Estas funciones son las siguientes:

title('título')	Añade un título al dibujo.
xlabel(‘texto’)	Añade una etiqueta al eje de abscisas. Con xlabel off desaparece.
ylabel(‘ texto ‘)	Añade una etiqueta al eje de ordenadas. Con ylabel off desaparece.

text(x,y,'texto')	Introduce 'texto' en el lugar especificado por las coordenadas x e y. Si x e y son vectores, el texto se repite por cada par de elementos. Si texto es también un vector de cadenas de texto de la misma dimensión, cada elemento se escribe en las coordenadas correspondientes.
gtext('texto')	Introduce texto con ayuda del ratón: el cursor cambia de forma y se espera un click para introducir el texto en esa posición.
legend()	Define rótulos para las distintas líneas o ejes utilizados en la figura. Para más detalle, consultar el Help.
grid	Activa la inclusión de una cuadrícula en el dibujo. Con grid off desaparece la cuadrícula.

Guardar y Recuperar Variables

En algunas ocasiones es necesario interrumpir el trabajo, pero con la posibilidad de poderlo recuperar en el mismo punto en el que se dejó (con las mismas variables, con los mismos resultados intermedios, etc.). Para tal efecto, MATLAB dispone del comando "*save*". Si se teclea dicho comando antes de cerrar el programa, se crea en el directorio de trabajo actual, un archivo llamado "matlab.mat" con el estado de la sesión. Dicha sesión se puede recuperar con el comando *"load"*. Este par de comandos pueden utilizarse también para guardar y recuperar la sesión en archivos con nombres personalizados, por ejemplo:

>> save *nombrearchivo*

El comando anterior guardará todas las variables del espacio de trabajo en un archivo de nombre "nombrearchivo.mat". Para recuperar los datos bastará con teclear

>> load *nombrearchivo*