MATRIX LABORATORY

MATLAB

APUNTES DE MATLAB DEL
PROF. BALEBONA JENRY
PARA LOS ALUMNOS DEL
INSTITUTO UNIVERSITARIO DE TECNOLOGÍA CUMANÁ

Cumaná abril de 2005

1 Antes de iniciar los cálculos

1.1. ¿ Cómo abrir MATLAB ?

Si su computadora (PC) tiene instalado MATLAB esta deberá presentar en, el escritorio de Windows, un ícono como el de la figura No.1. si por el contrario no aparece, pero sabe que esta instalado el software del MATLAB, recurra a inicio de Windows y siga las siguientes instrucciones:

FIG.No.1

Inicio → Programa → Matlab → Matlab

Una vez que halla entrado al editor de programa encontrará un "prompt" (**»**) como indicador, el cual le invitará a introducir los comandos del Matlab. Estos comandos a que se hace referencia es el objetivo del curso. Mientras más comandos conozcamos más entenderemos el mundo de Matlab.

1.2. ¿ Cómo salir de MATLAB ?

Existen cuatro formas de salir de Matlab, que son las siguientes:

- ✓ A través del menú File → Exit Matlab
- ✓ Utilizando Ctrl+Q.
- ✓ Utilizando la ventana de comandos (Command Window) y el prompt **» quit** ↵

  Nota: ↵ = Teclear la Tecla ENTER
- ✓ Por último, también se puede hacer "click" en la parte superior izquierda de la pantalla, donde aparece el siguiente ícono.

1.3. ¿ Cómo solicitar ayuda a MATLAB ?

Si no entiende bien el significado de un comando, teclee **help** y el nombre del comando en cuestión. El comando **help** presenta una síntesis de los comandos muy precisa. Por ejemplo podemos empezar por solicitar el primer comando visto hasta ahora "**quit**", para ello nos valemos del "prompt" y escribimos:

» help quit ¿

a lo cual el presentará la siguiente respuesta:

```
QUIT Quit MATLAB session.
    QUIT terminates MATLAB without saving the
workspace.  To save
    your workspace variables, use SAVE before
quitting.
```

1.4. ¿ Tipos de archivos que puedes encontrar en MATLAB ?

Los diferentes archivos los cuales produce y se alimenta el software del Matlab, y que pueden ser identificados por su extensión, son los siguientes:

.M : Archivos de función o guión.
.MAT : Archivos que contienen datos en binario.
.MET : Archivos ejecutables compilados en Fortran o en "C".
.MDL : Archivos de Simulink♠.

♠**Simulink**: Interfase del Matlab usado para simular sistemas dinámicos. Se basa en sistemas gráficos de bloques.

1.5. ¿ Algunos comandos iniciales ?

Existen comandos los cuales se pueden catalogar como el souvenir de las instrucciones, esto porque son de carácter informativo más que de calculo, como tal. Entre ellos los siguientes:

a) **Versión**: Lo primero que un usuario debe saber es la versión del software con la cual trabaja, para ello teclee:
   » version ¿ (Observe que no lleva acento en ingles)
b) **Demo**: Si quiere hacer turismo lo invito a seguir la demostración (Demo) del Matlab. **» demo** ¿
c) **Qué**: El comando **what** produce una lista de archivos .M, .MAT, Y . MEX presentes en directorio en uso.
d) **Quién**: El comando **who** produce una lista de variables del espacio de trabajo en uso .
e) **Camino**: El comando **path** presenta la ruta de búsqueda vigente del Matlab.
f) **Directorio actual**: con el comando **dir** se muestra los archivos contenidos en el directorio actual.
g) **Diario**: El comando **diary on** grava todo lo que se introduce por teclado, así como todo lo que se imprime en pantalla como resultado a un archivo con el nombre de *diary* si no es especificado el archivo. Si quieres gravar en otro archivo deberás indicárselo de esta forma diary('nombredelarchivo').
   Para dejar de gravar se utiliza **diary off.**
   El útil para recuperar con algun editor de texto el trabajo realizado.
h) **Reloj**: El comando **clock** exhibe números como:

```
ans =
  1.0e+003 *
    2.0030    0.0010    0.0110    0.0060    0.0410    0.0512
```

   el primer número, 1.0e+003, es un multiplicador. La siguiente linea tiene el siguientes significado:

AÑO MES DIA HORA MINUTOS SEGUNDOS.

Lo que indica que la fecha y hora ES: AÑO 2003 el primer mes, el día 11, hora 6 am, 51 segundos.

i) **Escape**:
j) **Fecha**: el comando **date**, genera la fecha, por ejemplo:
» date
ans =
11-Jan-2003
k) **Comentarios**: Podemos ilustrar los programas si les adicionamos comentarios, para ello utilizaremos "%" antes del comentario que quiera agregar, ejemplo:

» quit ¿ **%** "comando para salir de Matlab"

l) **El Punto y Coma**: un punto y coma (**;**) después de un comando, evita que este produzca una respuesta.
m) **Epsilon "e" de la máquina**: si deseas utilizar una variable muy pequeña para realizar cálculos en tu programa, Matlab te ofrece **eps**:
» eps
ans =
2.2204e-016
n) **Limpiar pantalla**: Para limpiar pantalla se utiliza el comando **clc**.
o) **Limpiar variables**: Para limpiar variables se utiliza el comando **clear**, al utilizarla limpias todas las variables. Si quiere ser más especifico puedes limpiar algunas de las variables así:

clear x, y, z

1.6. ¿ Cómo grabar y cargar ?

El comando **save** guarda todas las variables en uso en forma binaria de doble precisión o en forma ASCII agregando **–ascii** o **/ascii**, por defecto en un archivo llamado matlab.mat. Si se especifica, las variables, pueden ser guardadas en un archivo nombrado por usted, como sigue

save nombredelarchivo
save nombredelarchivo **–ascii**

y si deseas gravar algunas variables en particular:

save nombredelarchivo x,y,z
save nombredelarchivo x,y,z **–ascii**

Para cargar el valor de las variables se utiliza el comando **load**. Ejemplo:

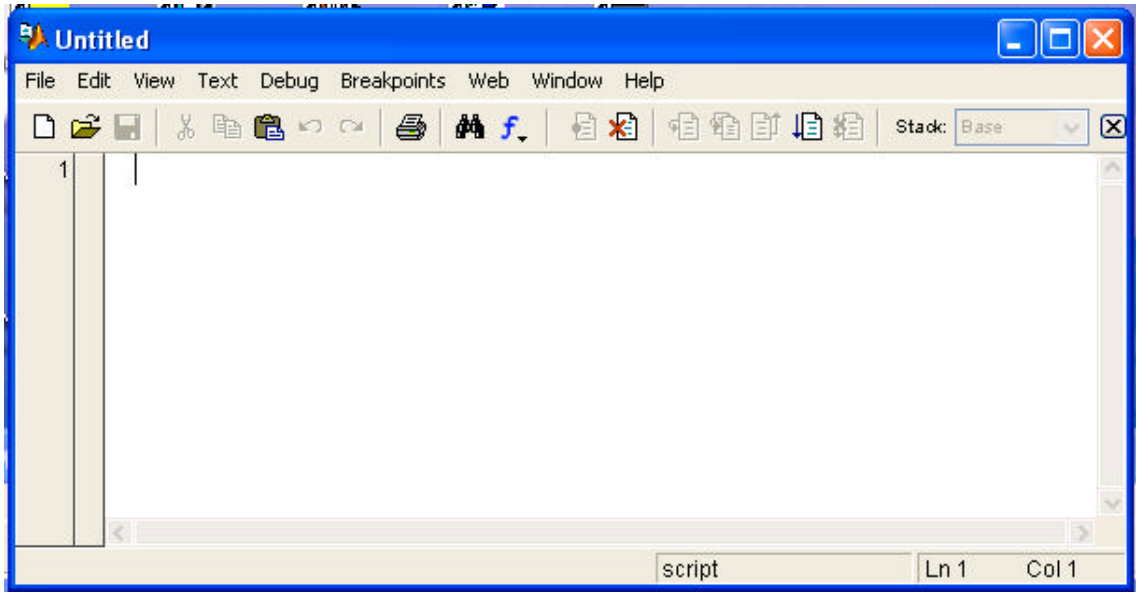
load nombredelarchivo
load nombredelarchivo x,y,z

1.7. ¿ Cómo utilizar el comando EDIT ?

EDIT es un editor de texto del Matlab, este sirve para escribir y guardar los programas en Matlab.

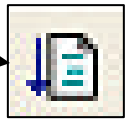
Para llamarlo solo debes escribir en el prompt de la ventana de comandos (Command Window) >>**edit** ¿, o a través del icono

del (Command Window) donde aparecerá una ventana como la que sigue con el nombre de Untitled (sin titulo).



Una vez escrito el programa procede a guardarlo, eso si, recuerda el nombre con el cual a sido guardado, ya que con el nombre puedes hacer que corra el programa solamente escribiendo el mismo en el Prompt de la de la ventana de comandos (Command Window). También es útil el comando copiar-pegar del Windows, copia cualquier programa que se te suministre y pégalo en el editor.

También puedes correr el programa a través del **icono** que se encuentra en la parte superior de la ventana **edit** otra alternativa sería la tecla F5 o con el ratón ir a Debug → Run.



2 FUNCIONES Y OPERACIONES MATEMATICAS

2.1. Operadores algebraicos básicos (SIEMPRE EN MINUSCULAS)

+	=	suma.	(X**+**Y)
-	=	resta.	(X**-**Y)
*****	=	multiplicación.	(X*****Y)
/	=	división.	(X**/**Y)
****	**=**	inversa	(8****3=? *3)
^	=	elevar a la potencia	(3**^**5=3^5).
.+	=	suma en arreglos.	(X**.+**Y)
.-	=	resta en arreglos.	(X**.-**Y)
.*	=	multiplicación en arreglos.	(X**.***Y)
./	=	división en arreglos.	(X**./**Y=X.**inv**(Y))
.^	=	elevar en arreglos	(3**.^**5=3^5).
inv	**=**	inversa.	(**inv**(Y))
sqrt	=	raíz cuadrada √.	(**sqrt**(2)=1.4142)
log	=	Logaritmo natural, base **e**.	(**log**(2)=0.6931)
log10	=	Logaritmo en base 10.	(**log10**(2)=0.3010)
abs	=	Valor absoluto.	(**abs**(-2)=2)
real	=	parte real de un complejo.	(**real**(3+4i)=3)
imag	=	parte real de un complejo.	(**imag**(3+4i)=4)
conj	=	conjugado de un complejo.	(**conj**(3+4i)= 3-4i)
angle	=	ángulo de fase de un valor complejo.	(**angle**(a+bi))
det	=	determinante de una matriz.	(**det**(A)=34)
eig	=	vector autovalores.	(**eig**(matriz cuadrada))
trace	=	traza de una matriz.	(**trace**(A))
norm	=	norma de una matriz.	(**norm**(A))
rank	=	rango de una matriz.	(**rank**(A))
kron	=	producto tensor Kronecker.	(**kron**(A,B))
fix	=	redondear un valor real hacia cero.	(**fix**(x))
rem	=	residuo de una división.	(**rem**(9,2)=1)

2.2. Constantes básicos

pi	=	La constante ?.	(**pi**=3.1416)
eps	=	Épsilon de la máquina.	(**eps**=2.2204e-16)
inf	=	Infinito.	(**inf**=∞)
i y **j**	=	Unidad imaginaria $\sqrt{-1}$.	
NaN	**=**	No es número.	
flops	**=**	Contador de operaciones de punto flotante.	

NOTA	
	✓ Matlab reconoce, tanto la **i** como la **j**, minúsculas, como la unidad imaginaria. ✓ El multiplicar i*inf nos da como resultado NaN+Infi, lo cual indica que no es número.

2.3. Operadores Trigonométricos Básicos (SIEMPRE EN MINUSCULAS)

Los argumentos asociados a grados se expresan en radianes.

sin(x) =	seno de x.	(**sin**(**pi**)=eps)
cos(x) =	coseno de x.	(**cos**(**pi**)=-1)
tan(x) =	tangente de x.	(**tan**(**pi**)=eps)
asin(x) =	inversa del seno de x.	(**asin**(1)= 1.5708)
acos(x) =	inversa del cos de x.	(**acos**(0.5)=1.0472)
atan(x) =	inversa de la tangente de x.	(**atan**(0.5)=0.4636)
atan2(y,x) =	inversa de la tangente IV cuadrante.	
sinh(x) =	seno hiperbólico de x.	(**sinh**(pi)= 11.5487)
cosh(x) =	coseno hiperbólico de x.	(**cosh**(pi)= 11.5920)
tanh(x) =	tangente hiperbólica de x.	(**tanh** (pi)= 0.9963)
asinh(x) =	inversa del seno hiperbólico de x.	
acosh(x) =	inversa del coseno hiperbólico de x.	
atanh(x) =	inversa del tangente hiperbólica de x.	

2.4. Arreglo de Variables.

Los arreglos (array) se presentan dimensionalmente, es decir, podemos encontrar arreglos unidimensionales o bidimensionales.

- Los <u>arreglos unidimensionales</u> los podemos representar como vectores filas o vectores columnas:

$$\vec{A} = \begin{bmatrix} a_1 & a_2 & a_3 & \cdots & a_n \end{bmatrix} \qquad \vec{A} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_m \end{bmatrix}$$

Vector Fila — Vector Columna

en Matlab la representación de <u>vectores filas</u> es la siguiente:
Sea a el siguiente vector fila

$$\vec{A} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \end{bmatrix}$$

```
» A=[1 2 3 4 5] ¿
A =
    1     2     3     4     5
```

también puede representarse con comas después de las variables, tal como sigue:

```
» A=[1,2,3,4] ¿
A =
    1     2     3     4
```

en cuanto a la representación de <u>vectores en columnas</u> es la siguiente:

Sea el vector $\vec{A}$ un vector columna

$$\vec{A} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{bmatrix}$$

Matlab lo representara de esta forma:

```
» a=[1;2;3;4;5] ¿

a =
     1
     2
     3
     4
     5
```

Si se quiere transformar un vector fila en un vector columna podéis agregarle **un punto y un apóstrofe (.')** al final del vector fila y lo transformaras en vector columna o viceversa:

Dado el vector $\vec{A}$ estructurado en filas:

```
» A=([1 2 3 4 5]).'

A =
     1
     2
     3
     4
     5

» A=[1;2;3;4;5] .'¿
A =     1     2     3     4     5
```

Punto y Apóstrofe

Los arreglos bidimensionales los podemos representar como matrices:

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1m} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2m} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3m} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nm} \end{bmatrix}$$

La matriz es formada por vectores filas y vectores columnas, de ahí se partirá para producir una matriz.
Supóngase una matriz cuadrada de 3x3 como sigue:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

en Matlab se codificaría de la siguiente forma:

```
» A=[1 2 3;4 5 6;7 8 9] ¿

A =
     1     2     3
     4     5     6
     7     8     9
```

o también

```
» A=[1,2,3;4,5,6;7,8,9] ¿

A =
     1     2     3
     4     5     6
     7     8     9
```

2.5. ¿Cómo recuperar un elemento de una Matriz?.

Supóngase una matriz cuadrada de 3x3 como sigue:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

Luego como se dijo anteriormente en Matlab se codifica así:

» A=[1,2,3;4,5,6;7,8,9] ¿

A =

```
1     2     3
4     5     6
7     8     9
```

si quieres hallar el valor del elemento (3,3), es decir, fila 3 con columna 3 simplemente teclea lo siguiente:

» A(3,3) ¿

ans = 9

de igual forma si quieres hallar el valor del elemento (2,1), es decir, fila 2 con columna 1, haz lo siguiente:

» A(2,1) ¿

ans = 4

si se desea una columna la columna 1, en especial, proceda como sigue:

» A(:,1) ¿
ans =

```
1
4
7
```

Imaginemos que se quiere obtener las filas 1 y 2 y las columnas 1,2 y 3 de la matriz A, procedemos de esta manera:

» A(1:2,1:3) ¿

ans =

```
1    2    3
4    5    6
```

2.6. Álgebra matricial.

Antes de incurrir álgebra, propiamente, definamos ciertas matrices.
Una matriz cuya toda su diagonal es uno se define como <u>matriz identidad o matriz unidad</u>, por ejemplo:

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

en matlab se procede a representarla por el comando **eye**(x,x) como:

» **eye**(3,3) ¿

ans =

```
1    0    0
0    1    0
0    0    1
```

si se quiere obtener una matriz cuyos elementos son todo 1, se recurre al comando **ones** (x,y), de esta forma:

» **ones**(3,4) ¿

ans =

```
1    1    1    1
1    1    1    1
1    1    1    1
```

También podemos definir la matriz vacía o nula, la cual es aquella que no existe.

```
» V=[]  ¿
V =

    []
```

si se desea la diagonal de la matriz A, el comandado **diag**(A) nos puede ayudar, como sigue:

```
» diag(A) ¿
ans =

    1
    5
    7
```

El determinante de una Matriz

El determinante de una matriz lo podemos calcular a través del comando **det()**, ejemplo:

```
» A=[1 2 3 ;4 5 6 ;7 8 9]  ¿

A =

    1    2    3
    4    5    6
    7    8    9

» C=det(A) ¿

C =  0

» B=[-9 0 0 ;1 7 0 ;6 2 3]  ¿

B =

-9    0    0
    1    7    0
    6    2    3

» C=det(A)  ¿

C =  -189
```

Suma de Matrices

Supóngase dos matrices:

$$A = \begin{bmatrix} 1 & 4 & 2 \\ 3 & 5 & 8 \\ 9 & 6 & 7 \end{bmatrix} \qquad B = \begin{bmatrix} 2 & 1 & 6 \\ 0 & 3 & 0 \\ 3 & 7 & 7 \end{bmatrix}$$

su suma será:

$$A + B = \begin{bmatrix} 3 & 5 & 8 \\ 3 & 8 & 8 \\ 12 & 13 & 14 \end{bmatrix}$$

En Matlab, se define la matriz A y B y luego se solicita la suma A+B

```
» A=[1 4 2;3 5 8;9 6 7] ¿

A =

     1     4     2
     3     5     8
     9     6     7


» B=[2 1 6;0 3 0;3 7 7] ¿

B =

     2     1     6
     0     3     0
     3     7     7

» A+B

ans =

     3     5     8
     3     8     8
    12    13    14
```

Similar para la resta.

Multiplicación y división de Matrices

Note que para la multiplicación de matrices existe los operadores "*****" y "**.*** " , a su vez, para la división "**/**" y "**./**".
Cada uno tiene un propósito especifico los operadores "*****" y "**/** " realizan la multiplicación y la división de matrices siguiendo el álgebra matricial en cambio que los operadores que le antecede un punto "**.***" y "**./** " realizan la multiplicación y división de elementos en la misma localidad de dos matrices.
Un ejemplo a seguir para aclarar dudas es el siguiente:
Dado la Matriz A y B, como:

```
» A=[1,4,2;3,5,8;9,6,7] ¿
A =
    1     4     2
    3     5     8
    9     6     7

» B=[2 1 6;1 3 1;3 7 7] ¿

B =
    2     1     6
    1     3     1
    3     7     7

» C=A*B  ¿

C =

   12    27    24
   35    74    79
   45    76   109

» C=A.*B  ¿

C =
    2     4    12
    3    15     8
   27    42    49

» C=A/B  ¿

C =
```

```
   -0.5000   -0.2500    0.7500
    0.5000   -0.2500    0.7500
    7.3333   17.8333   -7.8333

» C=A./B  ¿

C =

    0.5000    4.0000    0.3333
    3.0000    1.6667    8.0000
    3.0000    0.8571    1.0000
```

Obsérvese que C=A/B es lo mismo que $C=A*B^{-1}$, donde B^{-1} es la inversa de la matriz B, EJEMPLO:

```
» C=A*inv(B)  ¿

C =

   -0.5000   -0.2500    0.7500
    0.5000   -0.2500    0.7500
    7.3333   17.8333   -7.8333
```

La inversa de una Matriz

Dado una matriz A,

$$A = \begin{bmatrix} 1 & 4 & 2 \\ 3 & 5 & 8 \\ 9 & 6 & 7 \end{bmatrix}$$

su inversa es:

$$A^{-1} = \begin{bmatrix} -0.0949 & -0.1168 & 0.1606 \\ 0.3723 & -0.0803 & -0.0146 \\ -0.1971 & 0.2190 & -0.0511 \end{bmatrix}$$

En Matlab se recurre al comando **inv**(x), tal como sigue:

» **inv**(A) ¿

```
ans =

   -0.0949   -0.1168    0.1606
    0.3723   -0.0803   -0.0146
   -0.1971    0.2190   -0.0511
```

desde luego $A.A^{-1}$ sería:

» A***inv**(A) ¿

```
ans =

    1.0000    0.0000    0.0000
    0.0000    1.0000    0.0000
    0.0000    0.0000    1.0000
```

<u>La transpuesta de una matriz</u> (**.'**).

Dado una matriz A,

$$A = \begin{bmatrix} 1 & 4 & 2 \\ 3 & 5 & 8 \\ 9 & 6 & 7 \end{bmatrix}$$

su transpuesta es:

$$A^{-1} = \begin{bmatrix} 1 & 3 & 9 \\ 4 & 5 & 6 \\ 2 & 8 & 7 \end{bmatrix}$$

ejemplo:

```
» A=[1 4 2;3 5 8 ;9 6 7]  ¿

A =
     1     4     2
     3     5     8
     9     6     7

» C=A.'  ¿

C =
     1     3     9
     4     5     6
     2     8     7
```

<u>La matriz transpuesta compleja conjugada</u> (’)

Una matriz transpuesta compleja conjugada es aquella que además de ser transpuesta, es decir sus filas se intercambiaron a columnas, también sus elementos complejos se transforman en los conjugados de ese elemento.

```
» A=[1-2i, 4+4i, 2-8i;3-2i, 5, 8-5i ;9+12i, 6, 7]  ¿

A =

   1.0000 - 2.0000i        4.0000 + 4.0000i        2.0000 - 8.0000i
   3.0000 - 2.0000i        5.0000                  8.0000 - 5.0000i
   9.0000 +12.0000i        6.0000                  7.0000

» C=A'  ¿  % esta es su matriz transpuesta compleja conjugada

C =

   1.0000 + 2.0000i        3.0000 + 2.0000i        9.0000 -12.0000i
   4.0000 - 4.0000i        5.0000                  6.0000
   2.0000 + 8.0000i        8.0000 + 5.0000i        7.0000

» C=A .'  ¿  % esta es su matriz transpuesta.
```

Observa que el apóstrofe lleva un punto adelante

```
C =

   1.0000 - 2.0000i        3.0000 - 2.0000i        9.0000 +12.0000i
   4.0000 + 4.0000i        5.0000                  6.0000
   2.0000 - 8.0000i        8.0000 - 5.0000i        7.0000
```

NOTA:
Obsérvese, bien, que la transpuesta es un punto y un apóstrofe (**.'**)
Y la transpuesta compleja conjugada es un simple apóstrofe (**'**)

<u>La matriz con todos los elementos 1</u> <u>(Comando **ones**).</u>

Dado una matriz identidad A, su representación sería esta:

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

» C=**ones**(3) ¿

```
C =

     1     1     1
     1     1     1
     1     1     1
```

» C=**ones**(2,4) ¿

```
C =

     1     1     1     1
     1     1     1     1
```

» C=**ones**(2,4,2) ¿

```
C(:,:,1) =

     1     1     1     1
     1     1     1     1


C(:,:,2) =

     1     1     1     1
     1     1     1     1
```

Otros Comandos (Comando **size** y **max**).

✓ El comando **MAX**.
Para vectores, MAX(X) es el elemento MAYOR en el vector X.
Para Matrices, MAX(X) es un vector fila que contiene los elementos MAYORES de cada columna de la matriz X.

```
A =

    1     4     2
    3     5     8
    9     6     7

» size(A)  ¿

ans =

    3     3
```

✓ El comando **MIN**.
Para vectores, MAX(X) es el elemento MENOR en el vector X.
Para Matrices, MAX(X) es un vector fila que contiene los elementos MENORES de cada columna de la matriz X.

✓ **SIZE** Muestra tamaño de la Matriz.
D = SIZE(X), para una matrizX de M-por-N, SIZE retornará un vector fila D = [M, N] de dos elementos el cual indica el número de filas y de columnas en la matriz.

```
» A=[1,4,2;3,5,8;9,6,7]
A =

 1     4     2
 3     5     8
 9     6     7

» size(A)  ¿

ans =

 3     3
```

✓ **SORT** Ordena en orden ascendente.
Para vectores, SORT(X) ordena los elementos de X en orden ascendente.
Para Matrices, SORT(X) ordena cada columna de X en orden
Ejemplo:
Si X = [3 7 5 0 4 2]

entonces

» **sort**(X) ¿

ans =

0 2 3 4 5 7

» A=[1,4,2;3,5,8;9,6,7] ¿

A =

1 4 2
3 5 8
9 6 7

✓ **LENGTH** Longitud de un vector.
LENGTH(X) retorna la longitud del vector X. esto es equivalente a **MAX**(**SIZE**(X)).

» H=[1,4,2,3,5,8,9,6,7] ¿

H =

1 4 2 3 5 8 9 6 7

» **length**(H) ¿

ans =

9

2.7. Generación de elementos.

Para generar elementos de un vector, se recurre a los dos puntos (**:**) proceda de la siguiente forma:
Supóngase que se quiere generar un vector de tiempo el cual se inicie (t_i) en 0 ms. A intervalos de tiempo (t_s) de 10 ms. hasta llegar a 100 ms.(t_f).

$$t = [t_i : t_s : t_f]$$

donde:

t_i =tiempo inicial.

t_s = intervalo tiempo.

t_f = tiempo final.

» t=[0:10:100] ¿

t = 0 10 20 30 40 50 60 70 80 90 100

o también

» t=(0:10:100) ¿

t = 0 10 20 30 40 50 60 70 80 90 100

otro ejemplo combinado es el siguiente:

*» t=[0:10:20**,**20:1:22]* ¿ Coma

t = 0 10 20 20 21 22

*» t=[0:10:20**;**20:1:22]* ¿ Punto y Coma

t =

0	10	20
20	21	22

2.8. TIPOS DE MATRICES PREDEFINIDAS

Existen en MATLAB varias funciones orientadas a definir con gran facilidad matrices de tipos particulares. Algunas de estas funciones son las siguientes:

eye(4)	Forma la matriz unidad de tamaño (4x4)
zeros(3,5)	Forma una matriz de *ceros* de tamaño (3x5)
zeros(4)	Ídem de tamaño (4x4)
ones(3)	Forma una matriz de *unos* de tamaño (3x3)
ones(2,4)	Idem de tamaño (2x4)
linspace(x1,x2,n)	Genera un vector con **n** valores igualmente espaciados entre **x1** y **x2**
logspace(d1,d2,n)	Genera un vector con **n** valores espaciados logarítmicamente entre 10^d1 y 10^d2. Si **d2** es **pi6**, los puntos se generan entre 10^d1 y **pi**.
rand(3)	Forma una matriz de números aleatorios entre 0 y 1, con distribución uniforme, de tamaño (3x3)
rand(2,5)	idem de tamaño (2x5)
randn(4)	Forma una matriz de números aleatorios de tamaño (4x4), con distribución normal, de valor medio 0 y varianza 1.
magic(4)	Crea una matriz (4x4) con los números 1, 2, ... 4*4, con la propiedad de que todas las filas y columnas suman lo mismo
hilb(5)	Crea una matriz de Hilbert de tamaño (5x5). La matriz de Hilbert es una matriz cuyos elementos (i,j) responden a la expresión (1/(i+j-1)). Esta es una matriz especialmente difícil de manejar por los grandes errores numéricos a los que conduce
invhilb(5)	Crea directamente la inversa de la matriz de Hilbert
kron(x,y)	Produce una matriz con todos los productos de los elementos del vector **x** por los elementos del vector **y**. Equivalente a **x**'*__y__, donde **x** e **y** son vectores fila
compan(pol)	Construye una matriz cuyo polinomio característico tiene como coeficientes los elementos del vector **pol** (ordenados de mayor grado a menor)
vander(v)	Construye la matriz de ***Vandermonde*** a partir del vector **v** (las columnas son las potencias de los elementos de dicho vector)

2.9. Algunas ecuaciones de conversión de coordenadas.

[THETA,RHO,Z] = cart2pol(X,Y,Z)

[THETA,RHO] = cart2pol(X,Y)

[THETA,PHI,R] = cart2sph(X,Y,Z)

[X,Y] = pol2cart(THETA,RHO)

[X,Y,Z] = pol2cart(THETA,RHO,Z)

[THETA,RHO,Z] = cart2pol(X,Y,Z)

[THETA,RHO] = cart2pol(X,Y)

[THETA,PHI,R] = cart2sph(X,Y,Z)

[X,Y] = pol2cart(THETA,RHO)

[X,Y,Z] = pol2cart(THETA,RHO,Z)

[X,Y,Z] = sph2cart(THETA,PHI,R)

2.10. Algunos demos de utilidad.

cztdemo	**fvtool(b,a)**	**sigdemo1**	**specgramdemo**
filtdemo	**iddemo**	**sigdemo2**	**wavedemo**
funtool	**moddemo**	**sisotool**	**windtrandemo**
fuzzy	**nnd**	**sosdemo**	**Ltiview**
disttool	**instrschool**	**instrcomm**	**instrhelp**
rltool	**fdatool**	**polytool**	**nlintool**

2.11. La Transformada Discreta de Fourier. (Discrete Fourier Transform. FFT)

La **FFT(X)** es la Transformada de Fourier Discreta (**DFT**) del vector X.

Si la longitud del vector X es potencia de dos (2) se usa un Algoritmo de base-2 de la Transformada de Fourier Rápida (Fast-Fourier Transform Algorithm) la cual es muy rápido.

Si la longitud del vector X no es potencia de dos (2), se emplea un algoritmo muy lento para funciones de base diferente de dos (2).

En el caso de Matrices, la operación **FFT** es aplicada a cada columna.

Para arreglos **N-D**, la operación **FFT** opera sobre la primera dimensión no singular.

La **FFT(X,N)** es la **FFT** para N puntos, rellenos con cero si X tiene menos puntos que N y truncado si tiene más puntos.

FFT(X,[ ],DIM) o FFT(X,N,DIM) aplica la operación **FFT** a través de la dimensión DIM.

Para la longitud **N** del vector de entrada X, la **DFT** tiene un vector X de longitud N, con elementos:

$$X(k) = \sum_{n=1}^{N} x(n) \cdot e^{(-j*2*pi*(k-1)*(n-1)/N)}, \qquad 1 <= k <= N.$$

La inversa de c**DFT** (calculada por **IFFT**) es dada por:

$$X(n) = \frac{1}{N} \sum_{k=1}^{N} x(k) \cdot e^{X(k)*\exp(\ j*2*pi*(k-1)*(n-1)/N)}, \qquad 1 <= n <= N.$$

La relación entre la **DFT** y los coeficiente de Fourier **a** y **b** en

$$X(k) = a_0 + \sum_{k=1}^{N/2} \text{sum}\ a(k)*\cos(2*pi*k*t(n)/(N*dt)) + b(k)*\sin(2*pi*k*t(n)/(N*dt))$$

son:

$$a_0 = 2*X(1)/N, \qquad a(k) = 2*\text{real}(X(k+1))/N,\ b(k) = 2*\text{imag}(X(k+1))/N$$

donde **X** es una longitud **N** de la señal discreta muestreada en un tiempo **t** con espaciamiento **dt**.

Linspace

La sintaxis para generar vectores linealmente espaciado es:

y = linspace(a,b)

y = linspace(a,b,n)

Descripción:

La función linspace genera vectores de linealmente espaciado. Es semejante al operador dos punto ":", pero ofrece el control deirecto sobre el número de puntos.

y = linspace(a,b) genera un vector fila "y" de **100** puntos linealmente espaciados entre e incluyendo **a** y **b**.

y = linspace(a,b,n) genera un vector fila "y" de **n** puntos linealmente espaciados entre e incluyendo **a** y **b**

Polyarea

La sintaxis del Area de un polígono es:

A = polyarea(X,Y)

A = polyarea(X,Y,dim)

Descripción

A = polyarea(X,Y)

Retorna el área de un polígono especificado por los vértices en los vectores X y Y.

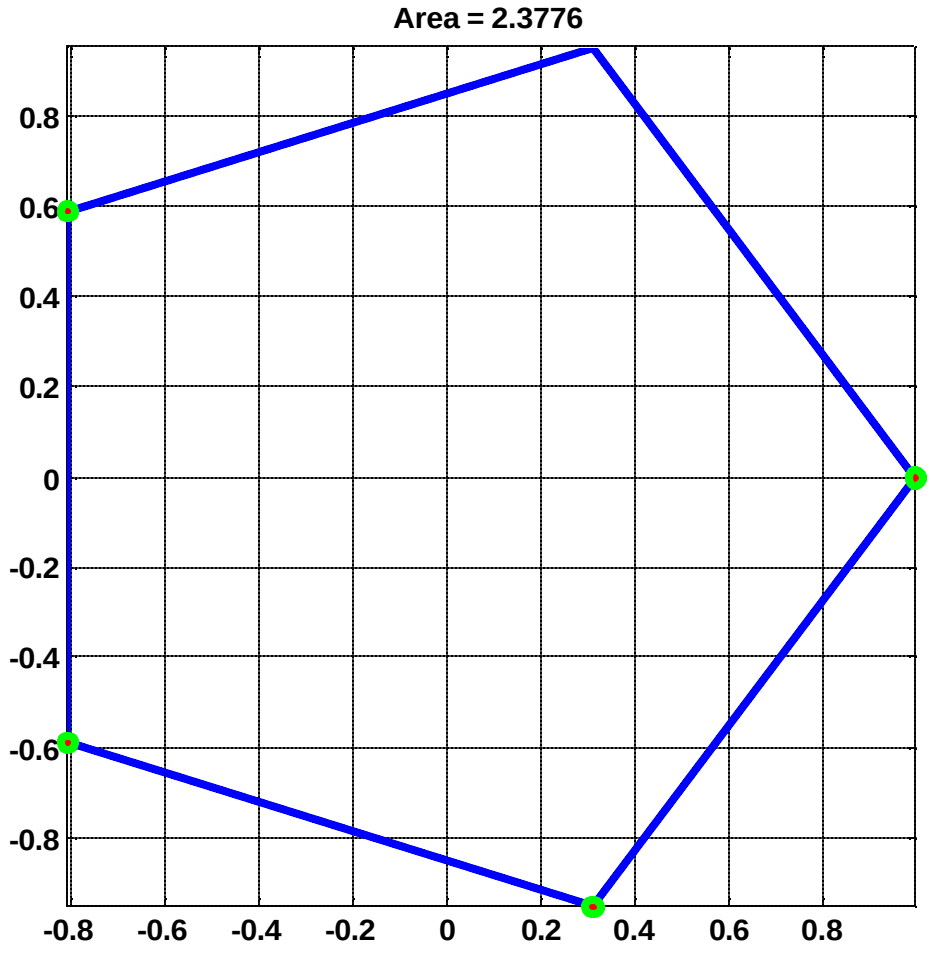
Si X y Y son matrices de la misma talla, entonces *polyarea* retornará el área del polígono definido por las columnas X y Y.

Si X y Y son arreglos multidimensionales, *polyarea* retornará el área de los polígonos en la primera dimensión de X y Y.

A = polyarea(X,Y,dim) opera en las dimensiones especificadas por el escalar dim.

Examples

```
L = linspace(0,2.*pi,6);
xv = cos(L)';
yv = sin(L)';
xv = [xv ; xv(1)];
yv = [yv ; yv(1)];
A = polyarea(xv,yv);
plot(xv,yv);
title(['Area = ' num2str(A)]); axis image
grid
```


Area = 2.3776
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8

✓ Si deseas codificar, en Matlab®, la siguiente función de transferencia,

$$\text{sys} = \frac{1{,}9s+8}{(s+1)(s+2)(s+3)}$$

puedes codificar de esta manera a través del editor "edit"

```
num = ([1.9,8])
den = poly([-1, -2, -3])
sys = tf(num,den)
```

cuya respuesta es la siguiente:

```
num =

    1.9000    8.0000


den =

     1     6    11     6


Transfer function:


      1.9 s + 8
----------------------
s^3 + 6 s^2 + 11 s + 6
```

✓ si por el contrario, tu función de transferencia es: $\text{sys} = \frac{1{,}9s+8}{s^3+6s^2+11s+6}$ deberás codificar de esta forma

```
num = ([1.9,8])
den = ([1, 6, 11, 6])
sys = tf(num,den)
```

cuya respuesta es la siguiente:

```
num =

    1.9000    8.0000


den =

     1     6    11     6

Transfer function:

       1.9 s + 8
  ----------------------
  s^3 + 6 s^2 + 11 s + 6
```

- ✓ también puedes multiplicar polinomios con la instrucción "**conv**", y una expresión en el denominador como sigue

$$\text{sys} = \frac{1{,}9s+8}{(s+1)(s+2)(s+3)}$$

puedes codificarla de la siguiente forma:

```
num = ([1.9,8])
w=[1 1]
x=[1 2]
y=[1 3]
z = conv(w,x)
den =conv(z,y)
sys = tf(num,den)
```

cuya respuesta sería la misma función de transferencia codificada anteriormente:

```
num =    1.9000    8.0000

w =     1     1

x =     1     2

y =     1     3

z =     1     3     2

den =     1     6    11     6

 Transfer function:

      1.9 s + 8
----------------------
s^3 + 6 s^2 + 11 s + 6

>>
```

✓ Otros Comandos disponibles en Matlab® son los siguientes:

```
num = ([1,8])
den = ([1, 6, 11, 6])
[z,p,k] = tf2zp(num,den)
```

este comando proporciona los ceros, los polos y la ganancia, su resultado es el siguiente:

```
num =
     1     8

den =
     1     6    11     6
z =
    -8
p =
   -3.0000
   -2.0000
   -1.0000
k =
     1
```

esto indica que existe un cero en –8 y tres polos (-3,-2,-1) con ganancia 1 (0 dB)

✓ Otro comando es el **pzmap**, ejemplo:

```
num = ([1,8])
den = ([1, 6, 11, 6])
[p,z] = pzmap(num,den)
```

cuyo resultado sería:

```
num =
     1     8

den =
     1     6    11     6

p =
   -3.0000
   -2.0000
   -1.0000

z =
    -8
```

✓ Para el calculo de fracciones parciales.

Definición:

Si no existe múltiples raíces, entonces

$$f(X) = \left(\frac{r_1}{(s - P_1)} + \frac{r_2}{(s + P_2)} + ... + \frac{r_n}{(s + P_n)} - K(X) \right)$$

El numero de polos n es n = longitud(a)-1 = longitud(r) = longitud(p)
El vector de coeficiente de términos directo estará vacío si
longitud(b) < longitud(a);
y por el contrario si la
longitud(k) = longitud(b)- longitud(a)+1

Si p(j) = ... = p(j+m-1) es un polo de multiplicidad m, entonces la expansión incluye términos de la forma:

$$f(X) = \left(\frac{r_j}{(s - P_j)} + \frac{r_{j+1}}{(s + P_j)^2} + ... + \frac{r_{j+1+m}}{(s + P_j)^m} \right)$$

Ejemplo: Sea la función f(x) dada por:

$$f(X) = \frac{5X^3 + 3X^2 - 2X + 7}{-4X^3 + 8X + 3}$$

num = [5 3 -2 7]
den = [-4 0 8 3]

y puedes calcular la expresión fracción parcial como:

[r, p, k] = **residue**(num,den)

```
r =
-1.4167
-0.6653
 1.3320

p =
 1.5737
-1.1644
-0.4093

k =
-1.2500
```

lo que sería:

$$\boxed{f(X) = \left(\frac{-1.4167}{(s-1{,}5737)} + \frac{-0{,}6653}{(s+1{,}1444)} + \frac{1{,}3320}{(s+0{,}4093)} - 1{,}2500 \right)}$$

También, puedes hacer el proceso contrario de expandir expresiones parciales a coeficientes de polinomios

[b,a] = **residue**(r,p,**k**)

```
b =
   -1.2500   -0.7500    0.5000   -1.7500

a =
    1.0000   -0.0000   -2.0000   -0.7500
```

obteniéndose:

$$\boxed{f(X) = \frac{-1{,}25X^3 - 0{,}75X^2 + 0{,}5X - 1{,}75}{X^3 - 2X - 0{,}75}}$$

El resultado puede ser expresado, como se nota, en un resultado normalizado. Dividiendo el numerador y el denominador por un mismo numero.

Algunas operaciones comunes con funciones de transferencias son las siguientes:

>> [nums,dens] = **series** (num1,den1,num2,den2)
>> [nums,dens] = **parallel** (num1,den1,num2,den2)
>> [nums,dens] = **feedback** (num1,den1,num2,den2,sign)
>> [nums,dens] = **cloop** (num,den, sign) % para sistemas con realimentados unitarios

donde el "**sign**" (opcional) es –1 para realimentación negativa y +1 para realimentación positiva.

el resultado de la función de transferencia en cada caso, esta dado por:

Serie(Cascada): $G_s(s) = G_1(s)G_2(s)$

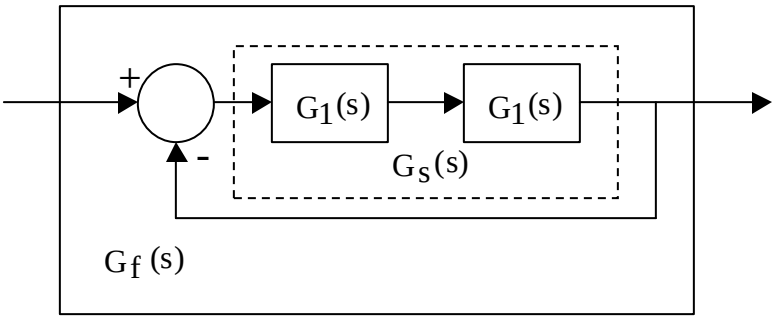
Paralelo: $G_p(s) = G_1(s) + G_2(s)$

Realimentado: $G_f(s) = \dfrac{G_1(s)}{1 + G_1(s)G_2(s)}$

Realimentación unitaria: $G_c(s) = \dfrac{G(s)}{1 + G(s)}$

Ejemplo: supóngase dos funciones de transferencia y

$G_1(s) = 1/(s+1)$ $\qquad$ $G_2(s) = 1/(s+2)$



```
[nums,dens] = series (1,[1 1],1,[1 2])
[numf,denf] = cloop (nums,dens, -1)
sys = tf(numf,denf)
```

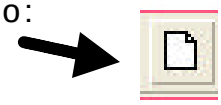
cuyo resultado es:

```
nums =     0     0     1
dens =     1     3     2
numf =     0     0     1
denf =     1     3     3
 Transfer function:
      1
--------------
s^2 + 3 s + 3

>>
```

FUNCIONES EN FICHEROS-M

Para crear un fichero-M escoger New del menú File y seleccionar M-file.
o en su defecto el icono:



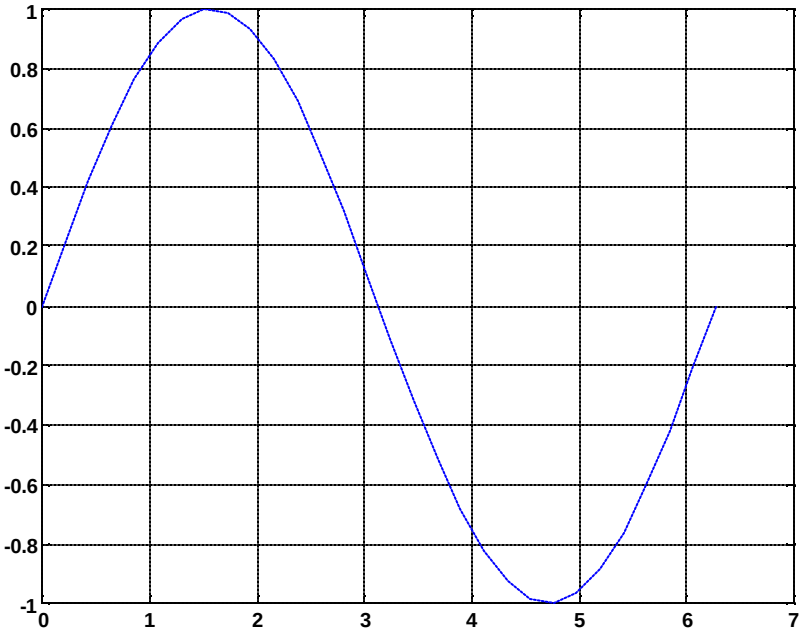
Este procedimiento devuelve una ventana para la edición de texto donde se introducen las órdenes de Matlab.

Ejemplo:

```
function  res = grafica(x)
%  representamos la función seno.
res = plot (x,sin(x))
grid
```

Se guarda el fichero como grafica .m
Desde la línea de comandos:
Matlab ejecutará las órdenes de grafica.m escribiendo grafica(x) en la línea de comandos:

```
>> x=linspace(0,2*pi,30);
>> grafica(x)
>> help grafica
     representamos la función seno.
```



- Son ficheros creados con un editor de texto.
- Al solicitar help grafica aparecen las líneas de comentario.
- Tengo que trabajar en el directorio donde guarde el fichero.

5 Gráficos en Matlab

MATLAB tiene una alta capacidad gráfica en aplicaciones técnicas y científicas.

Los comandos básicos son: **plot**, **loglog**, **semilogx**, **semilogy** y **polar**.

Daremos varios ejemplos el cual usted discutirá con su profesor en horas de seccion de Matlab.

Gráfico 1

```
» x=(0:0.05:10);
» y=sin(x).*exp(-0.4*x);
» plot(x,y)
» xlabel('x');
» ylabel('y');
```

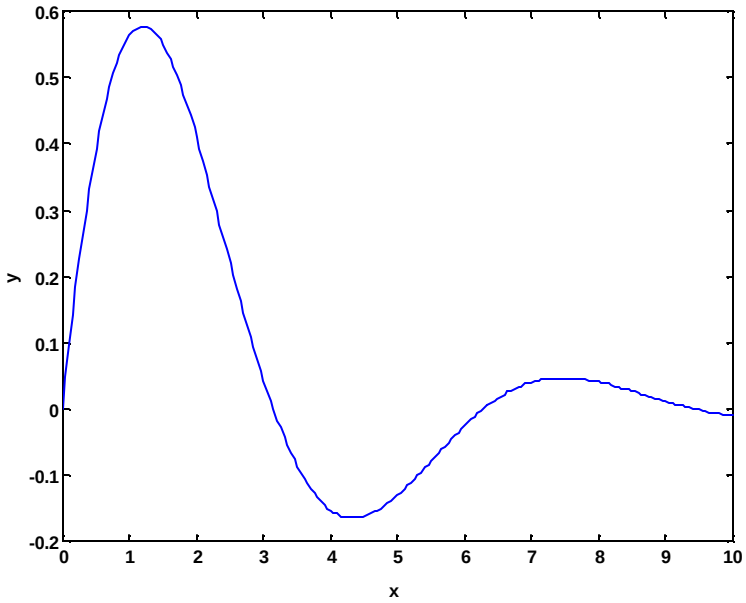


Gráfico 2

```
» x=(0:0.05:10);
» y=sin(x).*exp(-0.4*x);
» plot(x,y)
» xlabel('x');
» ylabel('y');
» title('G R A F I C O    2')
» grid
»
```

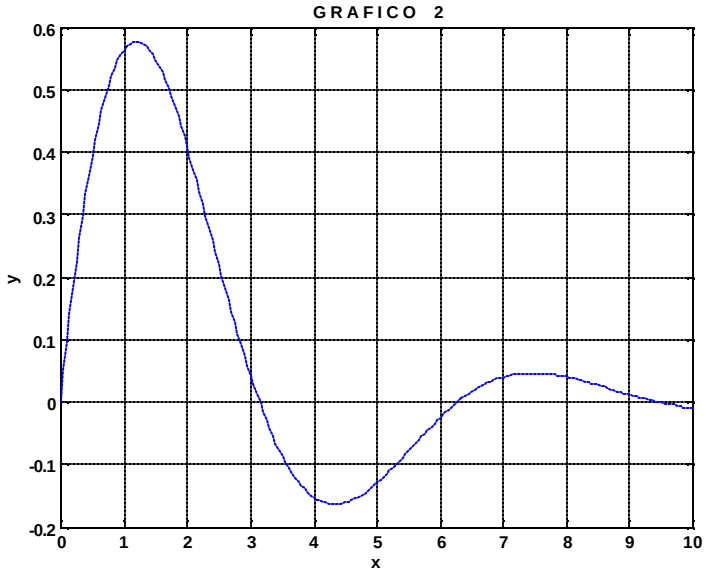


Gráfico 3

```
» t=0:0.05:pi+.01;
» y=sin(3*t).*exp(-0.4*t);
» polar(t,y)
» title('GRÁFICA POLAR')
» grid
»
```

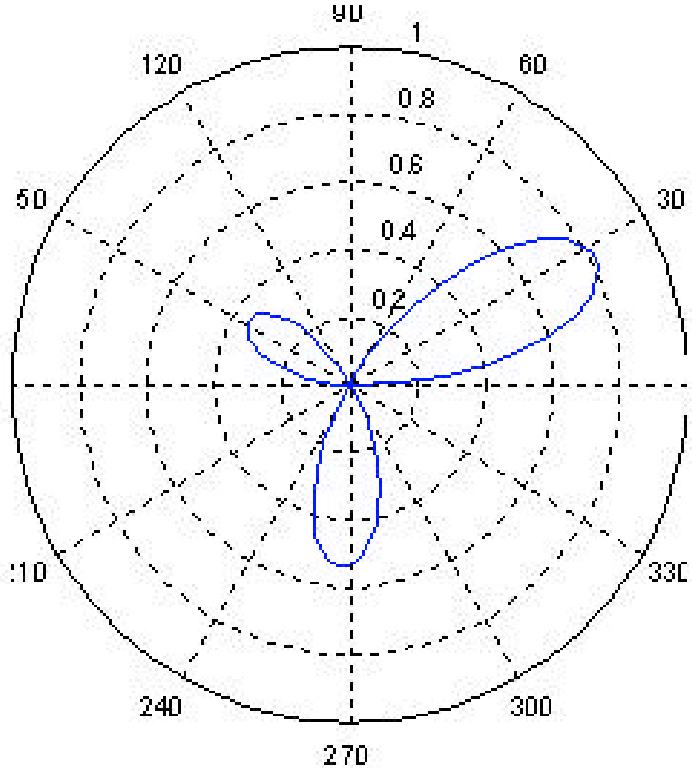


Gráfico 4

```
» t=0.1:0.1:3;
» x=exp(t);
» y=exp(t.*sinh(t));
» loglog(x,y);
» grid
» xlabel('x');ylabel('y')
» title('Gráfica Logarítmica')
```

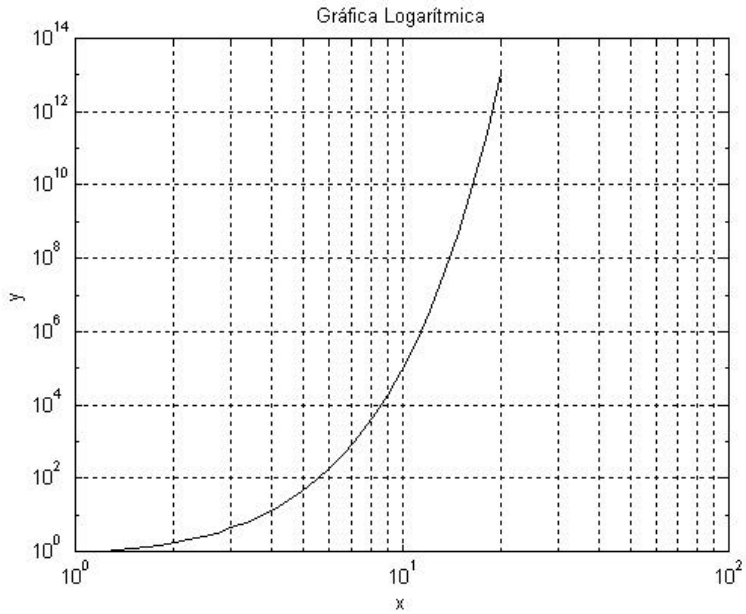


Gráfico 5

```
t=0.1:0.1:3;
» x=exp(t);
» y=exp(t.*sinh(t));
» semilogy(x,y);
» grid
» xlabel('x');ylabel('y')
» title('Gráfica Semilogarítmica')
```

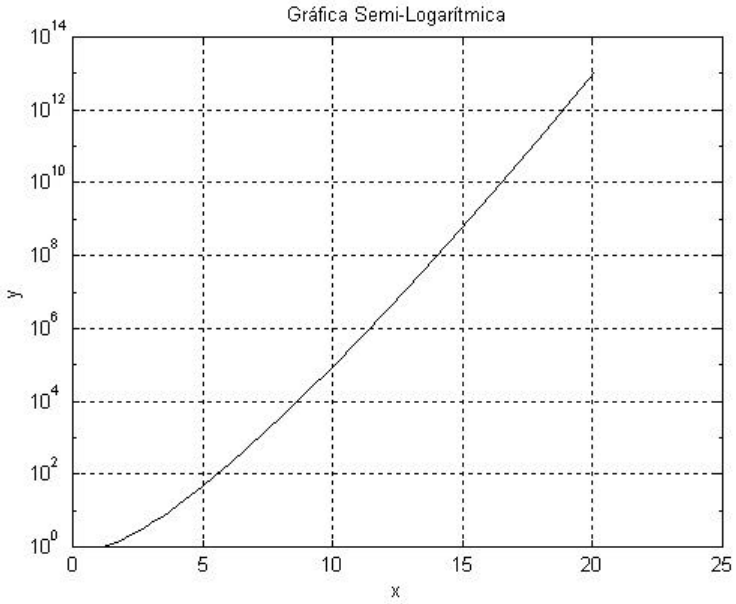


Gráfico 6

```
t=0.1:0.1:3;
» x=t;
» y=exp(t.*t);
» semilogy(x,y);
» grid
» xlabel('x');ylabel('y')
» title('Gráfica Semilogarítmica')
```

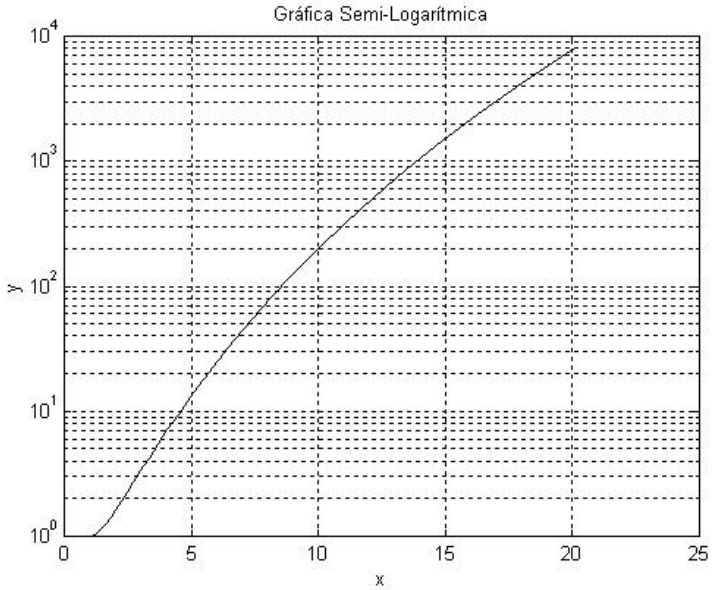


Gráfico 7

```
» x=0:0.05:5;
» y=sin(x);
» z=cos(x);
» plot(x,y,x,z)
» grid
```

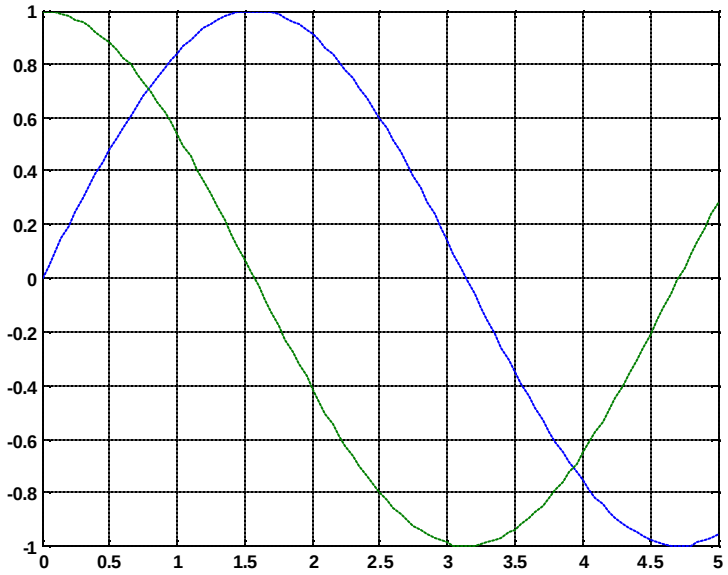
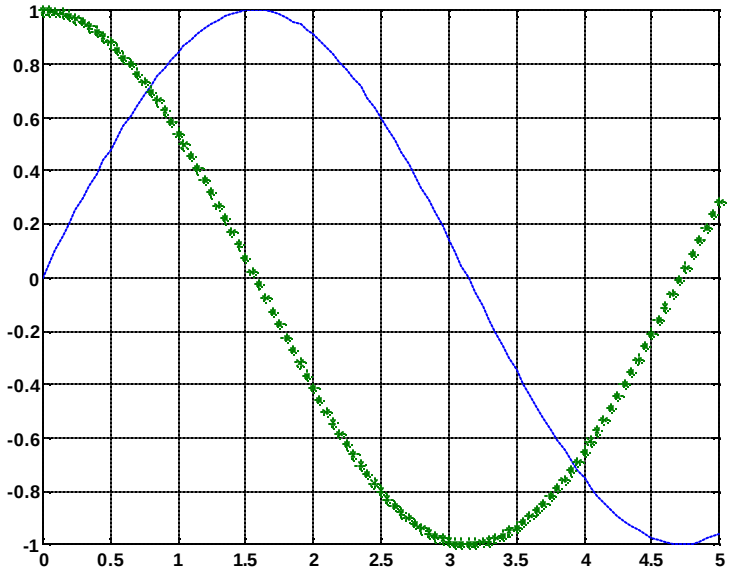


Gráfico 8

```
» x=0:0.05:5;
» y=sin(x);
» z=cos(x);
» plot(x,y,'--',x,z,'*')
» grid
```



Color de Línea	Símbolo
Rojo	r
Amarillo	y
Magenta	m
Turquesa	c
Verde	g
Azul	b
Blanco	w
Negro	k

Plot(x,y,'g')

Tipo de Línea	Símbolo
Continuas	-
Guiones	--
Punteadas	:
Guiones y puntos	-.

Plot(x,y,'**:**')

Tipo de Marca	Símbolo
Punto	.
Más	+
Estrella	*
Círculo	o
Marca X	x

Plot(x,y,'+')

clf = borra todo lo que haya en la ventana de gráficos.
cla = borra la curva graficada y redibuja los ejes.

Gráfico 9

```
» xa=-2:.2:2;
» ya=-2:.2:2;
» [x,y]=meshgrid(xa,ya);
» z=x.*exp(-x.^2-y.^2);
» surf(x,y,z)
» title('Esta  es una Gráfica en 3-D')
» xlabel('x');ylabel('y');zlabel('z');
»
```

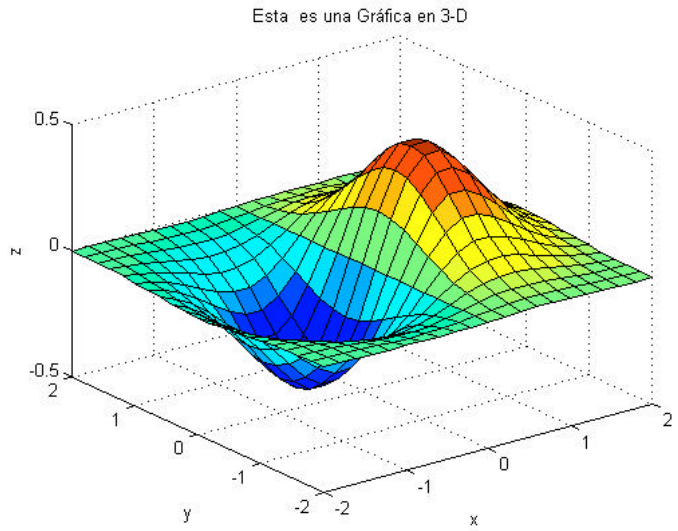
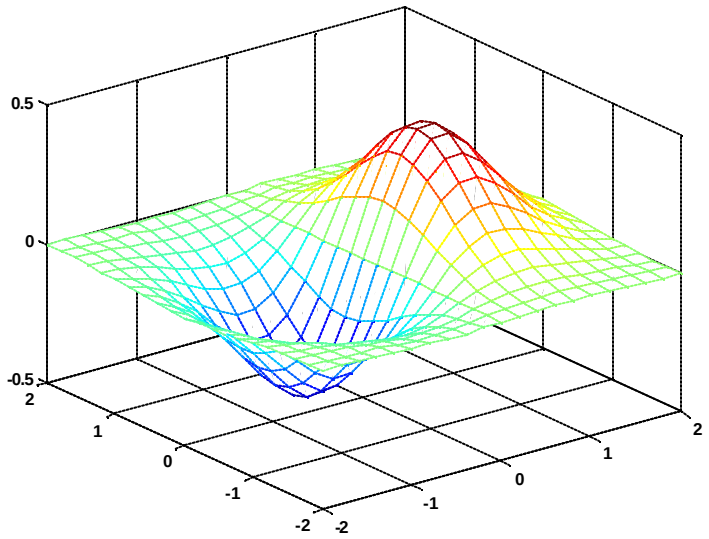


Gráfico 9

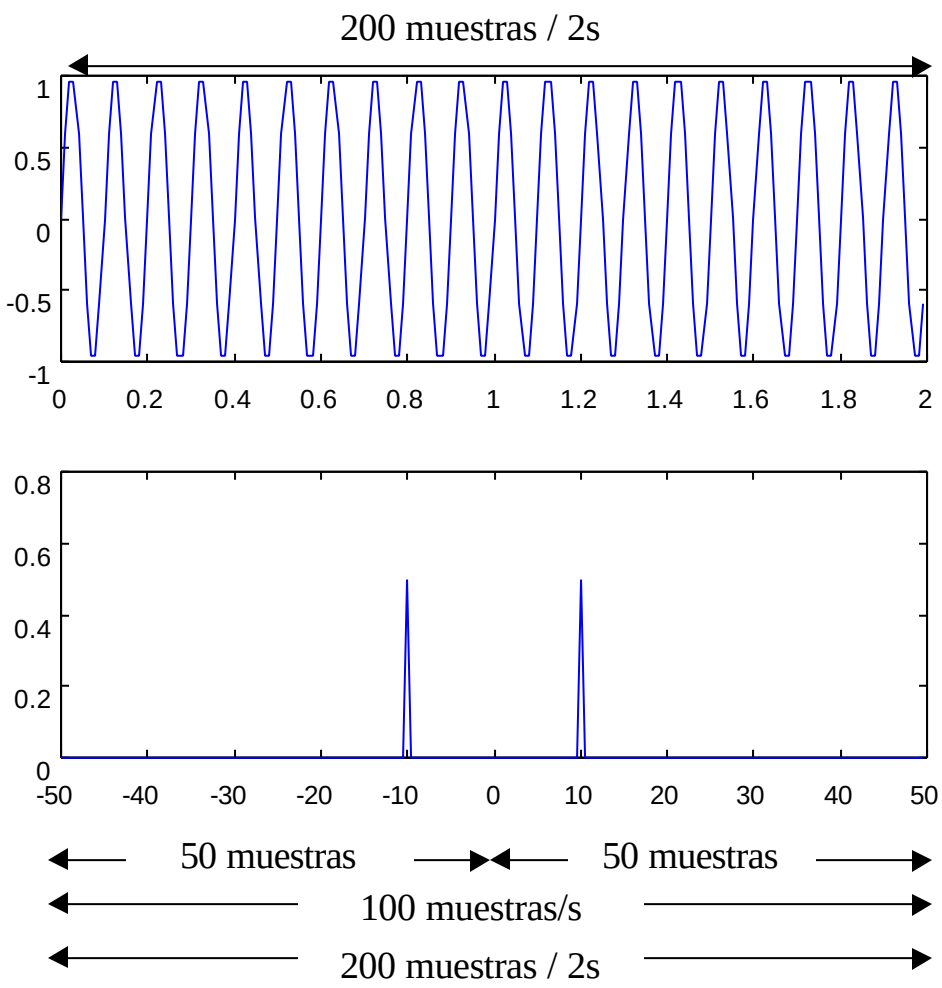
```
» xa=-2:.2:2;
» ya=-2:.2:2;
» [x,y]=meshgrid(xa,ya);
» z=x.*exp(-x.^2-y.^2);
» mesh(x,y,z)
»
```



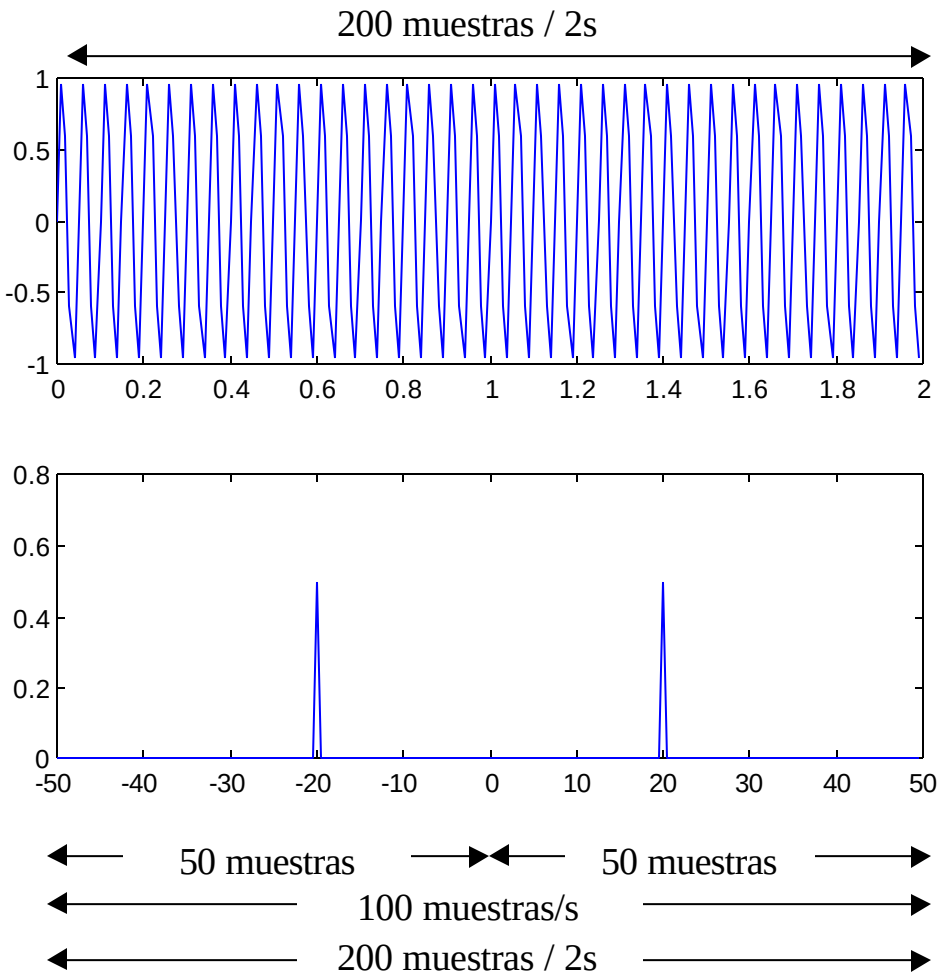
Para una frecuencia de 10 Hz (0,1 s es el periodo de la onda) se desea un muestreo cada 0,01 s. además de ver en pantalla 20 ciclos, quiere decir que 20 x 0,1s = 2s es el máximo de tiempo que necesitamos

```
f=10;
t=0:0.01:1.99;
s=sin(2*pi*f*t);
subplot(2,1,1);
plot(t,s)
Smag=abs(fftshift(fft(s,200)))/200;
f=[-100:99.9]/2;
subplot(2,1,2);
plot(f,Smag)
```

Muestreo=1,99/0,01+1=200



```
f=20;
t=0:0.01:1.99;
s=sin(2*pi*f*t);
subplot(2,1,1);
plot(t,s)
Smag=abs(fftshift(fft(s,200)))/200;
f=[-100:99.9]/2;
subplot(2,1,2);
plot(f,Smag)
```



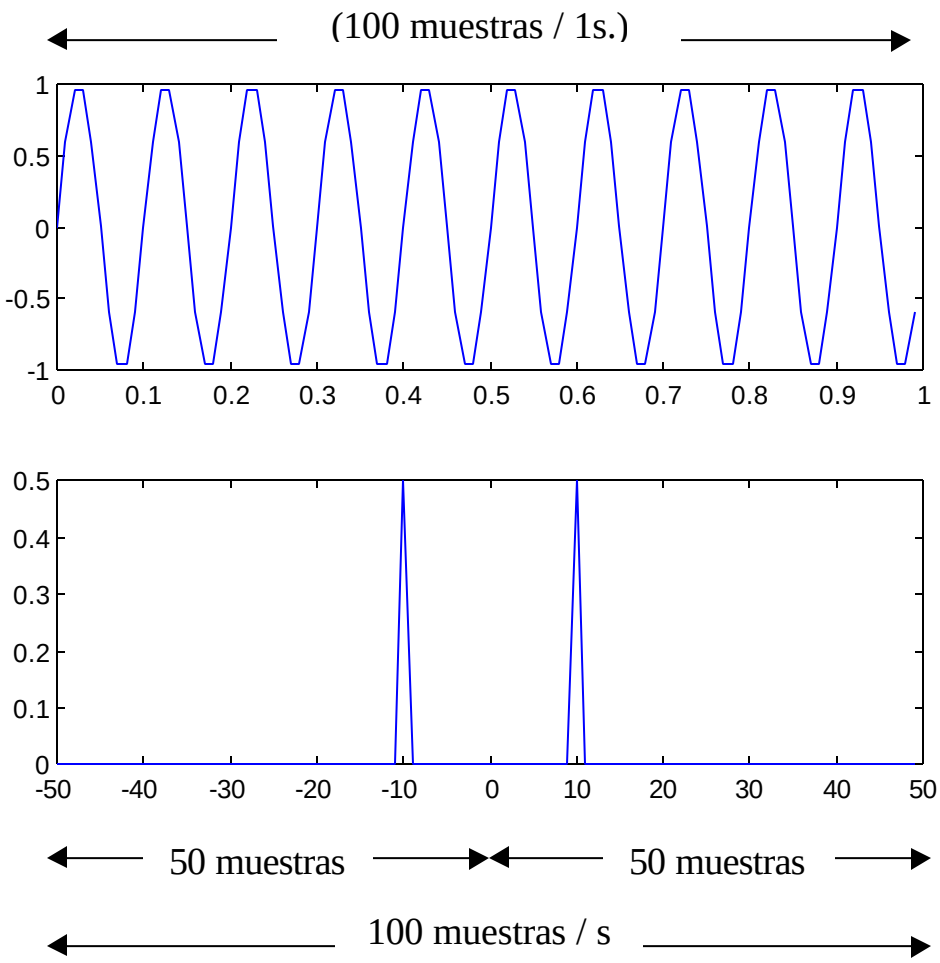
Supóngase una Onda senoidal de frecuencia 10 Hz el cual se desea analizar únicamente 10 ciclos completos de la misma a una tasa de muestreo de

$1/\tau$ =100 Hz.

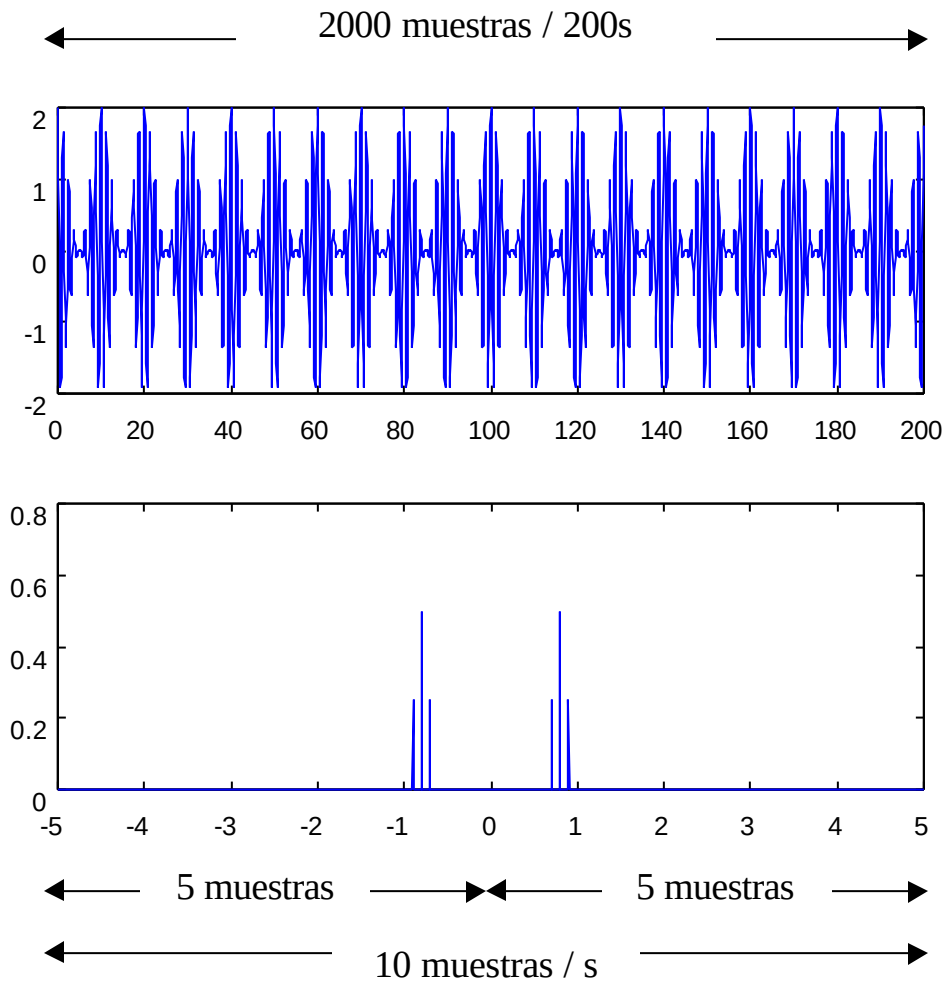
Esto implica que τ = 0,01 s (100 muestras en total de los 10 ciclos, es decir 10 muestras por ciclo)

τ

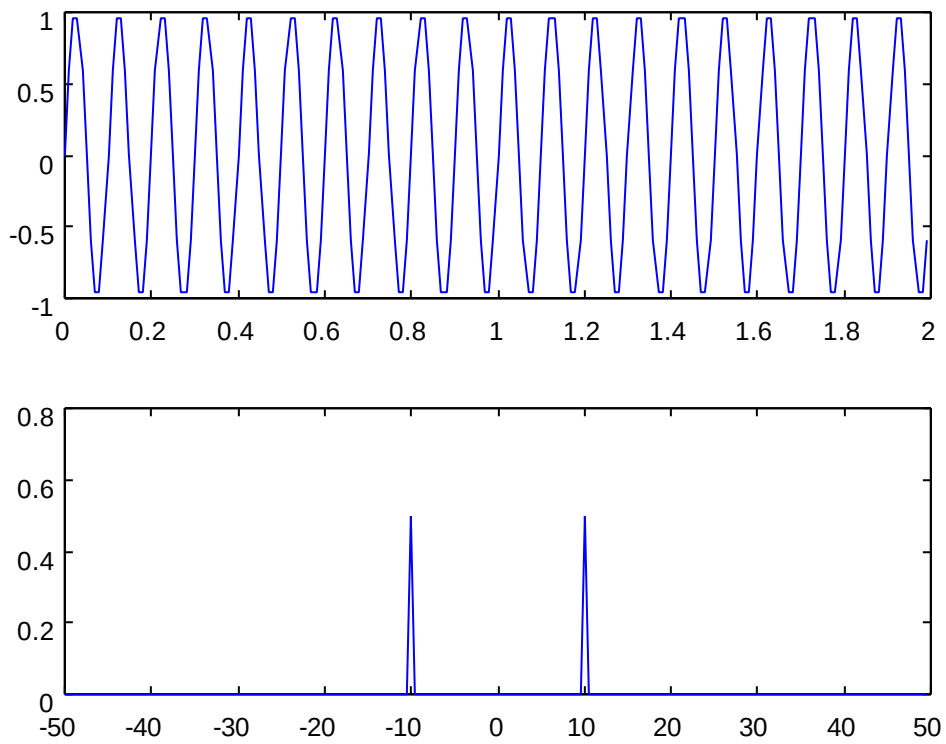
```
f=10;
t=0:0.01:0.99;
s=sin(2*pi*f*t);
subplot(2,1,1);
plot(t,s)
Smag=abs(fftshift(fft(s,100)))/100;
f=[-50:49.9]/1;
subplot(2,1,2);
plot(f,Smag)
```



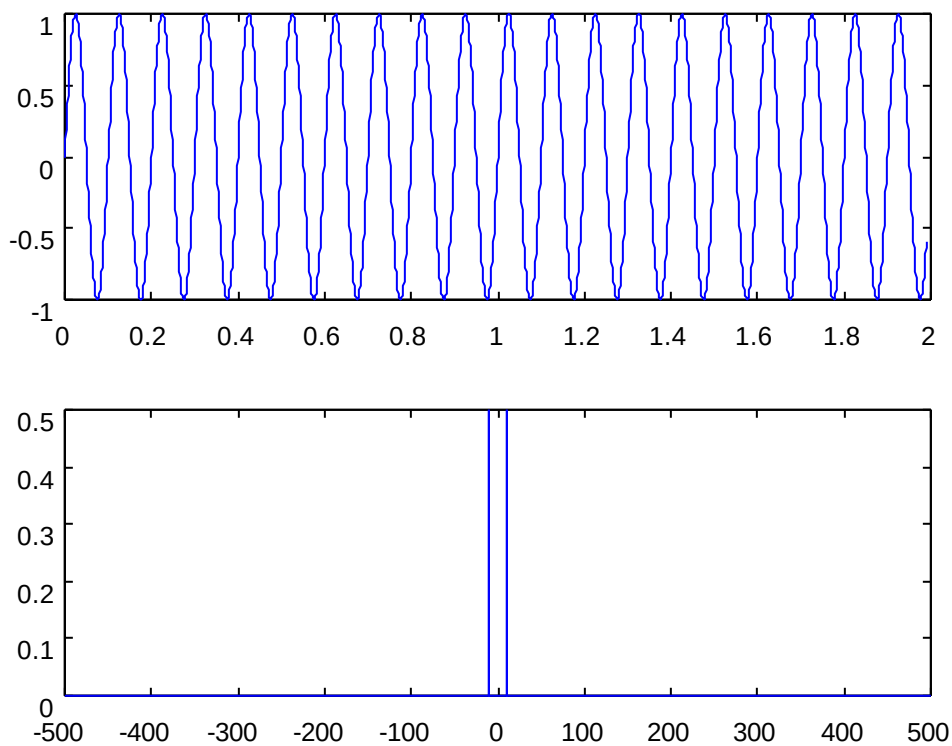
```
Ac=1;
A0=1;
fc=0.8;
f0=0.1;
t=0:0.1:199.9;
s=Ac*[1+cos(2*pi*f0*t)].*cos(2*pi*fc*t);
Smag=abs(fftshift(fft(s,2000)))/2000;
w=10*[-1000:999]/2000;
subplot(2,1,1);
plot (t,s)
subplot(2,1,2);
plot (w,Smag)
```



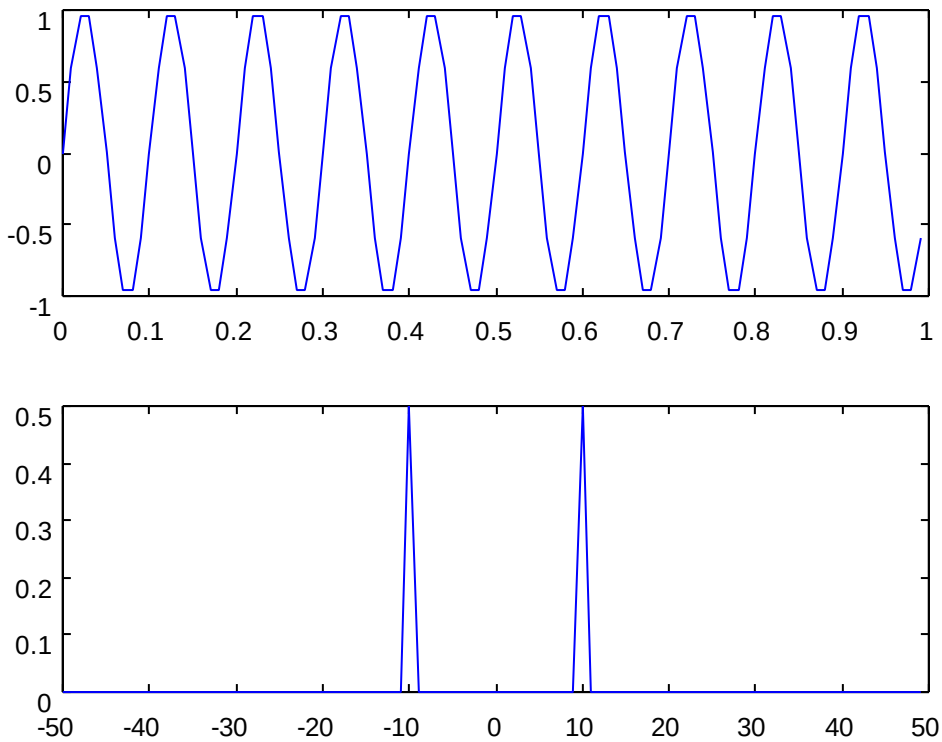
```
f=10;
t=0:0.01:1.99;
s=sin(2*pi*f*t);
subplot(2,1,1);
plot(t,s)
Smag=abs(fftshift(fft(s,200)))/200;
f=[-100:1:99.9]/2;
subplot(2,1,2);
plot(f,Smag)
```



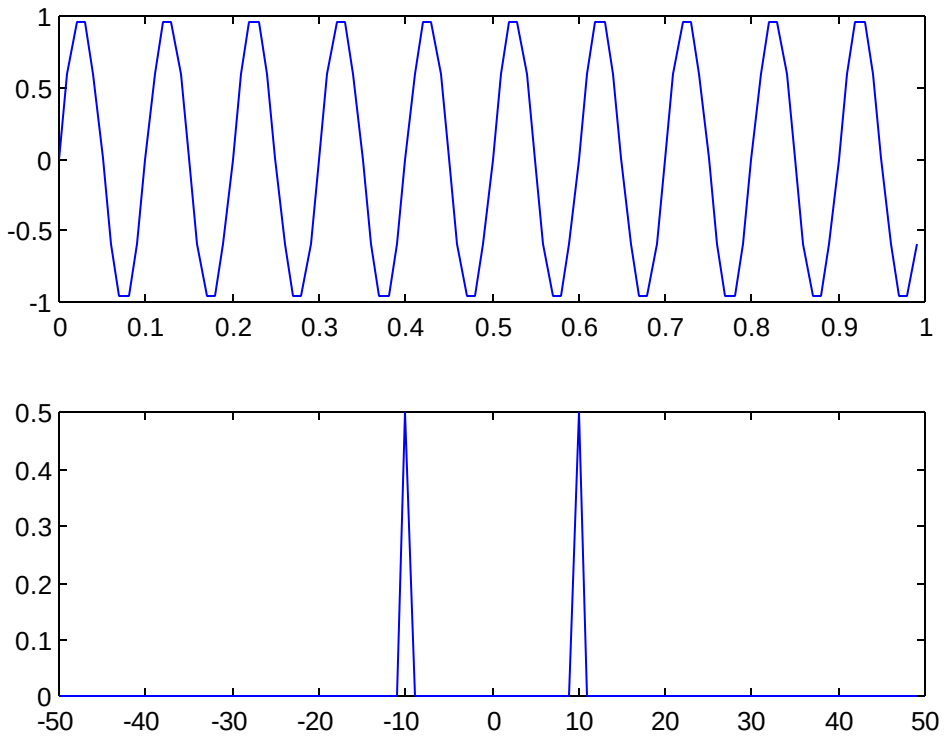
```
f=10;
t=0:0.001:1.99;
s=sin(2*pi*f*t);
subplot(2,1,1);
plot(t,s)
Smag=abs(fftshift(fft(s,2000)))/2000;
f=[-1000:1:999]/2;
subplot(2,1,2);
plot(f,Smag)
```



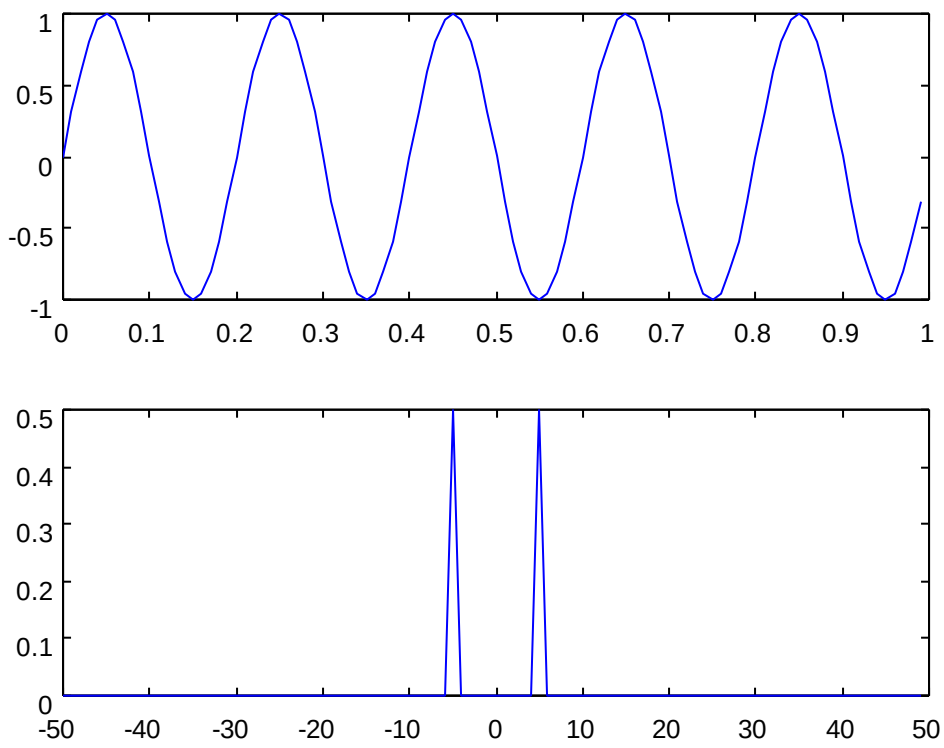
```
f=10;
t=0:0.01:0.99;   % tiempo de muestreo = 1 s.
s=sin(2*pi*f*t);
subplot(2,1,1);
plot(t,s)
Smag=abs(fftshift(fft(s,100)))/100;   % número de muestras =
100
% frecuencia de muestreo = 100 muestras/1s = 100 Hz
% frecuencia máxima de plot = 100 Hz/2 = 50 Hz.
f=[-50:49];
subplot(2,1,2);
plot(f,Smag)
```



```
f=10;
t=0:0.01:0.99;% tiempo de muestreo = 1 s.
s=sin(2*pi*f*t);
subplot(2,1,1);
plot(t,s)
Smag=abs(fftshift(fft(s,100)))/100; % número de muestras = 100
% frecuencia de muestreo = 100/1s = 100 Hz
% frecuencia máxima de plot = 100 Hz/2 = 50 Hz.
f=[-100:2:98]/2;
subplot(2,1,2);
plot(f,Smag)
```



```
f=5;
t=0:0.01:0.99;
s=sin(2*pi*f*t);
subplot(2,1,1);
plot(t,s)
Smag=abs(fftshift(fft(s,100)))/100;
f=[-100:2:99.9]/2;
subplot(2,1,2);
plot(f,Smag)
```



$$frecuencia\ \ \text{máx}\,ima\ \ del\ \ plot\ \ = \left(\frac{número\ \ de\ muestras}{tiempo\ de\ muestreo}\right)/2$$

$$frecuencia\ \ \text{máx}\,ima\ \ del\ \ plot\ \ = frecuencia\ \ de\ muestreo/2$$

4.1.- PRÁCTICA No. 01 AM COMPLETA

En la descripción en el dominio del tiempo la modulación en amplitud, la onda modulada está compuesta por la portadora más un producto de la señal del mensaje (es decir, la onda moduladora) y la portadora. De tal manera, en el caso de la modulación senoidal considerada, se tiene:

$$s(t) = A_c\,[1+\mu\cos(\omega_0 t)]\,\cos(\omega_c t)$$

donde μ es el factor de modulación.

Amplitud de la portadora	$Ac=1$
Frecuencia de la portadora	$\omega_c=0.8\pi$ rad/s
Frecuencia de la modulación	$\omega_0=0.1\pi$ rad/s

Se desea analizar 10 ciclos completos de la Onda AM.
A una tasa de muestreo de $1/\tau$ =10Hz. (2000 muestras)
La Banda de frecuencia es de $-10\pi \le \omega \le 10\pi$
Se desean analizar tres situaciones diferentes:

a. μ=0.5 correspondiente a una submodulación.
b. μ=1.0 correspondiente a una modulación 100%.
c. μ=2.0 correspondiente a una sobremodulación.

SOLUCION EN MATLAB

```
» Ac=1;
» wc=0.8*pi;
» w0=1.0*pi;
» mu=0.5;
» t=0:0.1:199.9;
» s=Ac*(1+mu*cos(w0*t)).*cos(wc*t);
» plot(t,s)
» Smag=abs(fftshift(fft(s,2000)))/2000;
» w=10*[-1000:999]*2*pi/2000;
» plot (w,Smag)
```

μ debe variar para cada caso

4.2.- PRÁCTICA No. 02 MODULACIÓN DSB-SC

En esta modulación se suprime la portadora y ambas bandas laterales se transmiten completas. Se produce multiplicando la onda moduladora por la onda portadora. De modo que en este caso de modulación senoidal, tenemos:

$$s(t) = A_c A_o \cos(\omega_c t)\cos(\omega_0 t)$$

Amplitud de la portadora	Ac=1
Amplitud de la portadora	Ao=1
Frecuencia de la portadora	ω_c=0.8π rad/s
Frecuencia de la modulación	ω_0=0.1π rad/s

Se desea analizar 10 ciclos completos de la Onda AM.
A una tasa de muestreo de $1/\tau$ =10Hz. (2000 muestras)
La Banda de frecuencia es de $-10\pi \leq \omega \leq 10\pi$

SOLUCION EN MATLAB

```
» Ac=1;
» A0=1;
» wc=0.8*pi;
» w0=0.1*pi;
» t=0:0.1:199.9;
» s=Ac* A0 *cos(w0*t)).*cos(wc*t);
» plot(t,s)
» Smag=abs(fftshift(fft(s,2000)))/2000;
» w=10*[-1000:999]*2*pi/2000;
» plot (w,Smag)
```

4.3.- PRÁCTICA No. 03
Detección Coherente de DSB-SC

Para la detección coherente de la modulación **DSB-SC** multiplicamos la señal s(t) por una replica de la onda portadora, y luego pasamos el resultado por un filtro pasabajo.

(se supone sincronismo perfecto entre el transmisor y el receptor).

$$s(t) = A_c A_o \cos(\omega_c t)\cos(\omega_0 t)$$

$$v(t) = s(t)\cos(\omega_c t)$$

Amplitud de la portadora $Ac = 1$
Amplitud de la portadora $Ao = 1$
Frecuencia de la portadora $\omega_c = 0.8\pi$ rad/s
Frecuencia de la modulación $\omega_0 = 0.1\pi$ rad/s
Se desea analizar 10 ciclos completos de la Onda AM.
A una tasa de muestreo de $1/\tau = 10$Hz. (2000 muestras)
La Banda de frecuencia es de $-10\pi \leq \omega \leq 10\pi$

En el experimento que estamos considerando, el orden del filtro es 3 y su frecuencia de corte normalizado es de 0.025 que se calcula así:

$$\frac{\text{Frecuencia real del corte del filtro}}{\text{Mitad de la velocidad de muestreo}} = \frac{0.25\,\Pi \text{ rad/s}}{10\,\Pi \text{ rad/s}} = 0.025$$

SOLUCION EN MATLAB

```
» AC=1;
» A0=1;
» wc=0.8*pi;
» w0=0.1*pi;
» t=0:0.1:199.9;
» s=AC*A0*cos(w0*t).*cos(wc*t);
» v=s.*cos(wc*t);
» plot(t,v)
» Smag=abs(fftshift(fft(v,2000)))/2000;
» w=10*[-1000:999]*2*pi/2000;
» plot(w,Smag)
» [b,a]=butter(3,0.025);
» output=filter(b,a,v);
» plot(t,output)
» Smag=abs(fftshift(fft(output,2000)))/2000;
» w=10*[-1000:999]*2*pi/2000;
» plot(w,Smag)
```

4.4.- PRÁCTICA No. 04 Coordenadas polares

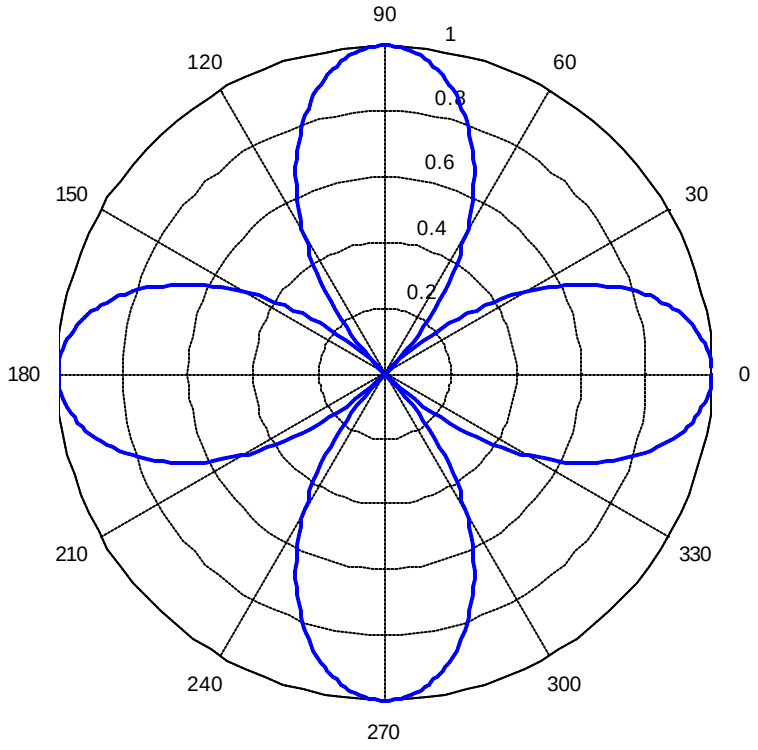
La ecuación de cuatro pétalos en forma polar es:

$$r=\cos(2\theta)$$

Gráficar en Matlab

SOLUCION EN MATLAB

```
» t= [pi/200:pi/200:2*pi]
» r= cos(2*t)
» polar(t,r)
» grid;
```



Pruebe Ud. mismo la figura que presenta la ecuación:

$$r=\cos(\cos(2\theta))$$

4.5.- PRÁCTICA No. 05 Patrones de Radiación

```
% Este programa Calcula y Gráfica los patrones de los campos uniformes de
% arreglos de fuentes.
% Del Libro Electromagnetismo del Profesor John Kraus 4ta. edición.
clear;
timesrun=0;
while timesrun <1000,
if timesrun==0,
     SP=0.5;
     PH=0;
     N=5;
     MF=1;
  else,
     SP=input('Espaciamiento de los elementos en Longitud de Onda: ');
     PH=input('Diferencia de Fase entre los elementos en grados: ');
     N=input('Numero de elementos: ');
     MF=input('Factor de multiplicacion del patron de radiacion: ')';
  end;

A=0.01:0.01:6.27;
U=(2*pi*SP*cos(A)+(pi*PH/180))/2;
FP=(sin(N*U)./sin(U));
FPD=(sin(N*U)./sin(U)).*cos(A);
R=MF.*abs(FP);
pause(0.9);
B=0.01:0.01:3.14;
  W=(2*pi*SP*cos(B)+(pi*PH/180))/2;
  PP=(sin(N*W)./sin(W)).^2;
  Z=0.01*sin(B).*PP;
  SUM=sum(Z);
DR=(2*(N^2))/SUM;
DBI=10*log10(DR);

theta=(0:2*pi/626:2*pi);
polar(theta,(round(R*100))/100);
xlabel(['Diagrama de Campo= ',num2str(N),',  Espaciado de Fuentes= ',num2str(SP),'
Lamda,  ',' Delta fase= ',num2str(PH),' Grados']);
text(max(R),max(R),['D = ',num2str((round(DBI*100))/100),' dBi']);

sep=max(R)/20;
xcent=sep*((-(N-1)/2)-1);
for elcount=1:N
  xcent=xcent+sep;
  ycent=0;
  radius=sep/5;
  xp=xcent+[-radius:radius/10:radius];
  yp=ycent+real(sqrt((radius^2)-(xp-xcent).^2));
  patch(xp,yp,'k');
  patch(xp,-yp,'k');
end;
```

```
rp=input('Escriba 1 para la Grafica rectangular: ');
if rp==1
   figure(2);
   plot(theta*(180/pi),R);
   grid on;
   zoom on;
   title(['Diagrama de Campo = ',num2str(N),',  Espaciado de Fuentes=',num2str(SP),'
Lamda,  ',' Delta fase= ',num2str(PH),' Grados']);
   xlabel('Angulo (Grados)');
   ylabel('Amplitud (Lineal)');
end;
logp=input('Escriba 1 para la Grafica en dB: ');
if logp==1
   figure(3);
   plot(theta*(180/pi),10*log10(abs(R)));
   grid on;
   title(['Diagrama del Campo = ',num2str(N),',  Espaciado de Fuentes= ',num2str(SP),'
Lamda,  ',' Delta fase= ',num2str(PH),' Grados']);
   xlabel('Angulo (grados)');
   ylabel('Amplitud (dB)');
end;

powerp=input('Escriba 1 para el patron de potencia: ');
if powerp==1
   figure(4);
   plot(theta*(180/pi),20*log10(abs(R)));
   grid on;
   title(['Diagrama de Potencia = ',num2str(N),',  Espaciado de Fuentes= ',num2str(SP),'
Lamda,  ',' Delta fase= ',num2str(PH),' Grados']);
   xlabel('Angle (deg)');
   ylabel('Amplitude (dB)');
end;

timesrun=timesrun+1;
done=input('Escriba 1 para modificar parametros: ');
if done~=1,
   timesrun=1000;
end;
end;
```

HELP topics:

matlab\general	-	General purpose commands.
matlab\ops	-	Operators and special characters.
matlab\lang	-	Programming language constructs.
matlab\elmat	-	Elementary matrices and matrix manipulation.
matlab\elfun	-	Elementary math functions.
matlab\specfun	-	Specialized math functions.
matlab\matfun	-	Matrix functions - numerical linear algebra.
matlab\datafun	-	Data analysis and Fourier transforms.
matlab\polyfun	-	Interpolation and polynomials.
matlab\funfun	-	Function functions and ODE solvers.
matlab\sparfun	-	Sparse matrices.
matlab\graph2d	-	Two dimensional graphs.
matlab\graph3d	-	Three dimensional graphs.
matlab\specgraph	-	Specialized graphs.
matlab\graphics	-	Handle Graphics.
matlab\uitools	-	Graphical user interface tools.
matlab\strfun	-	Character strings.
matlab\iofun	-	File input/output.
matlab\timefun	-	Time and dates.
matlab\datatypes	-	Data types and structures.
matlab\dde	-	Dynamic data exchange (DDE).
matlab\demos	-	Examples and demonstrations.
simulink\simulink	-	SIMULINK
simulink\blocks	-	SIMULINK block library.
simulink\simdemos	-	SIMULINK demonstrations and samples.
simulink\dee	-	Differential Equation Editor
toolbox\symbolic	-	Symbolic Math Toolbox.
toolbox\signal	-	Signal Processing Toolbox.
toolbox\control	-	Control System Toolbox.
control\obsolete	-	(No table of contents file)
toolbox\local	-	Preferences.

General purpose commands.
MATLAB Toolbox Version 5.0 Student Edition 31-Dec-1996

General information
help - On-line help, display text at command line.
helpwin - On-line help, separate window for navigation.
helpdesk - Comprehensive hypertext documentation and troubleshooting.
demo - Run demonstrations.
ver - MATLAB, SIMULINK, and toolbox version information.
whatsnew - Display Readme files.
Readme - What's new in MATLAB 5.

Managing the workspace.
who - List current variables.
whos - List current variables, long form.
clear - Clear variables and functions from memory.
pack - Consolidate workspace memory.
load - Load workspace variables from disk.
save - Save workspace variables to disk.
quit - Quit MATLAB session.

Managing commands and functions.
what - List MATLAB-specific files in directory.
type - List M-file.
edit - Edit M-file.
lookfor - Search all M-files for keyword.
which - Locate functions and files.
pcode - Create pre-parsed pseudo-code file (P-file).
inmem - List functions in memory.
mex - Compile MEX-function.

Managing the search path
path - Get/set search path.

```
   addpath     - Add directory to search path.
   rmpath      - Remove directory from search path.
   editpath    - Modify search path.

Controlling the command window.
   echo        - Echo commands in M-files.
   more        - Control paged output in command window.
   diary       - Save text of MATLAB session.
   format      - Set output format.

Operating system commands
   cd          - Change current working directory.
   pwd         - Show (print) current working directory.
   dir         - List directory.
   delete      - Delete file.
   getenv      - Get environment variable.
   !           - Execute operating system command.
   dos         - Execute DOS command and return result.
   unix        - Execute UNIX command and return result.
   vms         - Execute VMS DCL command and return result.
   web         - Open Web browser on site or files.
   computer    - Computer type.

Debugging M-files.
   debug       - List debugging commands.
   dbstop      - Set breakpoint.
   dbclear     - Remove breakpoint.
   dbcont      - Continue execution.
   dbdown      - Change local workspace context.
   dbstack     - Display function call stack.
   dbstatus    - List all breakpoints.
   dbstep      - Execute one or more lines.
   dbtype      - List M-file with line numbers.
   dbup        - Change local workspace context.
   dbquit      - Quit debug mode.
   dbmex       - Debug MEX-files (UNIX only).
```

Operators and special characters.

Arithmetic operators.

plus	- Plus	+
uplus	- Unary plus	
minus	- Minus	-
uminus	- Unary minus	-
mtimes	- Matrix multiply	*
times	- Array multiply	.*
mpower	- Matrix power	^
power	- Array power	.^
mldivide	- Backslash or left matrix divide	\
mrdivide	- Slash or right matrix divide	/
ldivide	- Left array divide	.\
rdivide	- Right array divide	./
kron	- Kronecker tensor product	kron

Relational operators.

eq	- Equal	= =
ne	- Not equal	~ =
lt	- Less than	<
gt	- Greater than	>
le	- Less than or equal	< =
ge	- Greater than or equal	> =

Logical operators.

and	- Logical AND	&
or	- Logical OR	|
not	- Logical NOT	~
xor	- Logical EXCLUSIVE OR	
any	- True if any element of vector is nonzero	
all	- True if all elements of vector are nonzero	

Special characters.

colon	- Colon	:
paren	- Parentheses and subscripting	( )
paren	- Parentheses and subscripting	( )

```
paren      - Brackets                          [ ]
paren      - Braces and subscripting           { }
paren      - Braces and subscripting           { }
punct      - Decimal point                     .
punct      - Structure field access            .
punct      - Parent directory                  ..
punct      - Continuation                      ...
punct      - Separator                         ,
punct      - Semicolon                         ;
punct      - Comment                           %
punct      - Invoke operating system command   !
punct      - Assignment                        =
punct      - Quote                             '
transpose  - Transpose                         .'
ctranspose - Complex conjugate transpose        '
horzcat    - Horizontal concatenation          [,]
vertcat    - Vertical concatenation            [;]
subsasgn   - Subscripted assignment            ( ),{ },.
subsref    - Subscripted reference             ( ),{ },.
subsindex  - Subscript index
```

Bitwise operators.

```
bitand     - Bit-wise AND.
bitcmp     - Complement bits.
bitor      - Bit-wise OR.
bitmax     - Maximum floating point integer.
bitxor     - Bit-wise XOR.
bitset     - Set bit.
bitget     - Get bit.
bitshift   - Bit-wise shift.
```

Set operators.

```
union      - Set union.
unique     - Set unique.
intersect  - Set intersection.
setdiff    - Set difference.
setxor     - Set exclusive-or.
ismember   - True for set member.
```

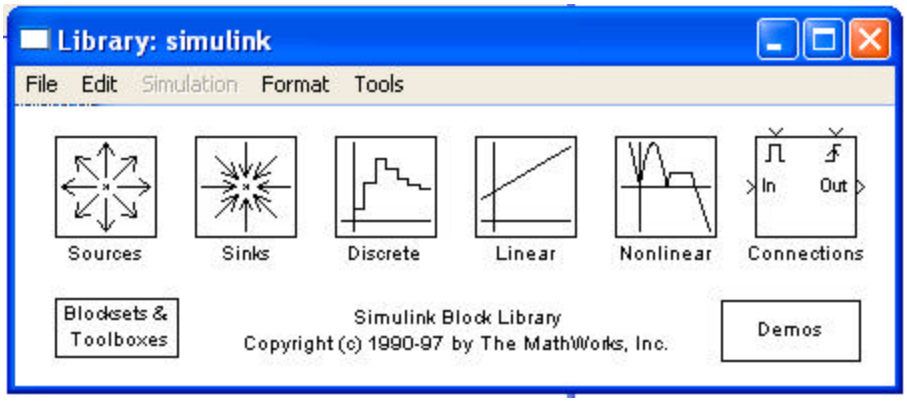
SIMULINK

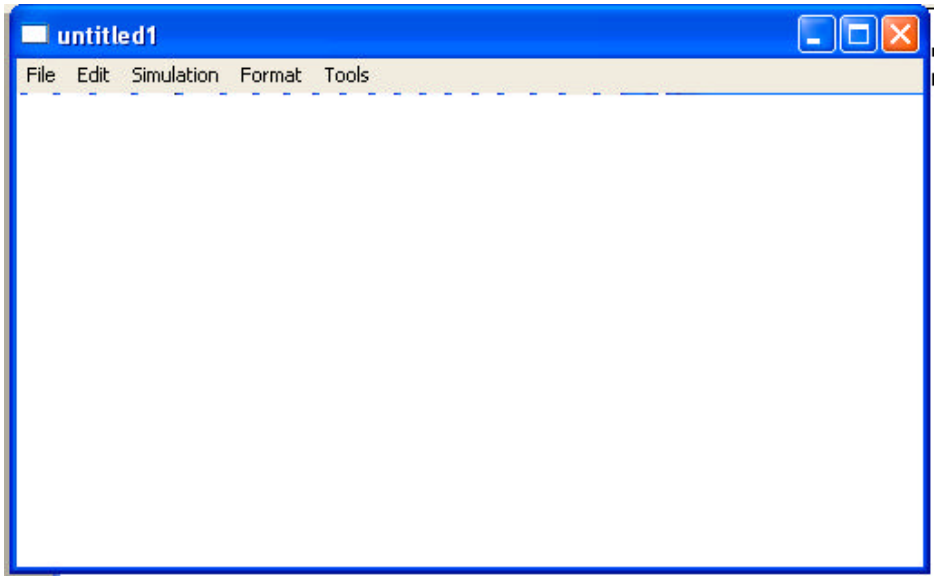
Simulink es una interface del Matlab usado para simular sistemas dinámicos. Se basa en sistemas gráficos de bloques.

El Icono del **New Simulink Model** es :

También puedes llamarlo por la ventana de comandos del Matlab de la siguiente forma: **» simulink** ¿

Al presionar el icono de **New Simulink Model** aparece dos ventanas, ellas son: **Library: simulink** y un **modelo (model)** con el nombre de **untitled1**, esta última sin titulo hasta que graves con un nombre en especial.

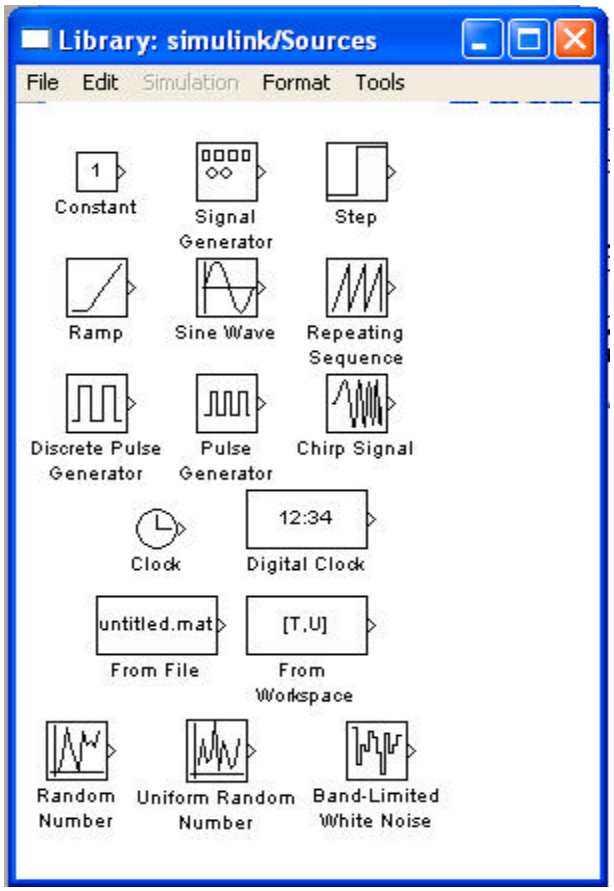


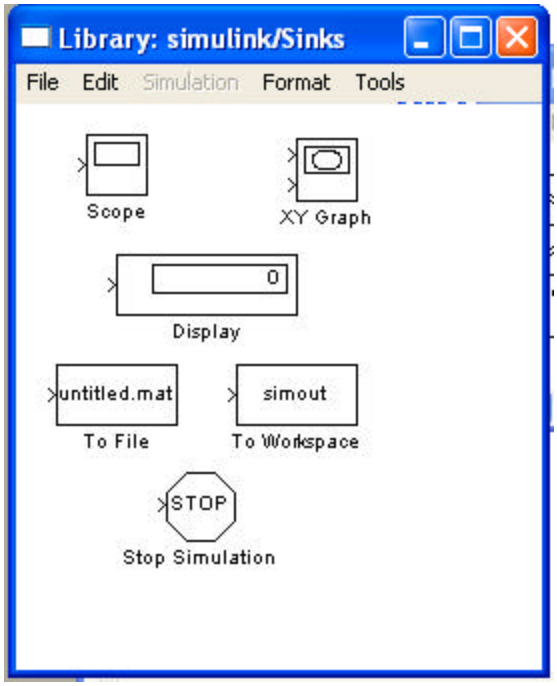


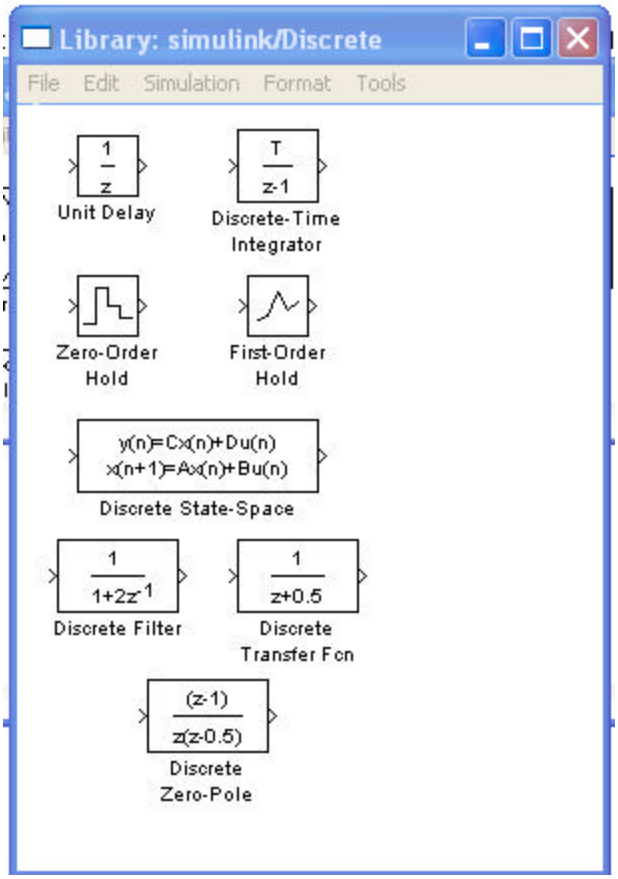
modelo (model)

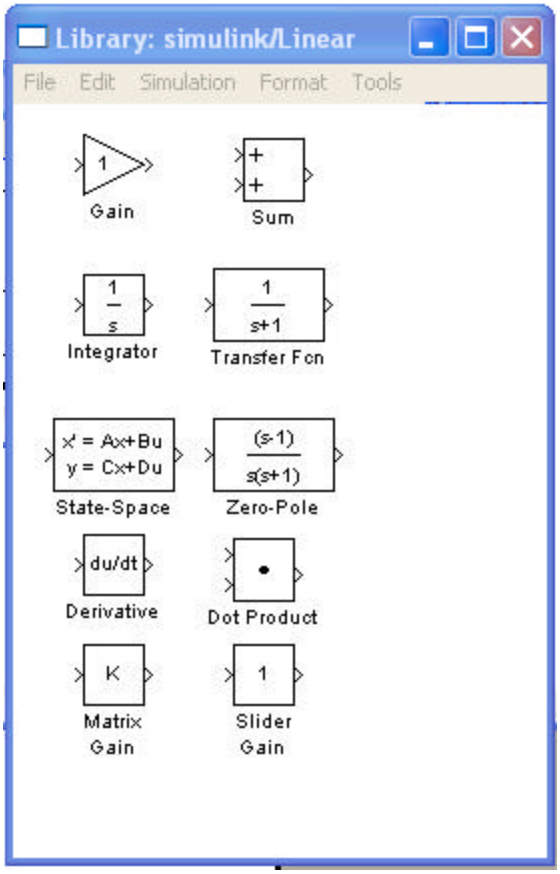
Debes tener un diagrama de sistema a diseñar y escoger entre los diferentes elementos que te brinda la ventana **Library:Simulink**, ejemplo: puedes escoger de ***source*** (Fuentes) el generador de escalón unitario (***step***) y presionando sobre su icono arrastrarlo hasta la ventana de trabajo **untitled1**. Luego escoger de la ventana ***Sink*** el icono ***Scope*** y arrastrarlo hacia la ventana de **untitled1.** Una vez que los dos elementos estén en la ventana de **untitled1** proceder a unirlos con una flecha. Para ello marque o presione con el botón del lado izquierdo del ratón la salida que representa el dibujo del ***step*** y sin soltar el botón arrástrelo hasta la entrada del **scope**, luego suelte el botón.

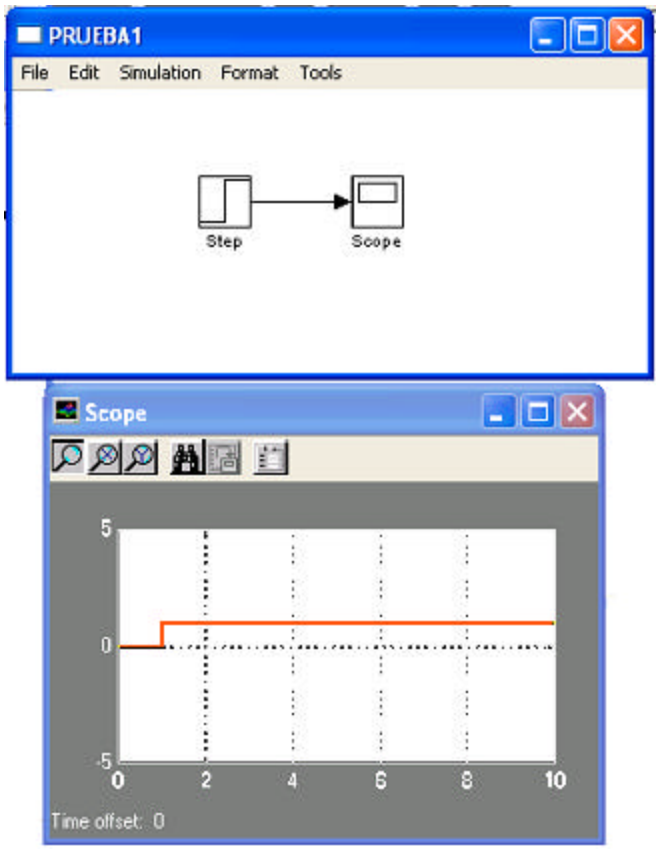
Una vez hecha la conexión proceda a su simulación, para ello marque **Simulation** de la parte superior de la ventana **Library:simulink** y aparecerá la representación gráfica de la salida del **step**.

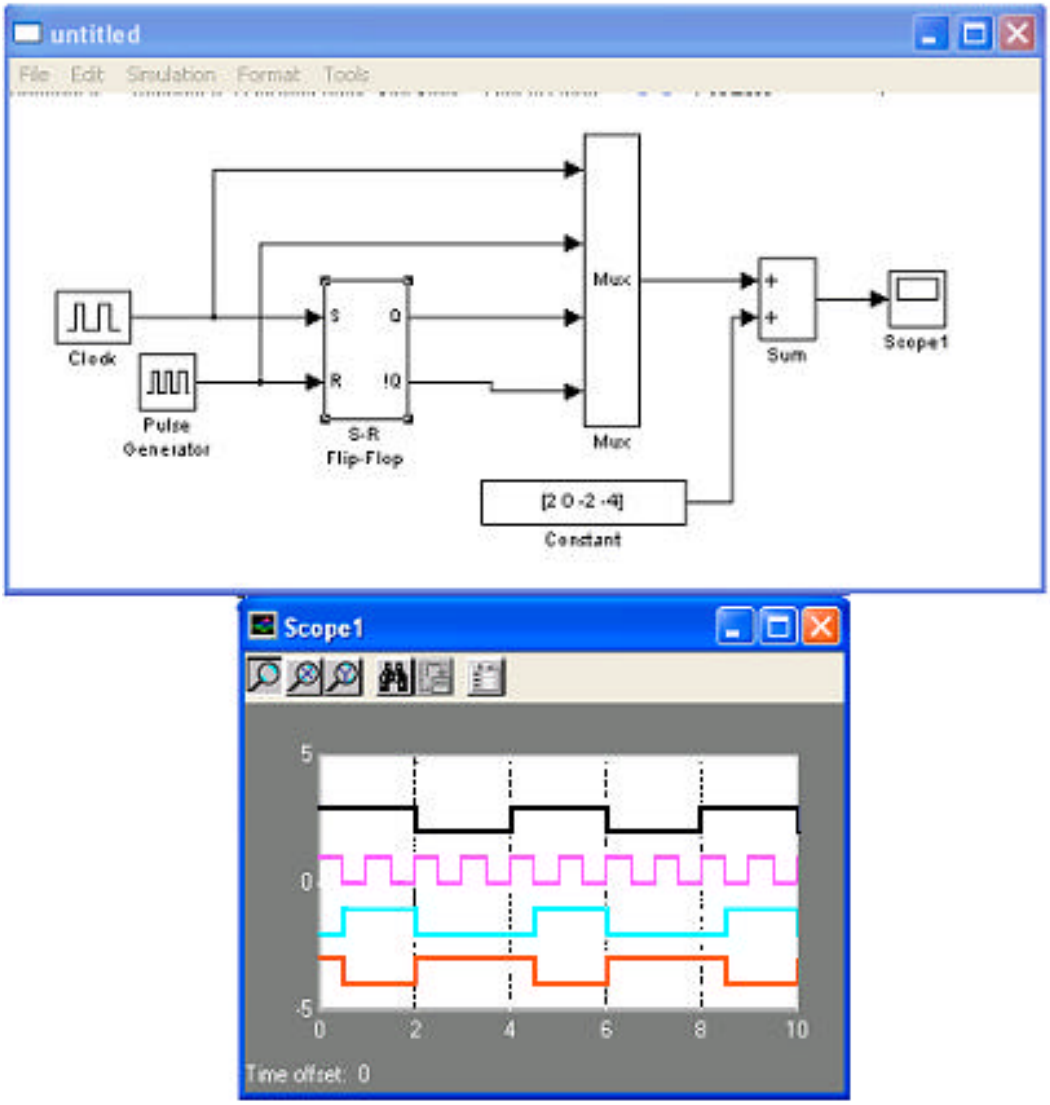





Library: simulink/Discrete
File Edit Simulation Format Tools
1/z
Unit Delay
T/(z-1)
Discrete-Time Integrator
Zero-Order Hold
First-Order Hold
y(n)=Cx(n)+Du(n)
x(n+1)=Ax(n)+Bu(n)
Discrete State-Space
1/(1+2z^-1)
Discrete Filter
1/(z+0.5)
Discrete Transfer Fcn
(z-1)/(z(z-0.5))
Discrete Zero-Pole


Library: simulink/Linear
File Edit Simulation Format Tools
1
Gain
Sum
1/s
Integrator
1/(s+1)
Transfer Fcn
x' = Ax+Bu
y = Cx+Du
State-Space
(s-1)/(s(s+1))
Zero-Pole
du/dt
Derivative
Dot Product
K
Matrix Gain
1
Slider Gain


PRUEBA1
File Edit Simulation Format Tools
Step
Scope
Scope
5
0
-5
0 2 4 6 8 10
Time offset: 0


untitled
File Edit Simulation Format Tools
Clock
Pulse Generator
S
Q
R
!Q
S-R Flip-Flop
Mux
Mux
+
+
Sum
Scope1
[2 0 -2 -4]
Constant
Scope1
5
0
-5
0
2
4
6
8
10
Time offset: 0