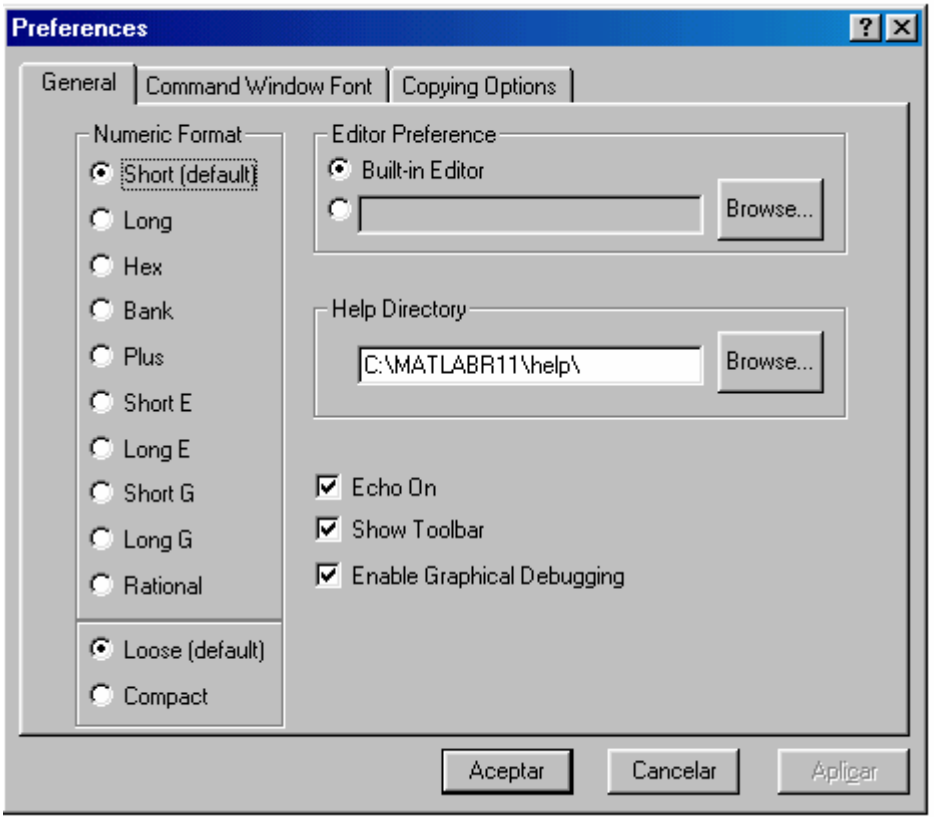
El Matlab se basa fundamentalmente en cálculos vectoriales y matriciales mientras que el Simulink se basa en cálculos matemático trabajados en forma visual llamados BLOQUES.

Teoría: MATLAB.

Control de los formatos de salida:

Los formatos de salida en la ventana principal de MATLAB se pueden controlar fácilmente a partir del cuadro de diálogo que se abre con el comando ***Preferences*** del menú ***File*.**



format short coma fija con 4 decimales ***(defect*o)**
format long coma fija con 15 decimales
format hex cifras hexadecimales
format bank números con dos cifras decimales
format short e notación científica con 4 decimales
format short g notación científica o decimal, dependiendo del valor
format long e notación científica con 15 decimales
format long g notación científica o decimal, dependiendo del valor
format loose introduce algunas líneas en blanco en la salida (defecto)
format compact elimina las líneas en blanco citadas ***(opción recomendad*a)**
format rat expresa los números racionales como cocientes de enteros

Cálculos censillos:

10+20 = 30	10-20 = -10	10*20 = 200	10/20 = 0.5
» 10+20 ans = 30	» 10-20 ans = -10	» 10*20 ans = 200	» 10/20 ans = 0.5000

Como definir una matriz:

Forma conocida:

$$A = \begin{bmatrix} 10 & 34 & -89 \\ -8 & 85 & 35 \\ 4 & 13 & 23 \end{bmatrix}$$

Forma de ingresarla al Matlab:

```
» A=[10 34 -89; -8 85 35; 4 13 23]
```

Forma de visualizarla dentro del Matlab:

```
A =
    10    34   -89
    -8    85    35
     4    13    23
```

Como extraer de una matriz un dato específico o una serie de datos:

Si queremos extraer valores:

10		34		-89	
» A(1) ans = 10	» A(1,1) ans = 10	» A(4) ans = 34	» A(1,2) ans = 34	» A(7) ans = -89	» A(1,3) ans = -89
-8		85		35	
» A(2) ans = -8	» A(2,1) ans = -8	» A(5) ans = 85	» A(2,2) ans = 85	» A(8) ans = 35	» A(2,3) ans = 35
4		13		23	
» A(3) ans = 4	» A(3,1) ans = 4	» A(6) ans = 13	» A(3,2) ans = 13	» A(9) ans = 23	» A(3,3) ans = 23

Definamos una matriz cualquiera con el comando MAGIC:

```
» A=magic(4)
A =
    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1
```

```
» A([2 4],[1 2])
ans =
     5    11
     4    14
```

Si queremos extraer datos específicos:

Ejemplo: extraer la intelección de las rectas.

16	2	3	13
2	11	10	8
9	7	6	12
4	14	15	1

Como formar un vector con un comando de secuencia:

Supongamos que queremos una secuencia que parta de “0” y llegue a “20” con salto de una unidad.

$$A = [0 \quad 1 \quad 2 \quad 3 \quad .... \quad 20]$$

```
» C=(0:1:20)
C =
  Columns 1 through 12
     0     1     2     3     4     5     6     7     8     9    10    11
  Columns 13 through 21
    12    13    14    15    16    17    18    19    20
```

Operación con vectores columnas:

$$A = \begin{bmatrix} 125 \\ 785 \\ 854 \\ 276 \end{bmatrix} \quad B = \begin{bmatrix} 745 \\ 368 \\ 175 \\ 942 \end{bmatrix}$$

Definimos las dos matrices A y B:

```
» A=[125;785;854;276]
A =
   125
   785
   854
   276
```

```
» B=[745;368;175;942]
B =
   745
   368
   175
   942
```

Suma	Resta	Multiplicación	División
» A+B ans = 870 1153 1029 1218	» A-B ans = -620 417 679 -666	» A.*B ans = 93125 288880 149450 259992	» A./B ans = 0.1678 2.1332 4.8800 0.2930

TIPOS DE MATRICES PREDEFINIDOS:

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)

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)

FORMACIÓN DE UNA MATRIZ A PARTIR DE OTRAS:

MATLAB ofrece también la posibilidad de crear una matriz a partir de matrices previamente ya definidas

[m,n]=size(A) devuelve el número de filas y de columnas de la matriz A. Si la matriz es cuadrada basta recoger el primer valor de retorno

n=length(x) calcula el número de elementos de un vector x

zeros(size(A)) forma una matriz de ceros del mismo tamaño que una matriz A previamente creada

ones(size(A)) ídem con unos

A=diag(x) forma una matriz diagonal A cuyos elementos diagonales son los elementos de un vector ya existente x

x=diag(A) forma un vector x a partir de los elementos de la diagonal de una matriz ya existente A

diag(diag(A)) crea una matriz diagonal a partir de la diagonal de la matriz A

blkdiag(A,B) crea una matriz diagonal de submatrices a partir de las matrices que se le pasan como argumentos

triu(A) forma una matriz triangular superior a partir de una matriz A (no tiene por qué ser cuadrada). Con un segundo argumento puede controlarse que se mantengan o eliminen más diagonales por encima o debajo de la diagonal principal.

tril(A) ídem con una matriz triangular inferior

rot90(A,k) Gira k*90 grados la matriz rectangular A en sentido antihorario. k es un entero que puede ser negativo. Si se omite, se supone k=1

flipud(A) halla la matriz simétrica de A respecto de un eje horizontal

fliplr(A) halla la matriz simétrica de A respecto de un eje vertical

reshape(A,m,n) Cambia el tamaño de la matriz A devolviendo una matriz de tamaño $m \times n$ cuyas columnas se obtienen a partir de un vector formado por las columnas de A puestas una a continuación de otra. Si la matriz A tiene menos de $m \times n$ elementos se produce un error.

FUNCIONES MATRICIALES ELEMENTALES:

B = A' calcula la traspuesta (conjugada) de la matriz **A**

B = A.' calcula la traspuesta (sin conjugar) de la matriz **A**

v = poly(A) devuelve un vector **v** con los coeficientes del polinomio característico de la matriz cuadrada **A**

t = trace(A) devuelve la traza **t** (suma de los elementos de la diagonal) de una matriz cuadrada **A**

[m,n] = size(A) devuelve el número de filas **m** y de columnas **n** de una matriz rectangular **A**

n = size(A) devuelve el tamaño de una matriz cuadrada **A**

nf = size(A,1) devuelve el número de filas de **A**

nc = size(A,2) devuelve el número de columnas de **A**

FUNCIONES MATRICIALES ESPECIALES:

Las funciones exp(), sqrt() y log() se aplican elemento a elemento a las matrices y/o vectores que se les pasan como argumentos. Existen otras funciones similares que tienen también sentido cuando se aplican a una matriz como una única entidad. Estas funciones son las siguientes (se distinguen porque llevan una "m" adicional en el nombre):

expm(A) si A=XDX', expm(A) = X*diag(exp(diag(D)))*X'

sqrtm(A) devuelve una matriz que multiplicada por sí misma da la matriz **A**

logm() es la función recíproca de expm(A)

Aunque no pertenece a esta familia de funciones, se puede considerar que el ***operador potencia*** (^) está emparentado con ellas. Así, es posible decir que:

A^n está definida si **A** es cuadrada y **n** un número real. Si **n** es entero, el resultado se calcula por multiplicaciones sucesivas. Si **n** es real, el resultado se calcula como: A^n=X*D.^n*X' siendo [X,D]=eig(A)

FUNCIONES DE FACTORIZACIÓN Y/O DESCOMPOSICIÓN MATRICIAL:

A su vez este grupo de funciones se puede subdividir en 4 subgrupos:

– Funciones basadas en la factorización triangular (eliminación de Gauss): [L,U] = lu(A) descomposición de Crout (A = LU) de una matriz. La matriz **L** es una permutación de una matriz

triangular inferior (dicha permutación es consecuencia del pivotamiento por columnas utilizado en la factorización)

B = inv(A) calcula la inversa de A. Equivale a B=inv(U)*inv(L)

d = det(A) devuelve el determinante **d** de la matriz cuadrada A. Equivale a d=det(L)*det(U)

E = rref(A) reducción a forma de escalón (mediante la eliminación de Gauss con pivotamiento por columnas) de una matriz rectangular **A**

[E,xc] = rref(A) reducción a forma de escalón con un vector **xc** que da información sobre una posible base del espacio de columnas de **A**

U = chol(A) descomposición de Cholesky de matriz simétrica y positivo-definida.

Sólo se utiliza la diagonal y la parte triangular superior de A. El resultado es una matriz triangular superior tal que A = U'*U

c = rcond(A) devuelve una estimación del recíproco de la condición numérica de la matriz **A** basada en la norma sub-1. Si el resultado es próximo a 1 la matriz **A** está bien condicionada; si es próximo a 0 no lo está.

– Funciones basadas en el cálculo de valores y vectores propios:

[X,D] = eig(A) valores propios (diagonal de D) y vectores propios (columnas de X) de una matriz cuadrada A. Con frecuencia el resultado es complejo (si **A** no es simétrica)

[X,D] = eig(A,B) valores propios (diagonal de D) y vectores propios (columnas de X) de dos matrices cuadradas **A** y **B** (**Ax** = **lB**x).

– Funciones basadas en la descomposición QR:

[Q,R] = qr() descomposición QR de una matriz rectangular. Se utiliza para sistemas con más ecuaciones que incógnitas.

B = null(A) devuelve una base ortonormal del subespacio nulo (kernel, o conjunto de vectores **x** tales que **Ax** = 0) de la matriz rectangular **A**

Q = orth(A) las columnas de **Q** son una base ortonormal del espacio de columnas de A. El número de columnas de **Q** es el rango de **A**

– Funciones basadas en la descomposición de valor singular

[U,D,V] = svd(A) descomposición de valor singular de una matriz rectangular (A=U*D*V'). **U** y **V** son matrices ortonormales. **D** es diagonal y contiene los valores singulares

B = pinv(A) calcula la pseudo-inversa de una matriz rectangular **A**

r = rank(A) calcula el rango **r** de una matriz rectangular **A**

nor = norm(A) calcula la norma sub-2 de una matriz (el mayor valor singular)

nor = norm(A,2) lo mismo que la anterior

c = cond(A) condición numérica sub-2 de la matriz A. Es el cociente entre el máximo y el mínimo valor singular. La condición numérica da una idea de los errores que se obtienen al resolver un sistema

de ecuaciones lineales con dicha matriz: su logaritmo indica el número de cifras significativas que se pierden.

Funciones matemáticas elementales que operan de modo escalar:

Estas funciones, que comprenden las funciones matemáticas trascendentales y otras funciones básicas, actúan sobre cada elemento de la matriz como si se tratase de un escalar. 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 -p/2 y +p/2)

atan2(x) arco tangente (devuelve un ángulo entre -p y +p); se le pasan 2 argumentos, proporcionales al seno y al coseno

sinh(x) seno hiperbólico

cosh(x) coseno hiperbólico

tanh(x) tangente hiperbólica

asinh(x) arco seno hiperbólico

acosh(x) arco coseno hiperbólico

atanh(x) arco tangente hiperbólica

log(x) logaritmo natural

log10(x) logaritmo decimal

exp(x) función exponencial

sqrt(x) raíz cuadrada

sign(x) devuelve -1 si <0, 0 si =0 y 1 si >0. Aplicada a un número complejo, devuelve un vector unitario en la misma dirección

rem(x,y) resto de la división (2 argumentos que no tienen que ser enteros)

mod(x,y) similar a rem (Ver diferencias con el Help)

round(x) redondeo hacia el entero más próximo

fix(x) redondea hacia el entero más próximo a 0

floor(x) valor entero más próximo hacia -

ceil(x) valor entero más próximo hacia +

gcd(x) máximo común divisor

lcm(x) mínimo común múltiplo

real(x) partes reales

imag(x) partes imaginarias

abs(x) valores absolutos

angle(x) ángulos de fase

Funciones que actúan sobre vectores:

Las siguientes funciones actúan sobre vectores (no sobre matrices ni sobre escalares)

[xm,im]=max(x) máximo elemento de un vector. Devuelve el valor máximo xm y la posición que ocupa im

min(x) mínimo elemento de un vector. Devuelve el valor mínimo y la posición que ocupa

sum(x) suma de los elementos de un vector

cumsum(x) devuelve el vector suma acumulativa de los elementos de un vector

mean(x) valor medio de los elementos de un vector

std(x) desviación típica

prod(x) producto de los elementos de un vector

cumprod(x) devuelve el vector producto acumulativo de los elementos de un vector

[y,i]=sort(x) ordenación de menor a mayor de los elementos de un vector x.

Devuelve el vector ordenado y, y un vector i con las posiciones iniciales en x de los elementos en el vector ordenado y.

Más sobre operadores relacionales con vectores y matrices:

Cuando alguno de los operadores relacionales vistos previamente (<, >, <=, >=, == y ~=) actúa entre dos matrices (vectores) del mismo tamaño, el resultado es otra matriz (vector) de ese mismo tamaño conteniendo unos y ceros, según los resultados de cada comparación entre elementos hayan sido true o false, respectivamente.

Por ejemplo, supóngase que se define una matriz magic A de tamaño 3x3 y a continuación se forma una matriz binaria M basada en la condición de que los elementos de A sean mayores que 4 (MATLAB convierte este cuatro en una matriz de cuatros de modo automático). Obsérvese con atención el resultado:.Capítulo 3: Funciones de librería página 31

```
» A=magic(3)
A =
      8 1 6
      3 5 7
      4 9 2
```

```
» M=A>4
M =
      1 0 1
      0 1 1
      0 1 0
```

De ordinario, las matrices "binarias" que se obtienen de la aplicación de los operadores relacionales no se almacenan en memoria ni se asignan a variables, sino que se procesan sobre la marcha. MATLAB dispone de varias funciones para ello. Recuérdese que cualquier valor distinto de cero equivale a true, mientras que un valor cero equivale a false. Algunas de estas funciones son:

any(x) función vectorial; chequea si alguno de los elementos del vector x cumple una determinada condición (en este caso ser distinto de cero). Devuelve un uno ó un cero

any(A) se aplica por separado a cada columna de la matriz A. El resultado es un vector de unos y ceros

all(x) función vectorial; chequea si todos los elementos del vector x cumplen una condición. Devuelve un uno ó un cero

all(A) se aplica por separado a cada columna de la matriz A. El resultado es un vector de unos y ceros

find(x) busca índices correspondientes a elementos de vectores que cumplen una determinada condición. El resultado es un vector con los índices de los elementos que cumplen la condición

find(A) cuando esta función se aplica a una matriz la considera como un vector con una columna detrás de otra, de la 1ª a la última.

A continuación se verán algunos ejemplos de utilización de estas funciones.

```
» A=magic(3)
A =
     8 1 6
     3 5 7
     4 9 2
```

```
» m=find(A>4)
m =
     5
     6
     7
     8
```

Comandos inteligentes o comandos en bloques:

Comando: switch

Primero determinamos la variable “a” ejemplo a=’febrero’

Luego creamos el comando w1:

```
switch a
case 'enero',s='$1500'
case 'febrero',s='$1150'
case 'marzo',s='$1150'
case {'junio','julio','agosto'},s='$7854'
otherwise ,s='no se sabe'
end
```

-Nota: En este comande al definir la variable “a” esta debe ser exacta a las previamente definidas o de lo contrario actuara el “`otherwise`”

Comando: if

Primero determinamos la variable “a” ejemplo a=15

Luego creamos el comando w2:

```
if a<12,s='$1,00'
elseif a>=12&a<=60,s='$2,50'
elseif a>60,s='$1,50'
end
```

-Nota: En este comando podemos definir extremos y rangos.

Comando: for

Primero determinamos las variables
"o" el comienzo o=1
"r" el intervalo r=1
"n" el fin n=5
Luego creamos el comando w3:

```
for i=o:r:f
   i^2
end
```

-Nota: En este comando podemos definir un rango y un intervalo para luego poder evaluarlo en el intervalo que se especifico

---

Comando: while

Este comando girará bucles hasta que la variable "a" es igual a cero.
Luego creamos el comando w4:

```
a=10
while a
  a=a-0.1
  if a<0,a=0
  end
```

-Nota: este comando se utiliza par generar cálculos repetitivos hasta llegar a un valor determinado y finalizar el bucle.

Polinomios

Los polinomios en Matlab son presentados por vectores filas conteniendo los coeficientes en orden decreciente

- Ejemplo: x4 - 12x3 + 0x2 + 25x +116

p=[1 -12 0 25 116]

Funciones para polinomios:

- Raíces del polinomio: **roots(p)**
- Polinomio asociado a unas raíces: **poly(p)**
- Multiplica dos polinomios: **conv(p1,p2)**
- Divide dos polinomios: **deconv(p1,p2)**
- Cálculo de derivadas: **polyder(p)**
- Cálcula el polinomio mas sercano: **polyfit(p,n)**
- Evaluación de polinomios: **polyval(p,n)**
- Desarrollo en fracciones parciales: **residue(p)**

Gráficos 2D:

- Comando **plot**
- Tipos de líneas y colores
- Añadir rejillas y etiquetas
- Gestión de los ejes
- Manipulando gráficos 2D
- Otros tipos de gráficos 2D

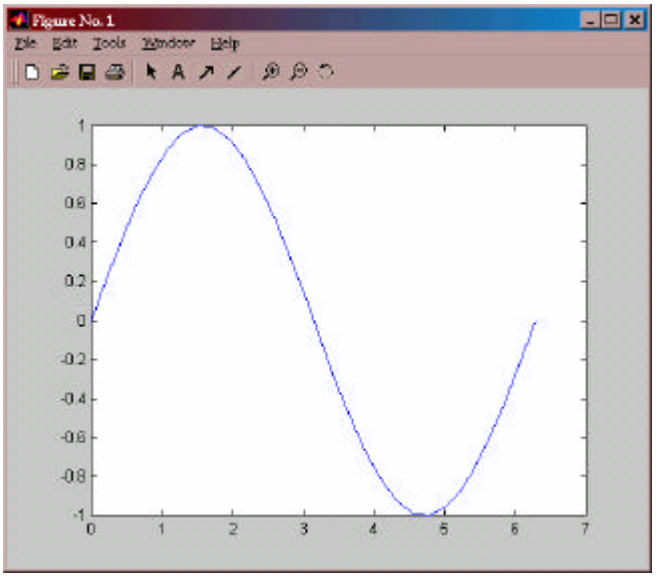
Comando plot:

- Es el comando más utilizado para gráficos en 2D
- Representa gráficamente conjuntos de *arrays* de datos:
- Elige automáticamente los ejes apropiados
- Por defecto, conecta los datos mediante líneas rectas

Ejemplo del comando plot:

Dibujar la función y=sin(x) en donde x es un *array* distrubuido uniformemente ente 30 valores de [0 a 2p]

```
» x=linspace(0,2*pi,30)
» y=sin(x)
» plot(x,y)
```



Ejemplo de superposición de graficas:

Dibujar la función y=sin(x) y z=cos(x)

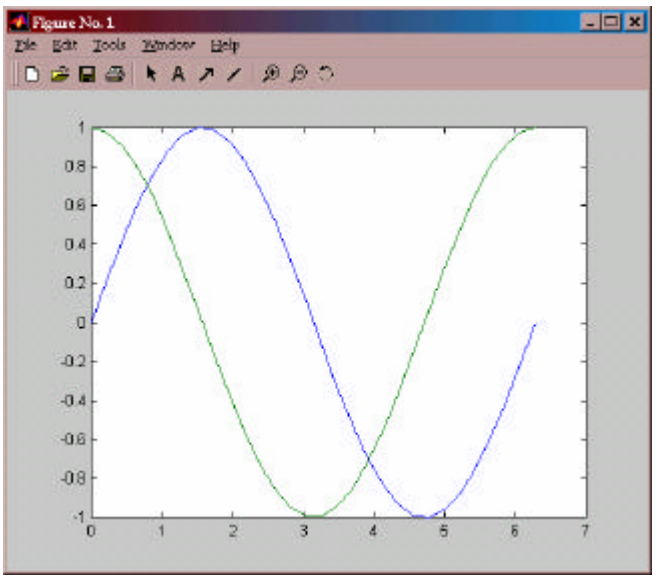
```
» x=linspace(0,2*pi,30)
» y=sin(x)
» z=cos(x)
» plot(x,y,x,z)
```

o tambien

```
» x=linspace(0,2*pi,30)
» y=sin(x)
» z=cos(x)
» W=[y;z]
» plot(x,W)
```

o tambien

```
» x=linspace(0,2*pi,30)
» y=sin(x)
» plot(x,y)
» hold on
» z=cos(x)
» plot(x,z)
```



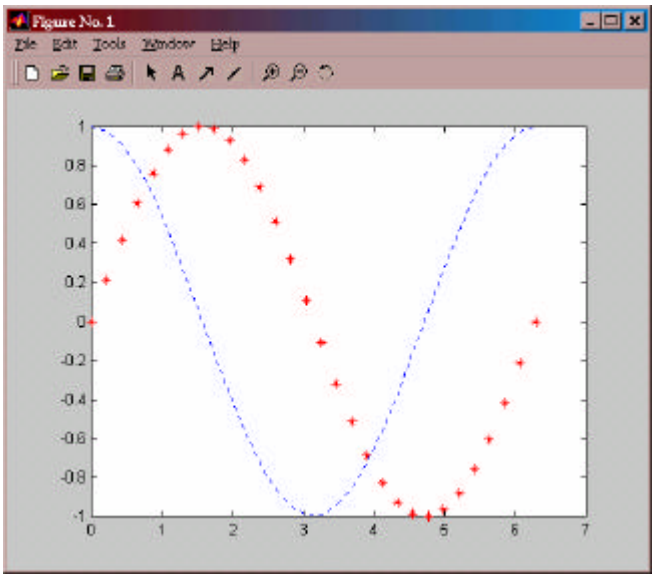
Tipos de líneas y colores:

Colores	Tipos de líneas
■ y Amarillo	■ . Puntos
■ m Magenta	■ o Círculos
■ c Cyan	■ x Marcas x
■ r Rojo	■ + Marcas +
■ g Verde	■ * Marcas *
■ b Azul	■ - Línea continua
■ w Blanco	■ : Línea punteada
■ k Negro	■ -. Líneas y puntos
	■ -- Líneas discontinuas

Ejemplo de colores y linas:

Dibujar la función y=sin(x) y z=cos(x)

```
» x=linspace(0,2*pi,30)
» y=sin(x)
» z=cos(x)
» plot(x,y,'r*',x,z,'b:')
```



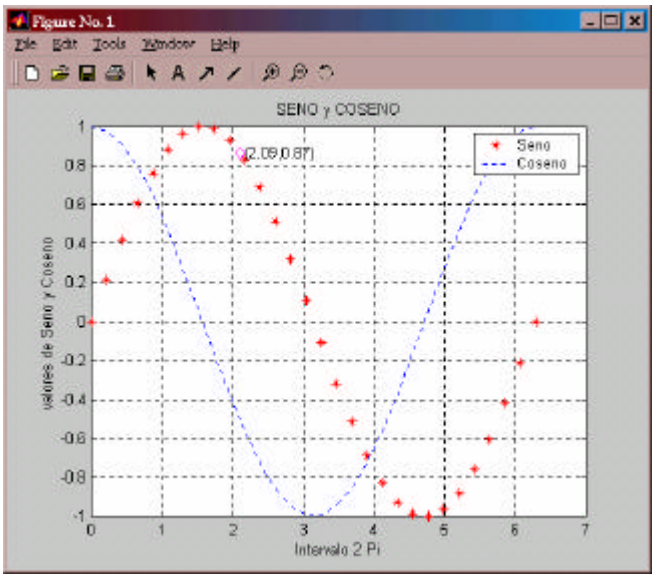
Añadir rejillas y etiquetas

- Añadir rejillas: grid
- Etiquetar eje x: xlabel(‘texto’)
- Etiquetar eje y: ylabel(‘texto’)
- Añadir título: title(‘texto’)
- Texto en un punto especifico : text(x,y,’texto’)
- Texto en un punto especifico determinado por el MOUSE: gtext(‘texto’)
- Leyenda: legend(‘var1’,....,’varn’)

Ejemplo de rejillas y etiquetas

Dibujar la función y=sin(x) y z=cos(x)

```
» x=linspace(0,2*pi,30)
» x=linspace(0,2*pi,30);
» y=sin(x);
» plot(x,y);
» z=cos(x);
» plot(x,y,'r*',
x,z,'b:',2/3*pi,
sin(2/3*pi),'mo')
» grid
» xlabel('Intervalo 2 Pi' )
» ylabel('valores de Seno y Coseno')
» title('SENO y COSENO')
» text(2/3*pi,sin(2/3*pi),' (2.09,0.87)')
» legend('Seno','Coseno')
```



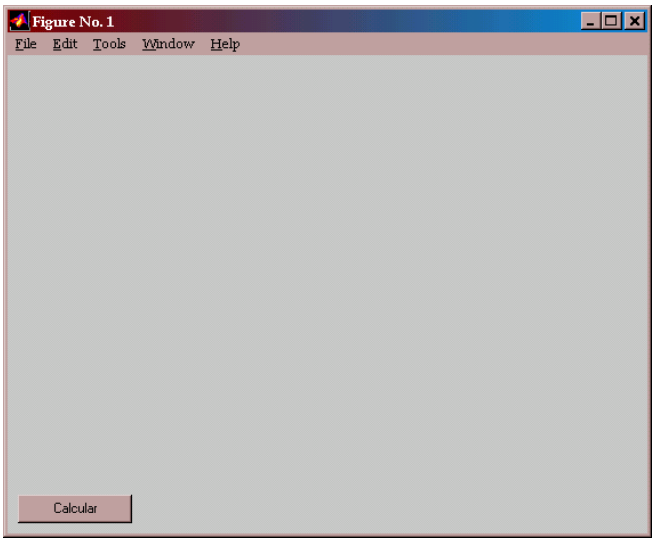
Comandos de programación:

- BOTONES (PUSHBUTTONS)
- BOTONES DE SELECCIÓN (CHECK BOXES)
- TEXTO (LABEL)
- BOTONES DE OPCIÓN (RADIO BUTTONS)
- BARRAS DE DESPLAZAMIENTO (SCROLLING BARS O SLIDERS)
- CAJAS DE SELECCIÓN DESPLEGABLES (POP-UP MENUS)
- CAJAS DE TEXTO EDITABLES (EDITABLE TEXTBOXES)
- MARCOS (FRAMES)

BOTONES (PUSHBUTTONS):

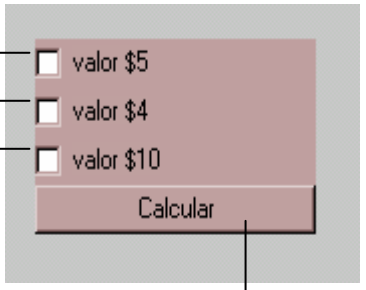
Funcion:

```
boton_calculo = uicontrol(gcf,...
'Style','push',...
'Position',[10 10 100 25],...
'String','Calcular',...
'CallBack','a+b');
```



BOTONES DE SELECCIÓN (CHECK BOXES)

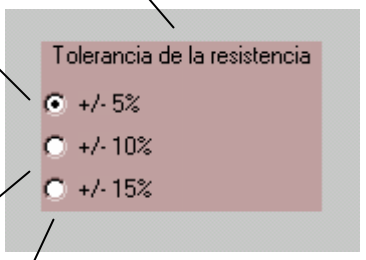
```
a=0,b=0,c=0,d=0,e=0,f=0,
Box_01 = uicontrol(gcf,...
'Style','checkbox',...
'Units','normalized','Position',[0.4 0.600 0.25 0.05],...
'String','valor $5',...
'CallBack',['a=b;if a==0,b=5;else a==5,b=0;end;b']);

Box_02 = uicontrol(gcf,...
'Style','checkbox',...
'Units','normalized','Position',[0.4 0.550 0.25 0.05],...
'String','valor $4',...
'CallBack',['c=d;if c==0,d=4;else c==4,d=0;end;d']);

Box_03 = uicontrol(gcf,...
'Style','checkbox',...
'Units','normalized','Position',[0.4 0.500 0.25 0.05],...
'String','valor $10',...
'CallBack',['e=f;if e==0,f=10;else e==10,f=0;end;f']);

boton_calculo = uicontrol(gcf,...
'Style','push',...
'Units','normalized','Position',[0.4 0.450 0.25 0.05],...
'String','Calcular',...
'CallBack','b+d+f');
```



TEXTO (LABEL)

BOTONES DE OPCIÓN (RADIO BUTTONS)

```
% Definir el texto de título para este grupo de
controles
txt_01 = uicontrol(gcf,...
'Style','text','String','Tolerancia de la resistencia',...
'Units','normalized','Position',[0.4 0.60 0.25 0.05]);

% Definir la propiedad TickDir In con radi obutton (defecto)
tol=5
td_5 = uicontrol(gcf,...
'Style','radio', 'String','+/- 5%',...
'Units','normalized','Position',[0.4 0.55 0.25 0.05],...
'Value',1,...
'CallBack','set(td_5,"Value",1),set(td_10,"Value",0),set(td_1
5,"Value",0),1,0,0,tol=5');

% Definir la propiedad TickDir Out con radiobutton
td_10 = uicontrol(gcf,...
'Style','radio','String','+/- 10%',...
'Units','normalized','Position',[0.4 0.50 0.25 0.05],...
'Value',0,...
'CallBack','set(td_5,"Value",0),set(td_10,"Value",1),set(td_1
5,"Value",0),0,1,0,tol=10');

% Definir la propiedad TickDir Out con radiobutton
td_15 = uicontrol(gcf,...
'Style','radio','String','+/- 15%',...
'Units','normalized','Position',[0.4 0.45 0.25 0.05],...
'Value',0,...
'CallBack','set(td_5,"Value",0),set(td_10,"Value",0),set(td_1
5,"Value",1),0,0,1,tol=15');
```

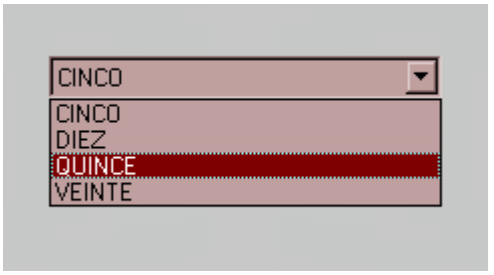


BARRAS DE DESPLAZAMIENTO (SCROLLING BARS O SLIDERS)

```
barra_01 = uicontrol(gcf,...
'Style','slider',...
'Units','normalized','Position',[0.4 0.55 0.35 0.05],...
'Min',0,'Max',20000,'Value',5,...
'CallBack',['a=num2str(get(barra_01,"Val"))']);
```

CAJAS DE SELECCIÓN DESPLEGABLES (POP-UP MENUS)

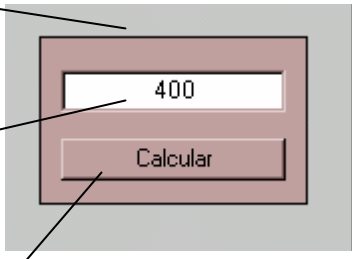
```
popcol = uicontrol(gcf,...
'Style','popup',...
'String','CINCO|DIEZ|QUINCE|VEINTE',...
'Units','normalized','Position',[0.4 0.55 0.35 0.05],...
'CallBack',['pop=[5,10,15,20];',...
'a=pop(get(popcol,"Value"))']);
```



CAJAS DE TEXTO EDITABLES (EDITABLE TEXTBOXES)

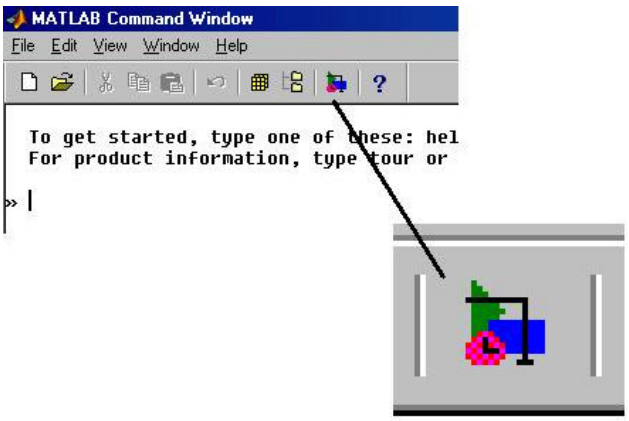
MARCOS (FRAMES)

```
ft_dir = uicontrol(gcf,...
'Style','frame',...
'Units','normalized','Position',[0.38 0.49 0.24 0.20]);

valor_01 = uicontrol(gcf,...
'Style','edit',...
'BackgroundColor','white',...
'FontSize',9,'FontName','Arial',...
'String',[400],...
'Units','normalized','Position',[0.4 0.60 0.20 0.05],...
'Max',100,...
'CallBack','r=str2double(get(edmulti,"String"))');

boton_calculo = uicontrol(gcf,...
'Style','push',...
'Units','normalized','Position',[0.4 0.52 0.20 0.05],...
'String','Calcular',...
'CallBack','r*3');
```



Teoría: SIMULINK.

Este programa se basa en la utilización de comandos en sistemas de bloques, cada bloque ejecuta un comando matemático.

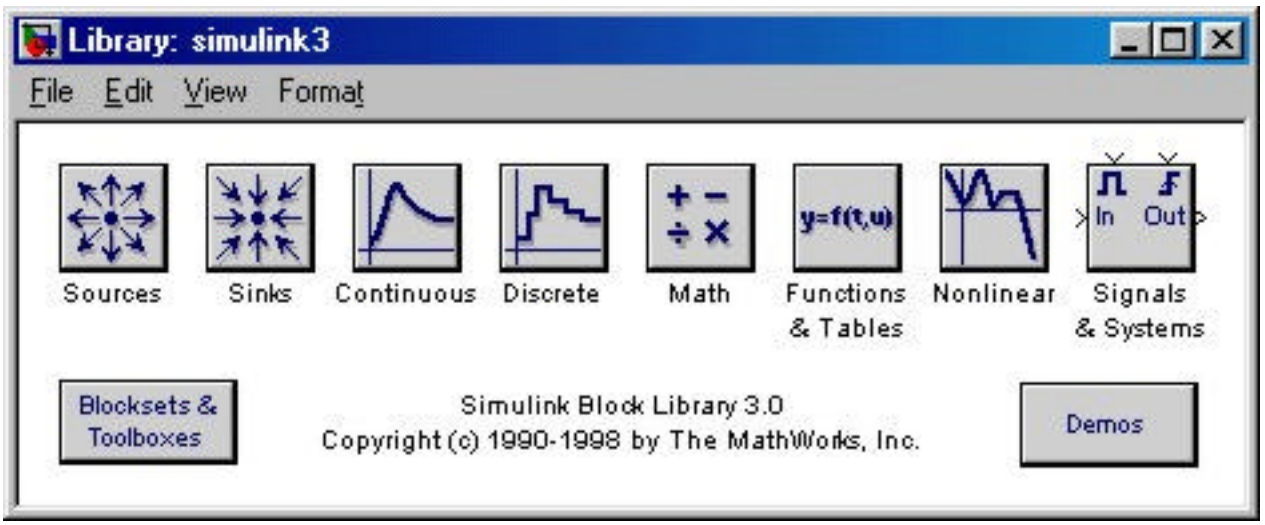


Dentro del entorno de Matlab se encuentra el botón del Simulink llamado "Simulink Liberaly Browser"

Esta ventana muestra las distintas galerías donde se encuentran los comandos de bloques.

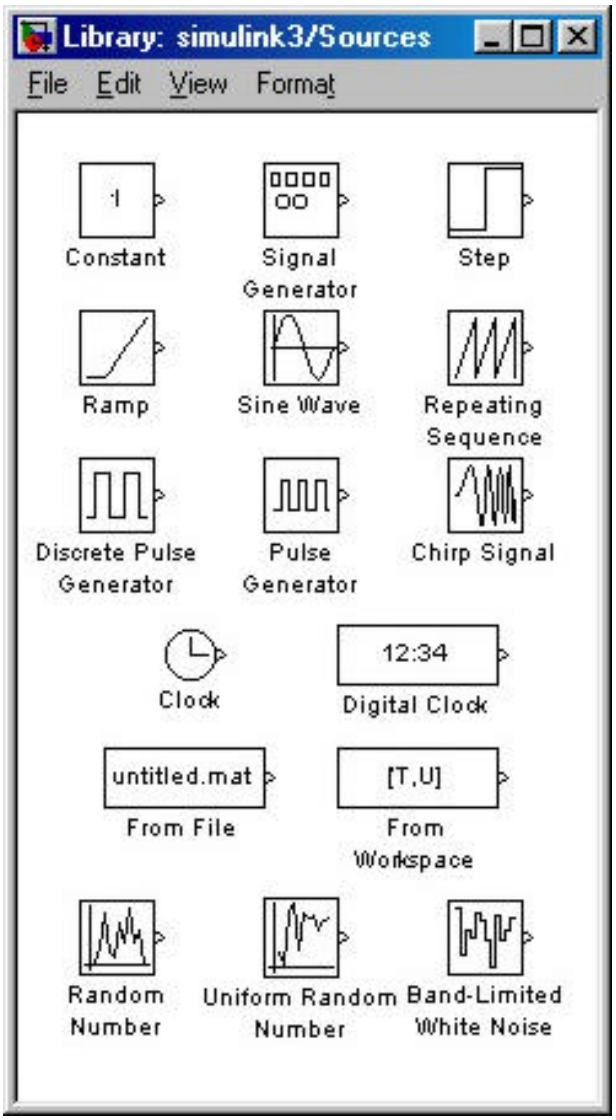
En este curso no profundizaremos en todas las galerías ni en todos los comandos ya que por razones de tiempo y necesidad de uso solo utilizaremos los sistemas de bloques más necesarios para Ingeniería Eléctrica.

Comenzaremos con la galería Simulink:



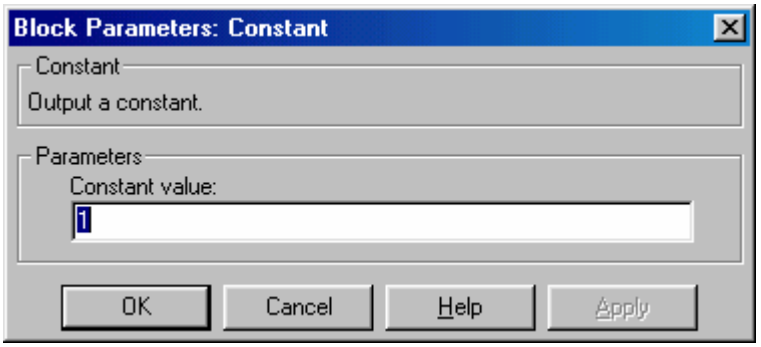
En esta galería se encuentran casi todos los comandos básicos de entrad procesamiento y calidad de datos.

La galería SOUCES:



Los bloques que se encuentran dentro son los bloques de entrada, los generadores de señales.

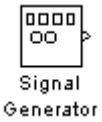
Constant

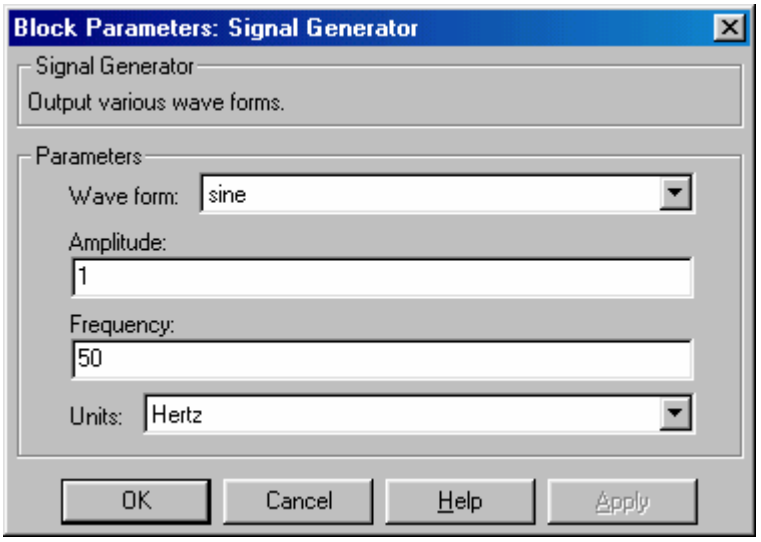


Este bloque se utiliza para ingresar constantes.

Constant value: Aquí se ingresa la contante.

Signal Generator

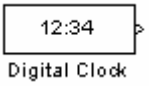




Este bloque se utiliza para generar distintos tipos de señales.

Wave form: El tipo de señal
Amplitude: La amplitud de la señal
Frequency: La frecuencia de la señal
Units: Herts o rad/seg

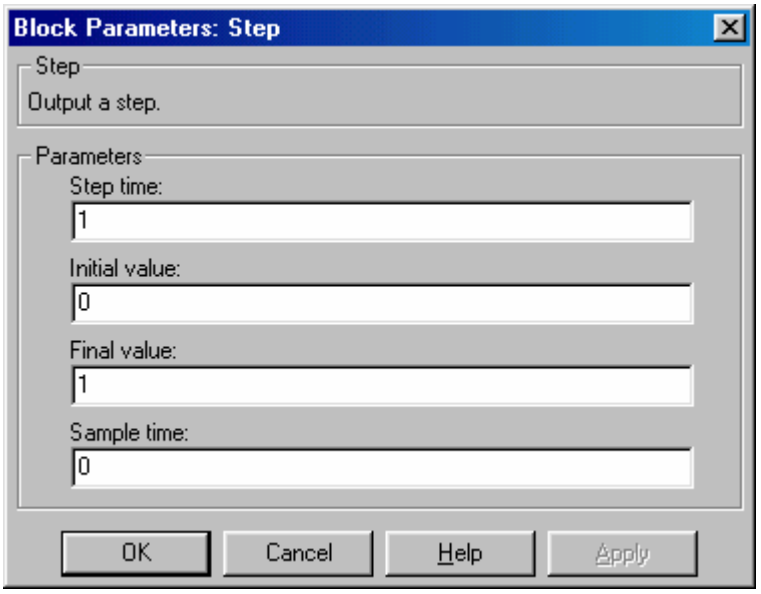
Clock and Digital Clock



Este bloque se utiliza para generar una señal análoga correspondiente al tiempo de evaluación.

Este bloque se utiliza para generar una señal digital correspondiente al tiempo de evaluación.

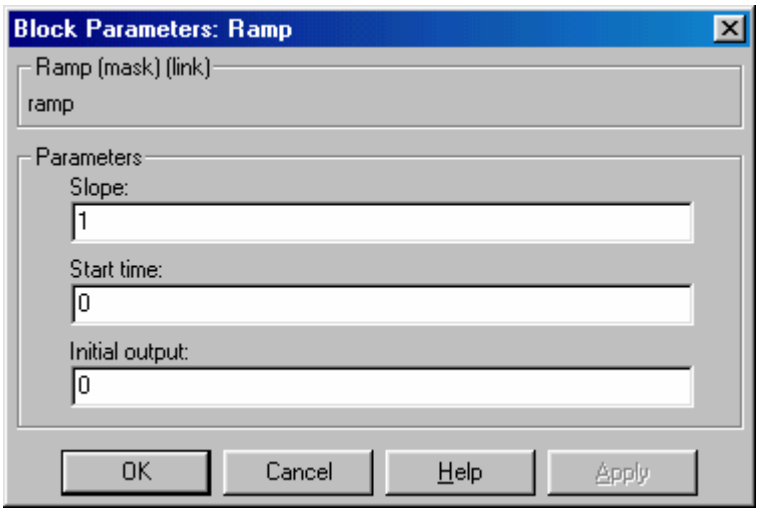
Step



Este bloque se utiliza para generar una señal de pulso.

Step time: Tiempo de retardo
Inicial value: Valor de inicio
Final value: Valor después del cambio
Sample time: Rate (evalúa la señal en un tiempo especificado si se le ingresa “0” la evaluación es automática)

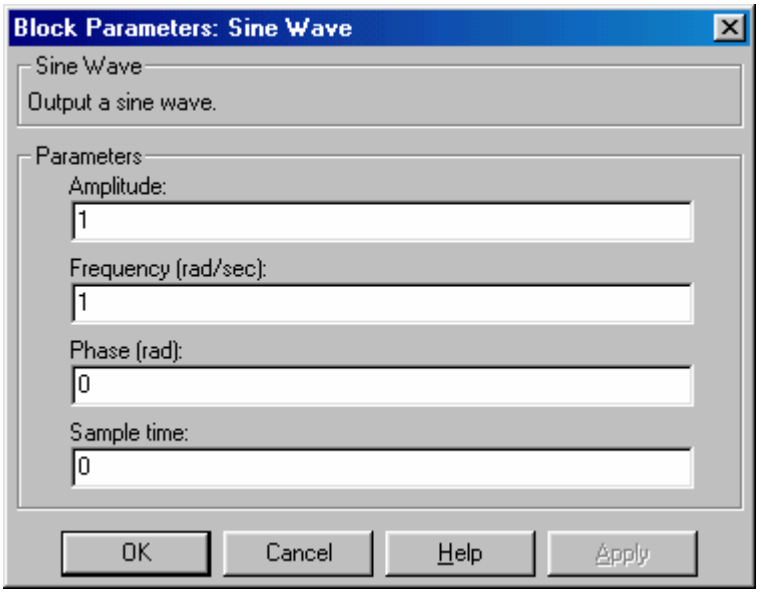
Ramp



Este bloque se utiliza para generar una señal del tipo y = mx + h.

Slope: m
Start time: cuando comienza a funcionar
Initial output: h

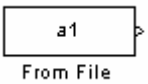
Sine Wave

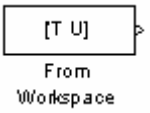


Este bloque se utiliza para generar una señal del tipo senoidal.

Amplitude: Amplitud de la señal.
Frequency (rad/sec): Frecuencia en radianes por segundo
Phase (rad): Desfasage en radianes
Sample time: Rate (evalúa la señal en un tiempo especificado si se le ingresa "0" la evaluación es automática)

From File and From Workspace

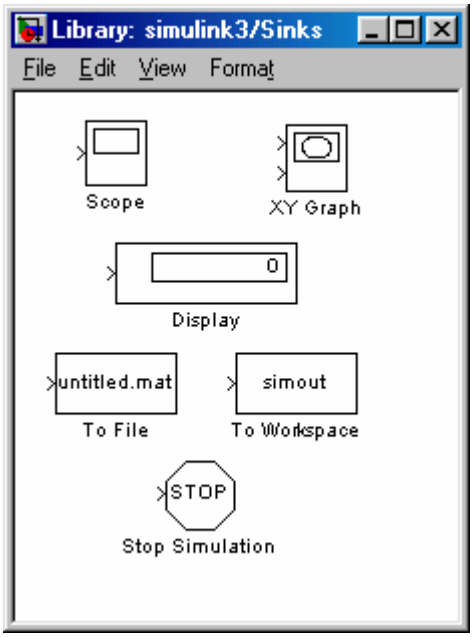




Este bloque utiliza una memoria mat para generar la señal

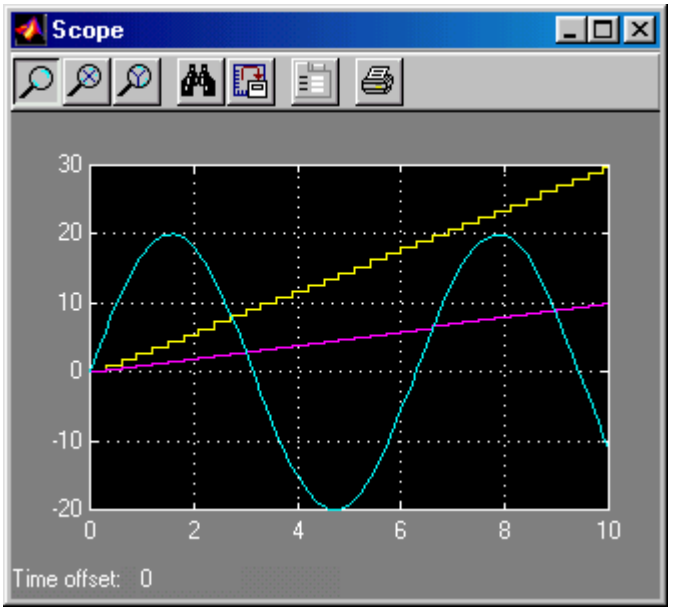
Este bloque utiliza una memoria común para generar la señal

La galería SINKS:



Los bloques que se encuentran dentro son los que registran las salidas graficando o guardando en memorias.

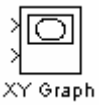
Scope

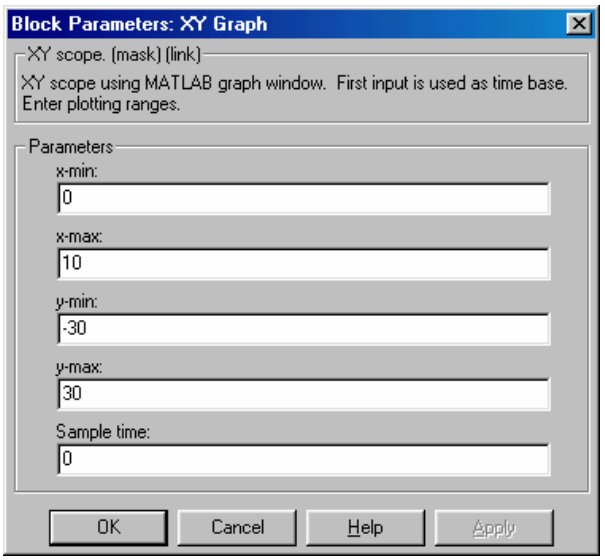


Este bloque grafica las señales

Las opciones que dispone el graficador son:
Zoom in: zoom más cerca.
zoom out: zoom más lejos.
zoom box: zoom sobre una región determinada.
zoom autoscale: zoom automático viendo todo el grafico.

XY Graph

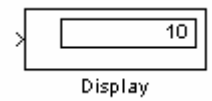


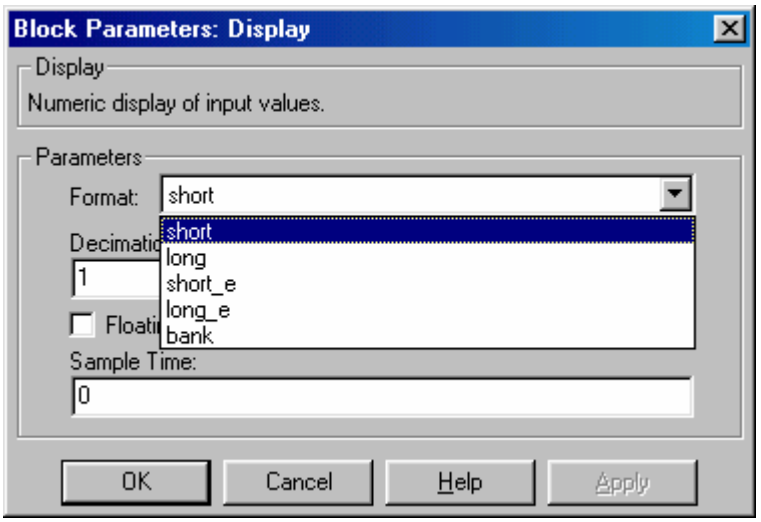


Este bloque grafica las señales según X e Y

Las opciones que dispone el graficador son:
x-min: El valor de comienzo según el eje x.
x-max: El valor de final según el eje x.
y-min: El valor de comienzo según el eje y.
y-max: El valor de final según el eje y.
Sample time: Rate (evalúa la señal en un tiempo especificado si se le ingresa “0” la evaluación es automática)

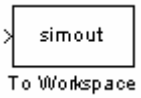
Display





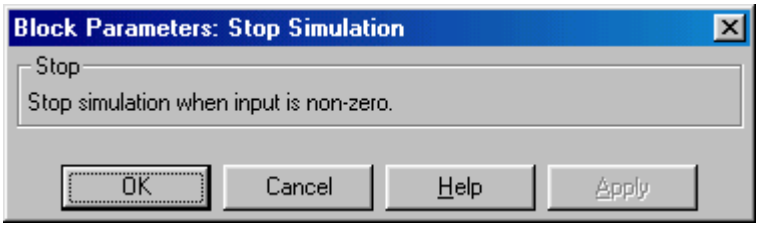
Este bloque muestra el valor de la señal punto a punto o hasta llegar a un valor cantante.	Fomat: El tipo de forma que se va a visualizar. Decimation: la precisión. Sample time: Rate (evalúa la señal en un tiempo especificado si se le ingresa “0” la evaluación es automática)

To File and To Workspace



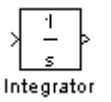
Este bloque utiliza una memoria mat para guardar la señal	Este bloque utiliza una memoria común para guardar la señal

Stop Simulation

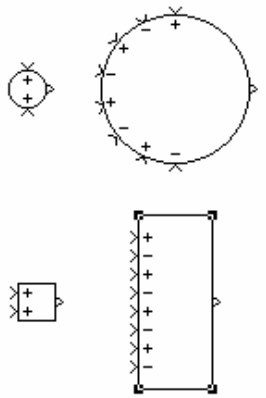
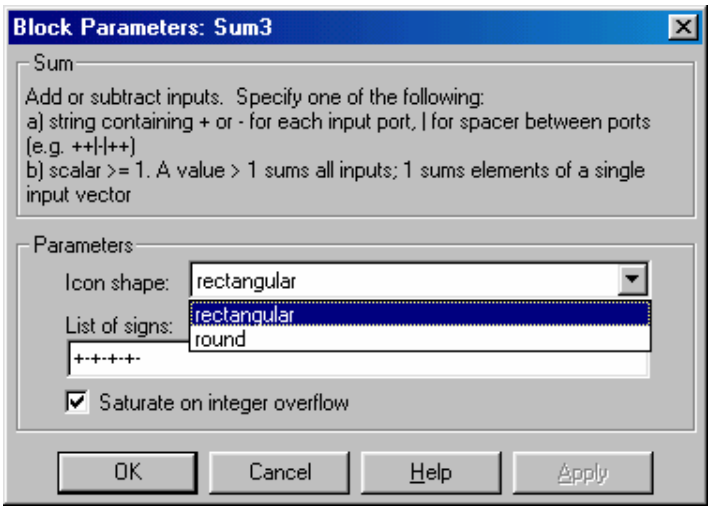


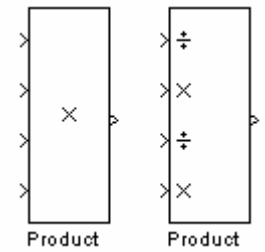
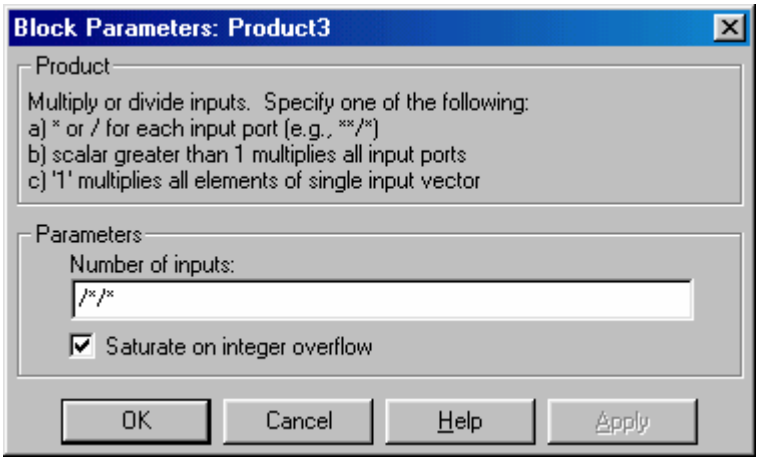
Este bloque detiene la simulación	Este bloque detiene la simulación cuando el valor que se le ingresa es distinto de “0” (cero).

La galería CONTINUOUS:

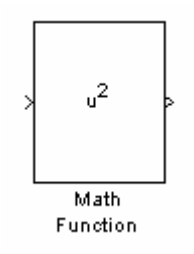
Integrator and Derivatrive	
	
Este bloque integra la señal punto a punto y muestra el acumulado hasta el momento	Este bloque deriva la señal punto a punto y muestra el acumulado hasta el momento

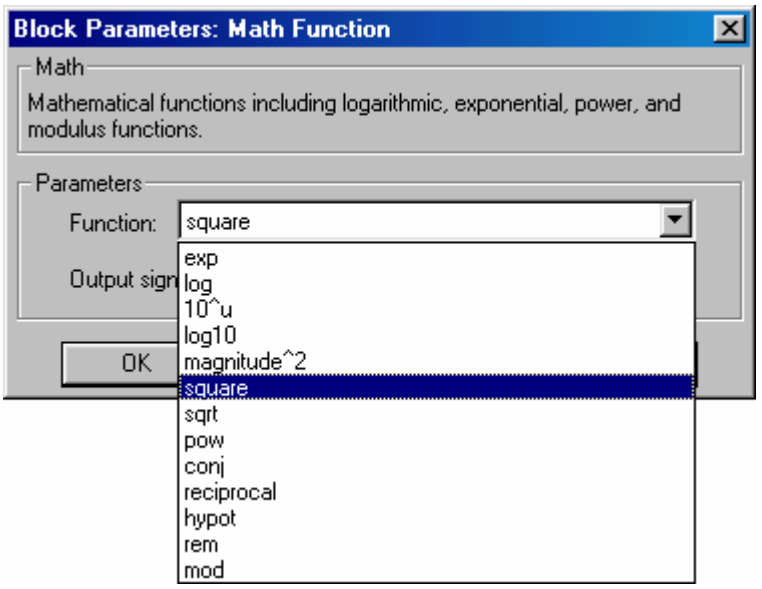
La galería MATH:

Sum	
	
Este bloque suma o resta las señales punto a punto. Se puede visualizar circular o rectangular.	Icon shape: La forma Rectangular o Circular. Listo of signs: Aquí se agrega + o – dependiendo lo que se necesite.

Product	
	
Este bloque multiplica o divide las señales.	Number of input: El numero de entradas, si se ingresan "/" la señas es dividida y si se coloca "*" se multiplica, y si se coloca un numero todas las entradas se multiplican.

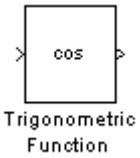
Math Function

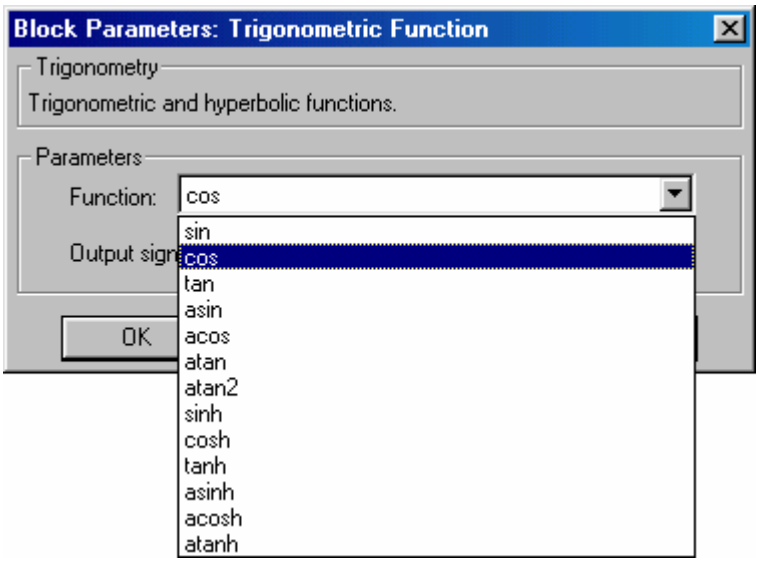




Este bloque aplica a la señal de entrada de varias operaciones matemáticas.	Function: Es la lista de las funciones disponibles.

Math Function





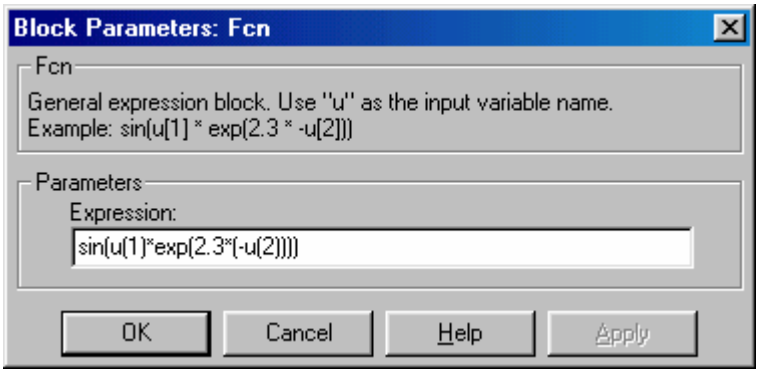
Este bloque aplica a la señal de entrada de varias operaciones trigonométricas matemáticas.	Function: Es la lista de las funciones trigonométricas disponibles.

Function

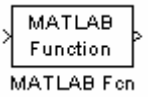
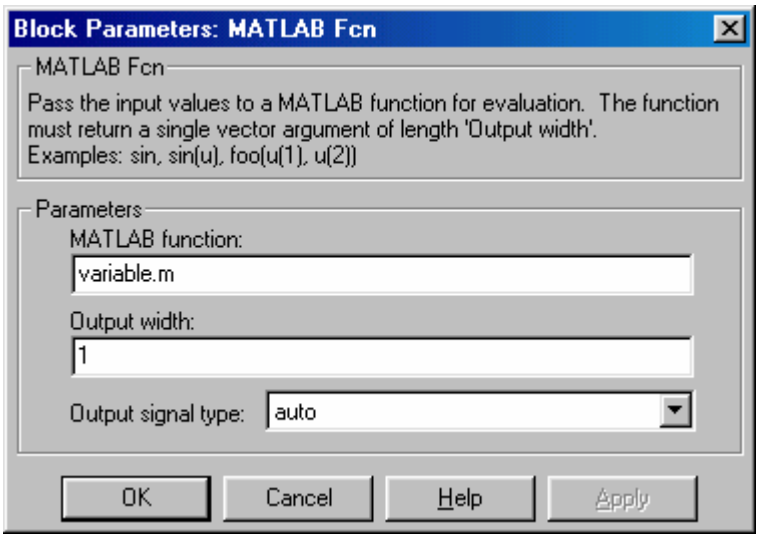
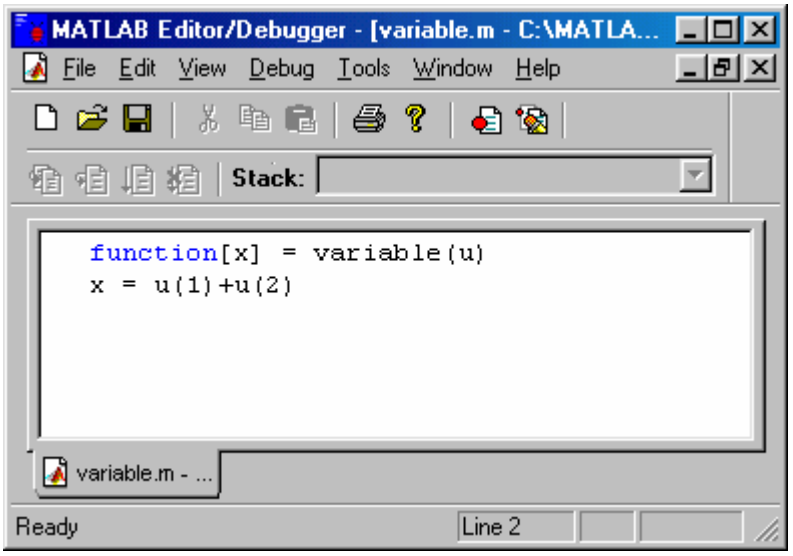
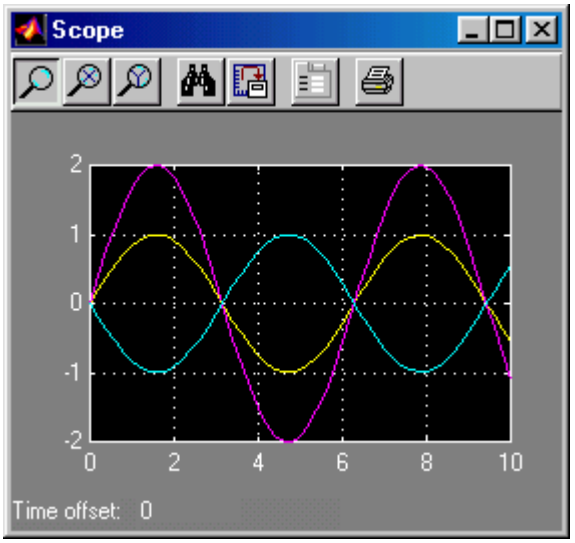
|u| Abs	Este bloque aplica a la señal de entrada el comando de valor absoluto.
Sign	Este bloque entrega el signo de la señal de entrada. Ej: +20 → +1 0 → 0 -20 → -1
AND Logical Operator	Este bloque actúa como un sistema digital interactuando con 0 y 1.
Relational Operator	Este bloque compara las señales digitales o análogas pero con un resultado digital.
Re Im Real-Imag to Complex	Este bloque une dos señales trasformándolas en complejas para luego poder resolver cualquier calculo de complejo sobre una señal única.
Re(u) Im(u) Complex to Real-Imag	Este bloque separa a la señal compleja en dos señales comunes.
|·| ∠ Magnitude-Angle to Complex	Este bloque convierte una señal polar en una rectangular.
|u| ∠u Complex to Magnitude-Angle	Este bloque convierte una señal rectangular en una polar.
f(z) Solve f(z) = 0 z Algebraic Constraint	Este bloque resuelve sistemas.

La galería FUNCTION AND TABLES:

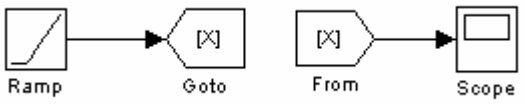
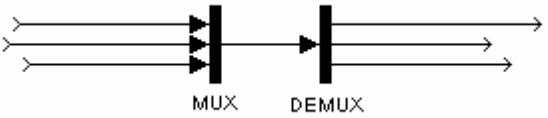
Fcn

f(u) Fcn	
Este bloque aplica a la señal de entrada de varias operaciones trigonométricas o matemáticas.	Function: Es la lista de las funciones trigonométricas disponibles.

MATLAB Fcn

Bloque de función	
En el parámetro MATLAB function: se coloca el nombre del archivo de extencion *.m generado en Matlab, Output width: es el valor de multiplicación final de la salida.	
Función que se genera en el editor de Matlab	
Resultado visualizado con el SCOPE	

La galería SIGNAL & SYSTEMS:

From & GOTO	
Estos bloques se utilizan para trasmitir una señal de un lado al otro a través estos bloques. Es muy útil cuando los sistemas se vuelven muy complejos y se necesita una señal determinada	
MUX & DEMUX	
Estos bloques sirven para unir señales y para separarlas.	
IN & OUT	
Estos bloques son utilizados en la conformación de subsistemas.