

## 4. Graficación en Matlab

Una gran parte de las matemáticas que se desarrollan en las ciencias y la ingeniería expresan relaciones entre variables, dominios y contradominios en dos o más dimensiones a través de ecuaciones escritas utilizando una simbología y notación un tanto complicada la mayoría de las veces y que no nos permite observar más que relaciones numéricas entre escalares, vectores o matrices.

Es aquí donde la graficación adquiere una gran importancia, pues el razonar y comprender muchos conceptos y comportamientos matemáticos está íntimamente ligado a poder visualizar los desarrollos, análisis y conclusiones de las hipótesis planteadas de una manera clara, mediante la utilización de entornos gráficos accesibles y fáciles de desarrollar por el usuario.

La graficación es una parte medular de Matlab y su existencia es una consecuencia lógica de la necesidad de complementar los análisis matemáticos con gráficas de alta calidad y sencillez en su elaboración. Todo esto se ha logrado debido, por un lado, al rapidísimo desarrollo que ha tenido el software y en especial los lenguajes de programación y por otro lado, las capacidades de cómputo y despliegue gráfico de las computadoras actuales que se han incrementado enormemente en nuestros días.

El tener la posibilidad de observar gráficamente las relaciones entre series de datos, funciones y ecuaciones, hace que el aprendizaje de las matemáticas sea más amigable, accesible y eficiente.

Cuando deseamos crear algún gráfico, debemos de recordar que el **entorno gráfico** se genera en una ventana aparte de la ventana de comandos. Es posible generar una o varias ventanas de entorno gráfico de forma simultánea.

- ❖ Para generar ventana(s) de entorno gráfico:

```
[70] >>figure ;
ó
    >>figure( handle ) ;
```

Al utilizar *figure* en su forma simple, generará una nueva ventana con su *handle* identificador, que no es más que un número entero. Dicho identificador será el número siguiente al último que exista; si no hay ninguna ventana gráfica, le asigna el No. 1.

La segunda opción del comando sirve para hacer referencia a una pantalla gráfica específica que ya existe. A partir del momento de referenciar a una pantalla, ésta se convierte en la **pantalla activa** y en la cuál se ejecutarán los comandos de graficación siguientes.

- ❖ Para cerrar ventanas de entorno gráfico:

```
[71] >>close ; ó >>close( handle ) ;
```

- ❖ Para borrar la pantalla de entorno gráfico activa

```
[72] >>clf ; ó >>clf reset ;
```

## 4.1 Graficación 2-D de funciones XY

La gráfica más común que utilizan científicos e ingenieros es la **Gráfica XY**. Básicamente este tipo de gráfica consiste en utilizar un conjunto de puntos de datos digamos  $(x_i, y_i)$  para  $i = 1, 2, 3, \dots, n$ ; en donde  $x$  e  $y$  son dos vectores de fila o columna de igual longitud.

Por lo general, se considera a  $x$  como el vector que contiene los valores de la variable independiente y a  $y$  como el vector de valores de la variable dependiente. Por lo tanto, los valores de  $y$  pueden obtenerse como una función de  $x$ :  $y_i = f(x_i)$ ; como es usual en matemáticas.

Sin embargo, también podría darse el caso de que los valores de ambos vectores fueran resultado de mediciones tabulares de algún experimento o generados por algún fenómeno físico.

Dichos datos podrían haber sido almacenados en algún medio magnético y posteriormente cargados para efectuar su lectura en variables de trabajo que permitan realizar su graficación y análisis.

### 4.1.1 Comandos básicos de graficación XY

❖ Para graficar vectores o matrices en XY

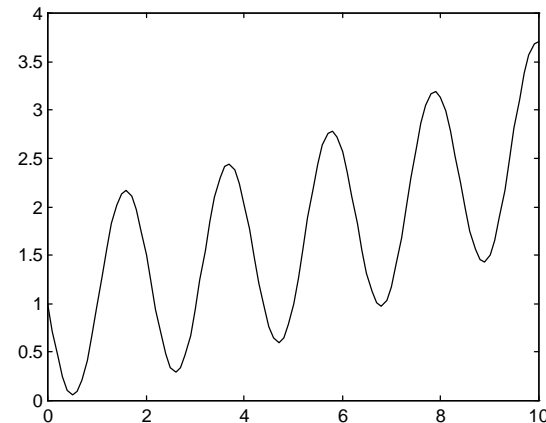
```
[73] >>plot ( x , y ) ;
ó
>>plot ( x , y , 'características' ) ;
```

Además, si  $x$  o  $y$  fuera una matriz y el otro un vector, se graficará el vector contra cada una de las columnas de la matriz. Si  $x$  e  $y$  son dos matrices, se graficará la relación de columna de  $x$  a correspondiente columna de  $y$ . En ambos casos, los tamaños de vectores y matrices tienen que ser compatibles para que no exista el error: “*Matrix dimensions must agree*”.

Por ejemplo, construyamos una gráfica XY:

```
>>% definimos el dominio para el vector x
>>% se calcula la función en cada punto
>>% graficamos

>>x = 0 : .1 : 10 ;
>>y = exp ( .1 .* x ) - sin ( 3 .* x ) ;
>>plot ( x , y )
```



Nótese que si no existe un entorno gráfico ya definido previamente, *plot* lo crea al momento.

La gráfica anterior se elaboró utilizando los parámetros por defecto de las 'características' que proporciona Matlab al usuario para adecuar la forma de ver las líneas en los gráficos XY.

Se pueden modificar tres parámetros de las líneas y son:

## 1. Tipos de marcas en las líneas:

Línea sin marcas en los puntos (default)

Línea con marcas en los puntos. Los tipos de marcas son:

- . (puntos)
- + (signos +)
- \* (asteriscos)
- o (círculos)
- x (cruces)

## 2. Tipos de estilos de líneas:

Los formatos de línea son:

- (continua. por defecto)
- (guiones)
- : (punteada)
- . (guiones y puntos)

## 3. Tipos de color de líneas:

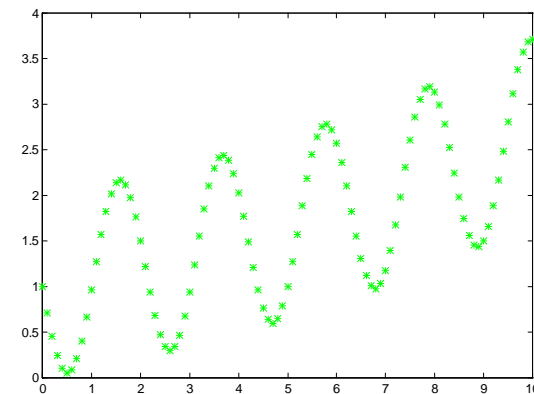
Los colores posibles son:

Amarillo (yellow)	'y'	Rojo (red)	'r'
Azul (blue)	'b'	Turquesa (cyan)	'c'
Blanco (white)	'w'	Verde (green)	'g'
Negro (black)	'k'	Violeta (Magenta)	'm'

Si modificamos un poco las instrucciones anteriores, digamos:

```
>>x = 0 : .1 : 10 ;
>>y = exp (.1 .* x) - sin ( 3 .* x ) ;
>>plot ( x, y , 'g*')
```

Obtenemos:



Como se puede apreciar en 'g\*', cambiamos las características para desplegar una línea de color verde y con un tipo de línea de asteriscos. El orden en que se especifiquen dichas características no es relevante.

A pesar de que la gráfica es lo suficientemente clara por sí misma, es recomendable que siempre se incluya un título principal, rótulos en los ejes, leyendas y texto en las gráficas.

Inclusive es aconsejable incluir las unidades en cada eje y cualquier otro comentario que sea pertinente.

Las instrucciones que se usan para incluir títulos, rótulos, leyendas y texto como anotaciones en una gráfica son:

- ❖ Para poner un título a la gráfica:

```
[74] >>title ( 'texto' );
ó
>>title ( [ 'texto' , 'prop1' , valor1 , ... ] );
```

El título de la gráfica se especifica en 'texto'. La(s) pareja(s) de 'prop<sub>n</sub>' y 'valor<sub>n</sub>' especifican los **atributos** que tendrá el título que pueden ser:

```
'fontname', 'nombre_de_fuente'
'fontsize', tamaño_de_fuente
'color', 'código_de_color'
```

El 'nombre\_de\_fuente' puede ser cualquier nombre de fuente válido de windows. Por ejemplo: 'arial' , 'courier' , 'garamond' , etc.

Si se usa 'symbol', será posible introducir **símbolos griegos** en las gráficas.

El *tamaño\_de\_fuente* es cualquier tamaño válido (en puntos) para la fuente seleccionada. Se especifica como un número entero, de acuerdo a los tamaños de fuentes en windows.

El 'código\_de\_color' usados para las fuentes, son los mismos que se especificaron al definir las 'características' para el comando *plot*.

- ❖ Para colocar rótulos en los ejes:

```
[75] >>xlabel ( 'texto' );
ó
>>xlabel ( [ 'texto' , 'prop1' , valor1 , ... ] );
```

El rótulo del *eje x* se especifica en 'texto'. La(s) pareja(s) de 'prop<sub>n</sub>' y 'valor<sub>n</sub>' especifican los **atributos** que tendrá el rótulo y que funcionan igual que en el comando *title*.

```
[76] >>ylabel ( 'texto' );
ó
>>ylabel ( [ 'texto' , 'prop1' , valor1 , ... ] );
```

El rótulo del *eje y* se especifica en 'texto'. La(s) pareja(s) de 'prop<sub>n</sub>' y 'valor<sub>n</sub>' especifican los **atributos** que tendrá el rótulo y que funcionan igual que en el comando *title*.

- ❖ Para añadir una leyenda al gráfico:

```
[77] >>legend ( 'texto' , pos );
ó
>>legend ( 'texto1' , 'texto2' , ... , pos );
ó
>>legend ( 'tipo_línea1' , 'texto1' , ... , pos );
```

El 'texto' indica la leyenda para la gráfica. El parámetro *pos* representa la posición de la leyenda en la pantalla: **0** → dentro del cuadro del gráfico (si hay lugar) y **-1** → fuera del cuadro del gráfico.

La segunda versión de este comando permite, si aparecen varias líneas en el gráfico, que se pueda mostrar una leyenda para cada una de ellas. En donde '*texto<sub>n</sub>*' indica una leyenda por línea en el mismo orden tal como se colocaron en el comando de graficación.

La última versión de este comando se usa para indicar manualmente en la leyenda diferentes tipos de línea con su correspondiente descripción. El '*tipo\_linea*' corresponde a los especificados anteriormente en las '*características*' del comando *plot*.

Si se desea mover la leyenda al estar visualizando la pantalla gráfica, se puede oprimir el botón izquierdo del ratón y arrastrarla hasta la posición deseada.

❖ Para desplegar texto en una gráfica:

```
[78] >>text ( x , y , 'texto' ) ;
ó
    >>text ( x , y , 'texto' , 'prop1' , valor1 , ... ) ;
```

Los valores *x* e *y* son las coordenadas del punto en donde se desea que comience a desplegarse la cadena '*texto*' a partir de su esquina inferior izquierda.

Dicha cadena puede ser una variable alfanumérica definida previamente o un texto encerrado entre apóstrofes.

La(s) pareja(s) de '*prop<sub>n</sub>*' y '*valor<sub>n</sub>*' , en la segunda versión, especifican los **atributos** que tendrá el texto y que funcionan igual que en el comando *title*.

Otra variante al comando *text* que es:

```
[79] >>gtext ( 'texto' ) ;
```

En este caso, *gtext* permite que se pueda determinar manualmente la ubicación del texto directamente en la pantalla gráfica. Esto puede resultar muy práctico.

Al dar el comando, aparece en la pantalla gráfica un cursor en forma de cruz el cuál podemos desplazar hasta una cierta ubicación. Al dar click con el botón izquierdo del mouse o también oprimir cualquier tecla, el '*texto*' se posicionará en dicho lugar, iniciando en su esquina inferior izquierda.

❖ Conversión de números a cadenas alfanuméricas

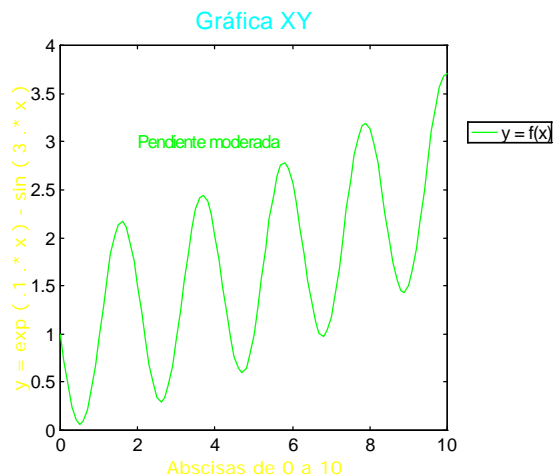
```
[80] >>num2str ( x ) ;
y
[81] >>int2str ( x ) ;
```

Estos comandos permiten convertir números a cadenas de letras. El primero convierte al número real *x* a una cadena con aproximadamente cuatro dígitos y un exponente (si fuese necesario); la segunda convierte al número entero *x* en la cadena alfanumérica correspondiente.

Estos comandos son muy útiles pues se pueden combinar con *title*, *xlabel*, *ylabel*, *legend*, *text* y *gtext*, para desplegar valores de variables, convertidos en cadenas, junto con los textos en la pantalla gráfica.

Finalmente, ya estamos en condiciones de dar un mejor aspecto a nuestra gráfica original del ejemplo anterior:

```
>>%graficamos
>>x = 0 : .1 : 10 ;
>>y = exp ( .1 .* x ) - sin ( 3 .* x ) ;
>>plot ( x, y , 'g-' )
>>%ponemos título y rótulos
>>title ( 'Gráfica XY' , 'fontname' , 'arial' , ...
'fontsize' , 16 , 'color' , 'c' )
>>xlabel ( 'Abscisas de 0 a 10' , 'color' , 'y' , ...
'fontsize' , 12 , 'fontname' , 'verdana' )
>>ylabel ( 'y = exp ( .1 .* x ) - sin ( 3 .* x )' , ...
'color' , 'y' , 'fontsize' , 12 , 'fontname' , ...
'verdana' )
>>%ponemos leyenda fuera de la gráfica
>>legend ( 'y = f(x)' , -1)
>>%colocamos el texto en las coord. (2,3)
>>text ( 2, 3, 'Pendiente moderada' , 'color' , 'g' , ...
'fontsize' , 12 , 'fontname' , 'comic sans' )
```



#### 4.1.2 Comandos adicionales para gráficos XY

❖ Para mantener activa la pantalla gráfica actual

[82] >>**hold on** ; ó >>**hold off** ;

Permite que a partir de ejecutarse el *hold on*, los nuevos comandos gráficos que se tecleen, se hagan sobre la misma pantalla gráfica que está activa (un efecto de **add** al entorno gráfico).

Al dar el *hold off*, los nuevos comandos gráficos que se tecleen generarán una nueva pantalla de entorno gráfico (un efecto de **replace** al entorno gráfico).

*Hold* por sí solo, invierte su estado actual; es decir, si estaba puesta la opción de mantener el entorno gráfico apaga dicha opción y viceversa.

❖ Para visualizar una retícula guía en la gráfica

[83] >>**grid on** ; ó >>**grid off** ;

Al dar *grid on* se despliega una retícula ó rejilla guía para ambos ejes en la gráfica. Si tecleamos *grid off*, dicha rejilla desaparece.

El teclear *grid* por sí solo, hace que la rejilla invierta su estado actual, apagándola ó prendiéndola según sea el caso. El tamaño de la rejilla guía dependerá de la definición de los valores para los ejes x e y.

❖ Para manipular los ejes coordenados XY

```
[84] >>axis ('on' );    ó    >>axis ('off' );
ó
    >>axis ('square' );
    >>axis ('normal' );
ó
    >>axis ( [ x_min, x_max, y_min, y_max ] );
```

Las opciones `axis('on')` y `axis('off')` de este comando, permiten poner o quitar el despliegue de los ejes coordenados en la gráfica.

La opción `axis('square')` despliega los ejes coordenados de la gráfica en forma de cuadrado, independientemente del rango de valores de los ejes.

Para restaurar cualquier cambio que se hubiese hecho a los ejes utilizamos la opción `axis('normal')`. Esto regresa a la normalidad tanto el rango de los valores máximos y mínimos para  $x$  e  $y$ , y también restaura la forma original de la gráfica, cancelando cualquier despliegue en forma cuadrada.

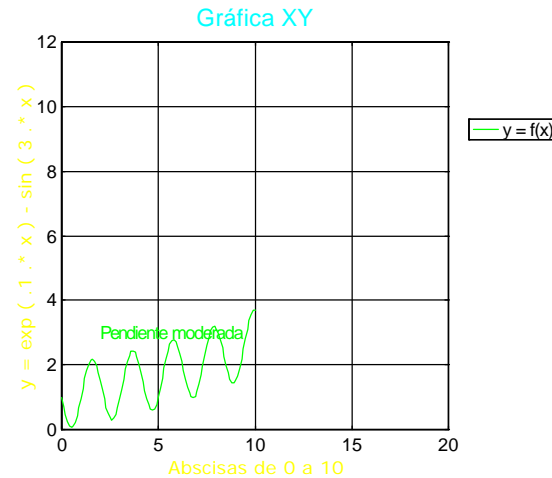
La última es la más utilizada de todas las opciones anteriores ya que permite establecer el rango de valores mínimos y máximos para cada uno de los ejes por separado.

Esta última opción permitirá examinar regiones particulares de gráficas que sean de interés para el usuario. Habrá que tener en cuenta que las líneas que salgan de una región particular serán recortadas.

Con el fin de observar su aplicación, se incluirán las siguientes instrucciones al final del listado anterior:

```
>>hold on ;
>>grid on ;
>>axis ( [0, 20, 0, 12 ] );
```

La gráfica resultante es:



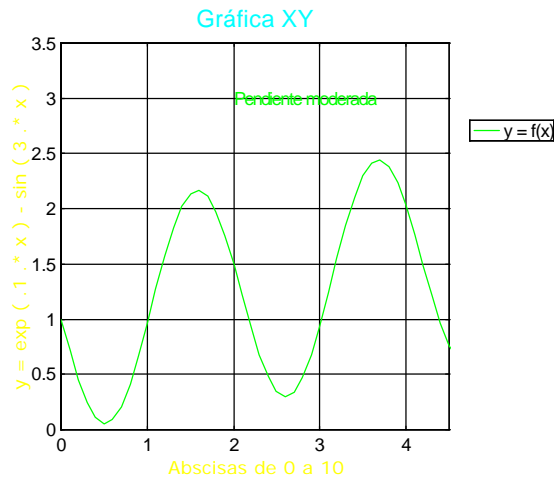
Observe que como el rango de los ejes se “agrandó” la gráfica quedó con su rango original y no llena totalmente el plano XY, pues no se ha hecho un nuevo llamado a la instrucción `plot` cambiando el rango para la variable independiente que es  $x$ , ahora de 0 a 20.

Sin embargo, si el cambio en la escala de los ejes se hace “achicando” sus valores límites, entonces, sí será posible observar con mas detalle una región pequeña de la gráfica de la función.

Una vez que se aplicó el comando:

```
>>axis ( [ 0, 4.5, 0, 3.5 ] );
```

La gráfica resultante es:



Que muestra el efecto de “acercamiento” que se esperaba.

Si lo que se desea es efectuar acercamientos más eficientes en el entorno gráfico existe otra instrucción diseñada especialmente para este fin y es:

❖ Comando de **zoom** en el entorno gráfico

```
[85] >>zoom on ; ó >>zoom off ;  
ó  
>>zoom out ;
```

Cuando en la pantalla de comandos se tecldea la orden **zoom on**, se muestra automáticamente la pantalla del entorno gráfico que esté activa en dicho momento.

Con el botón izquierdo del ratón (sosteniéndolo) podemos marcar una zona de la gráfica que nos interese y al soltar se hará un efecto de **zoom in** (acercamiento), aproximadamente para la zona seleccionada.

También podemos solamente dar *click* con el botón izquierdo del ratón sobre un punto de la pantalla gráfica y entonces el acercamiento automático será más o menos del doble para cada vez que se oprima el botón. El punto señalado pasará a ser aproximadamente el centro de la nueva gráfica. El botón derecho del ratón, tendrá efecto de **zoom out** (alejamiento) pero su funcionamiento es similar que en los casos mencionados.

Regresando a la pantalla de comandos podemos teclear la orden **zoom out**, que restaurará por completo la gráfica a su vista completa; justamente antes de que se comenzara a experimentar con el zoom.

**Nota:** Para evitar la posibilidad de comportamientos extraños de la computadora por el manejo directo del hardware de la interfaz de despliegue gráfico que realiza Matlab, se recomienda que después de efectuar cualquier operación que involucre el uso del entorno gráfico se dé un *clear all* para limpiar todas las variables y funciones, un *clf reset* para inicializar completamente el entorno gráfico ó mejor aún, salir por completo del Matlab y volver a entrar si esto fuera necesario.

### 4.1.3 Desplegando múltiples gráficas

Matlab brinda la oportunidad de poder graficar varias curvas en forma simultánea en una misma pantalla de entorno gráfico. Básicamente son **tres** las opciones a las que podemos recurrir para lograr esto:

#### ❖ Múltiples gráficas usando un vector columna

Se puede crear un vector columna que contenga en cada renglón a cada una de las funciones que se desean graficar. Por ejemplo:

```
>>%creamos el rango de abscisas
>>x = -2*pi : 0.1 : 2*pi ;

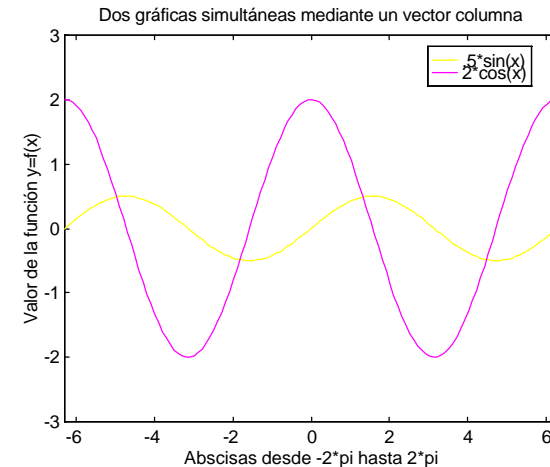
>>%se generan los puntos de ambas f(x)
>>f1 = .5 * sin ( x ) ;
>>f2 = 2 * cos ( x ) ;

>>%se crea el vector columna
>>V = [ f1 ; f2 ] ;

>>%se grafica la relación de x con el vector V
>>plot ( x , V ) ;
>>%se ajustan los ejes al tamaño deseado
>>axis ( [ -2*pi , 2*pi , -3 , 3 ] ) ;

%colocamos título, rótulos y leyenda
>>title ( 'Dos gráficas simultáneas mediante un
vector columna ' ) ;
>>xlabel ( 'Abscisas desde -2*pi hasta 2*pi' ) ;
>>ylabel ( 'Valor de la función y=f(x)' ) ;
>>legend ( '.5*sin(x)' , '2*cos(x)' ) ;
```

El listado anterior produce el resultado:



El único inconveniente de utilizar este método es que no existe la posibilidad de escoger ni tipo de línea ni color para cada una de las curvas; los que se despliegan son los valores por default (línea continua y colores en el orden preestablecido).

#### ❖ Múltiples gráficas usando una variante de *plot*

Para poder representar cada curva con un tipo de línea distinto y un color diferente usamos una versión modificada del comando *plot*.

```
>>plot ( x , f1 , 'y—' , x , f2 , 'm+' ) ;
```

Las instrucciones completas serían:

Por ejemplo,

```
>>x = -2*pi : 0.1 : 2*pi ;

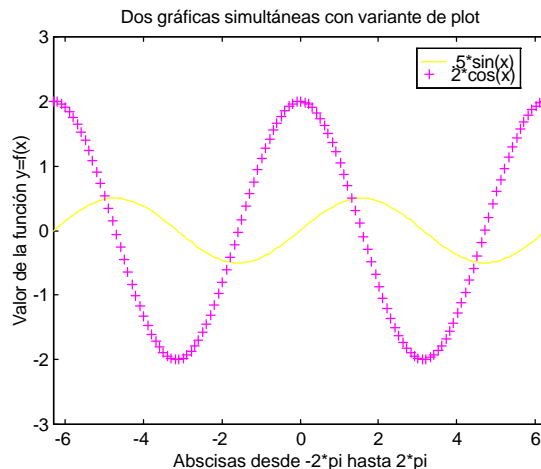
>>f1 = .5 * sin ( x ) ;
>>f2 = 2 * cos ( x ) ;

>>plot ( x , f1 , 'y—' , x , f2 , 'm+' ) ;

>>axis ( [ -2*pi , 2*pi , -3 , 3 ] ) ;

>>title ( 'Dos gráficas simultáneas con variante de
plot ' ) ;
>>xlabel ( 'Abscisas desde -2*pi hasta 2*pi' ) ;
>>ylabel ( 'Valor de la función y=f(x)' ) ;
>>legend ( '.5*sin(x)' , '2*cos(x)' ) ;
```

Produce la gráfica:



Una ventaja adicional de utilizar la variante de *plot* es que se podrían graficar más de dos curvas simultáneamente.

### ❖ Múltiples gráficas mediante el uso de **subgráficas**

La pantalla total del entorno gráfico puede ser dividida en pequeñas regiones, como si se tratara de una matriz, de tal forma que es posible graficar una *subgráfica* independiente en cada una de dichas áreas.

El comando *subplot* es el encargado de que podamos graficar  $m \times n$  subgráficas en una sola figura:

```
[86] >>subplot ( m , n , k ) ;
```

Donde la pareja de  $m$  y  $n$  son dos números enteros que indican el tamaño  $m \times n$  de la matriz en que se dividirá el entorno gráfico.  $k$  también es un entero que indica el número secuencial (por renglón) de subpantalla que se va a utilizar como pantalla activa.

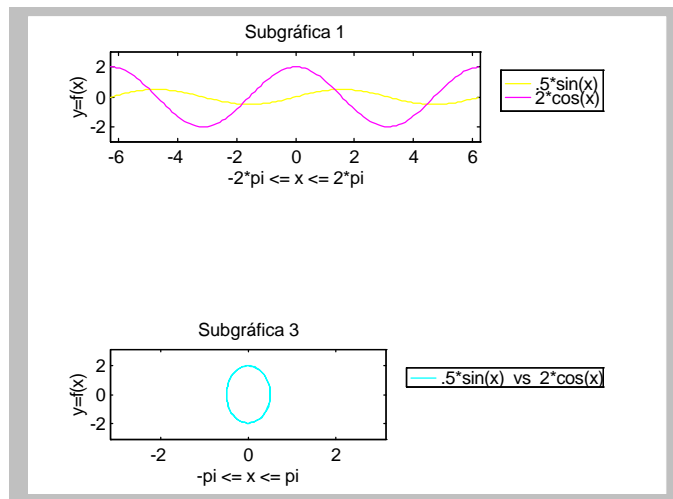
Por lo general el comando *subplot* se utiliza justamente en la línea anterior a un *plot* para referenciar y activar la subpantalla correspondiente al lugar en donde se desea graficar la curva.

Un aspecto muy versátil es que cada una de las subgráficas es tratada como una gráfica completamente independiente y, entonces, será posible establecer diferentes características de títulos, rótulos, leyendas, colores, tipos de letra, rangos de valores para los ejes y retícula en cada una de ellas.

Por ejemplo,

```
>>x = -2*pi : 0.1 : 2*pi ;
>>f1 = .5 * sin ( x ) ;
>>f2 = 2 * cos ( x ) ;
>>subplot ( 3 , 1 , 1 ) ;
>>plot ( x , f1 , 'y—' , x , f2 , 'm+' ) ;
>>axis ( [ -2*pi , 2*pi , -3 , 3 ] ) ;
>>title ( 'Subgráfica 1' ) ;
>>xlabel ( '-2*pi <= x <= 2*pi' ) ;
>>ylabel ( 'y=f(x)' ) ;
>>legend ( '.5*sin(x)' , '2*cos(x)' ) ;
>>subplot ( 3 , 1 , 3 ) ;
>>plot ( f1 , f2 , 'c' ) ;
>>axis ( [ -pi , pi , -pi , pi ] ) ;
>>title ( 'Subgráfica 3' ) ;
>>xlabel ( 'pi <= x <= pi' ) ;
>>ylabel ( 'y=f(x)' ) ;
>>legend ( '.5*sin(x) vs 2*cos(x)' ) ;
```

Produce la figura:



Observe que en el ejemplo anterior se utilizó una matriz de 3 x 1 subgráficas, de las cuáles sólo la uno y la tres se desplegaron. La dos, que va en medio, quedó vacía.

Otro ejemplo:

```
>>x = linspace(-2*pi , 2*pi ) ;
>>f1 = sin ( x ) ;
>>f2 = cos ( x ) ;
>>f3 = sec ( x ) ;
>>f4 = csc ( x ) ;

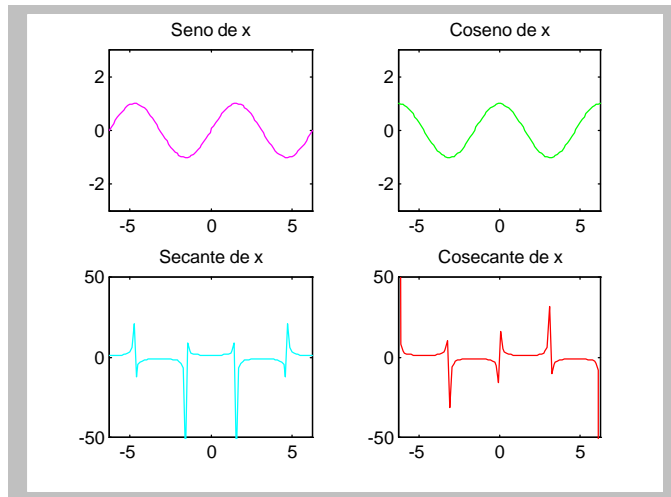
>>subplot ( 2 , 2 , 1 ) ;
>>plot ( x , f1 , 'm' ) ;
>>axis ( [ -2*pi , 2*pi , -3 , 3 ] ) ;
>>title ( 'Seno de x' ) ;

>>subplot ( 2 , 2 , 2 ) ;
>>plot ( x , f2 , 'g' ) ;
>>axis ( [ -2*pi , 2*pi , -3 , 3 ] ) ;
>>title ( 'Coseno de x' ) ;

>>subplot ( 2 , 2 , 3 ) ;
>>plot ( x , f3 , 'c' ) ;
>>axis ( [ -2*pi , 2*pi , -50 , 50 ] ) ;
>>title ( 'Secante de x' ) ;

>>subplot ( 2 , 2 , 4 ) ;
>>plot ( x , f4 , 'r' ) ;
>>axis ( [ -2*pi , 2*pi , -50 , 50 ] ) ;
>>title ( 'Cosecante de x' ) ;
```

La figura resultante es:



#### 4.1.4 Otras opciones de graficado

Adicionalmente a la instrucción *plot*, Matlab ofrece otras alternativas de comandos para la obtención de gráficos:

##### ❖ Gráficas *semilogarítmicas* y *logarítmicas*

Las funciones disponibles son:

[87] `>>semilogx ( x , y ) ;`

[88] `>>semilogy ( x , y ) ;`

[89] `>>loglog ( x , y ) ;`

Por ejemplo, tomemos la función:

$$y = e^{0.25 x \cdot \sin(x)}$$

Y grafiquemos las diferentes opciones que nos dan los comandos anteriores para un dominio de  $x$  de 0 a 10:

```
>>x = linspace(0 , 10 ) ;
```

```
>>y = exp ( 0.25 * x .* sin ( x ) ) ;
```

```
>>subplot ( 2 , 2 , 1 ) ;
```

```
>>plot ( x , y , 'y' ) ;
```

```
>>title ( 'exp ( 0.25 * x .* sin ( x ) ) con plot' ) ;
```

```
>>subplot ( 2 , 2 , 2 ) ;
```

```
>>semilogx ( x , y , 'r' ) ;
```

```
>>title ( 'con semilogx' ) ;
```

```
>>subplot ( 2 , 2 , 3 ) ;
```

```
>>semilogy ( x , y , 'g' ) ;
```

```
>>title ( 'con semilogy' ) ;
```

```
>>subplot ( 2 , 2 , 4 ) ;
```

```
>>loglog ( x , y , 'm' ) ;
```

```
>>title ( 'con loglog' ) ;
```

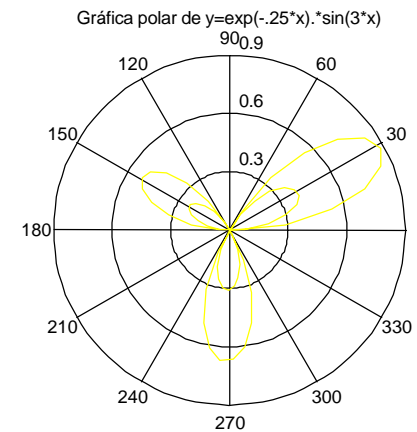
Como se podrá observar podemos tener una escala de tipo logarítmico ya sea en el eje  $x$ , en el eje  $y$  o en ambos.

Los resultados del ejemplo anterior se muestran en la siguiente gráfica:

Las instrucciones son:

```
>>x = 0 : .1 : 2*pi ;
>>y = exp ( -.25 * x ) .* sin ( 3 * x ) ;
>>polar ( x , y ) ;
>>title ( 'Gráfica polar de y = exp ( -.25 * x ) .*
sin ( 3 * x ) ' ) ;
```

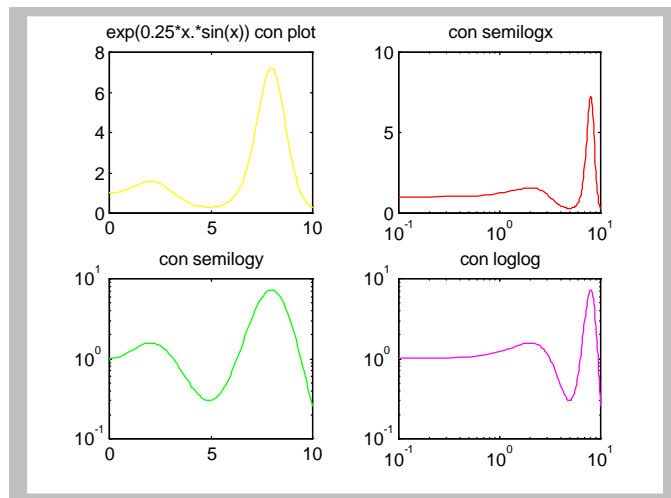
Cuyo resultado es :



Otro ejemplo,

Si,  $y = .25 * \sin ( 3 * x ) .* \cos ( 2.5 * x ) + \pi / 4$

```
>>x = 0 : .1 : 2*pi ;
>>y = .25 * sin ( 3 * x ) .* cos ( 2.5 * x ) + pi / 4 ;
>>polar ( x , y , 'r' ) ;
>>title ( 'Gráfica polar de y = .25 * sin ( 3 * x ) .*
cos ( 2.5 * x ) + pi / 4 ' ) ;
```



## ❖ Gráficas polares

Es posible graficar una función en coordenadas polares con el comando:

```
[90] >>polar ( x , y ) ;
```

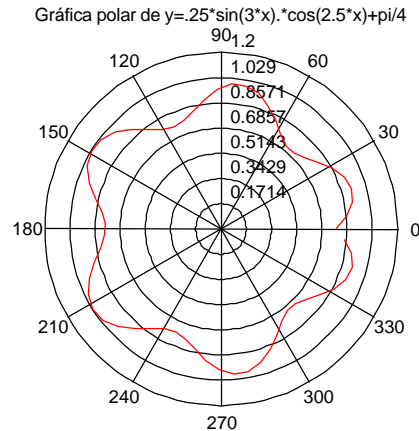
Por ejemplo:

Sea,

$$y = e^{-.25 x} \sin ( 3 * x )$$

Grafiquemos la función para un dominio de x de 0 a 2\*pi radianes.

Nos despliega:



❖ Gráficas de tipo estadístico

Matlab también cuenta con instrucciones que permiten visualizar los datos con un formato de tipo estadístico como lo son las **gráficas de barras** y los **histogramas**. Esto se logra con los comandos:

a) Para **gráficas de barras**

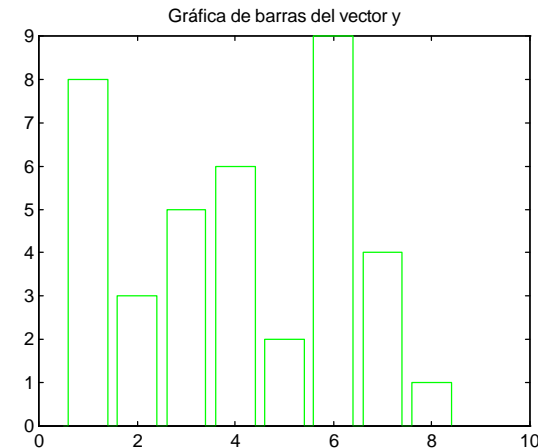
```
[91] >>bar ( y );
ó
    >>bar ( x , y );
ó
    >>[ xx , yy ] = bar ( x , y );
```

En su forma básica, el comando *bar* despliega una gráfica de barras de los valores contenidos en el vector *y*.

Por ejemplo: Si,  $y = [ 8 \ 3 \ 5 \ 6 \ 2 \ 9 \ 4 \ 1 ]$ , entonces:

```
>> y = [ 8 3 5 6 2 9 4 1 ];
>>bar ( y , 'g' );
>>title ( 'Gráfica de barras del vector y' );
```

Desplegará:



La segunda opción de este comando despliega una gráfica de barras de los valores del vector *y* pero situando las barras en las abscisas de acuerdo a los valores del vector *x*, que necesariamente deberá ser del mismo tamaño que *y*.

Por ejemplo:

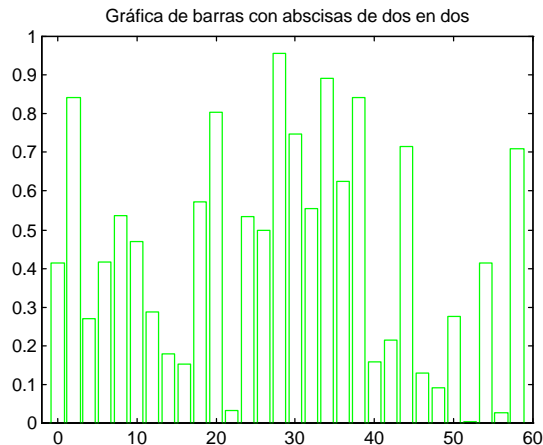
Supongamos que se quiere generar muestra de 30 valores aleatorios de distribución uniforme y se desea desplegarlos en una gráfica de barras con abscisas que se incrementen de dos en dos a partir de cero:

```
%generamos los valores aleatorios
>>y = rand ( 1 , 30 ) ;

%generamos las abscisas correspondientes
>>x = 0 : 2 : 58 ;

%desplegamos la gráfica
>>bar ( x , y , 'g' ) ;
>>axis ( [ -2 , 60 , 0 , 1 ] ) ;
>>title ( 'Gráfica de barras con abscisas de dos
          en dos' ) ;
```

Para obtener:



La última opción del comando *bar*, no despliega ninguna gráfica pero sirve para obtener los vectores *xx* e *yy* que pueden ser graficados con una instrucción del tipo `plot ( xx , yy )`, produciendo también una gráfica de barras en vez de obtener una línea quebrada.

b) Para gráficas de **histogramas**

```
[92] >>hist ( y ) ;
ó
    >>hist ( y , n ) ;
ó
    >>[ yy , xx ] = hist ( y ) ;
ó
    >>[ yy , xx ] = hist ( y , n ) ;
```

En su forma básica, el comando *hist* despliega una gráfica de barras de los valores contenidos en el vector *y* pero en forma de histograma.

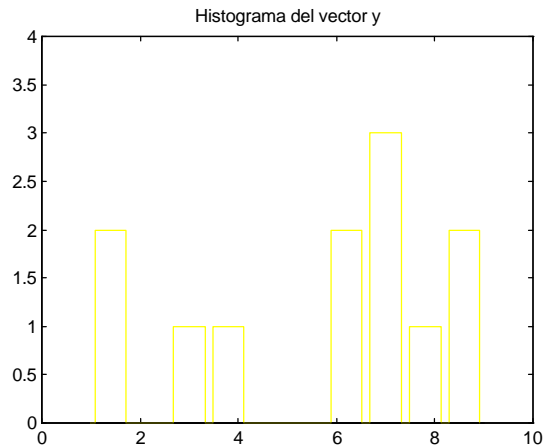
Recordemos que un histograma es muy similar a una gráfica de barras en la cuál se han agrupado los valores por **rangos de clase**, es decir de acuerdo a la frecuencia de aparición de valores repetidos.

Por ejemplo: Si,  $y = [ 1 1 4 6 6 3 7 7 7 8 9 9 ]$

Entonces,

```
>> y = [ 1 1 4 6 6 3 7 7 7 8 9 9 ] ;
>>hist ( y ) ;
>> title ( 'Histograma del vector y' ) ;
```

Produce la gráfica:



La segunda opción de este comando despliega un histograma de frecuencias de los valores del vector  $y$  pero mostrando los valores agrupados en solamente el número de *rangos de clase* que se hayan especificado en el escalar entero  $n$ .

Por ejemplo:

Supongamos que se tiene la siguiente información para el vector  $y$  :

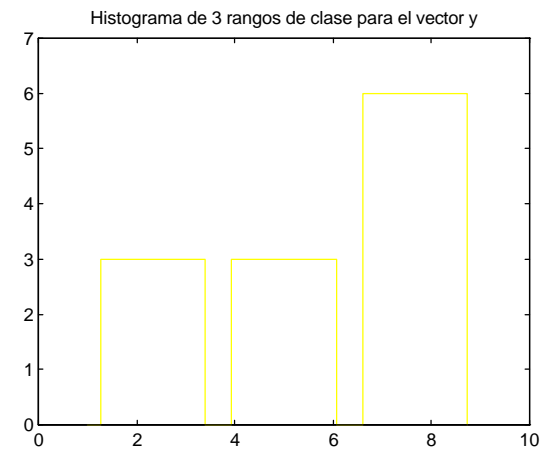
$$y = [1 \ 1 \ 4 \ 6 \ 6 \ 3 \ 7 \ 7 \ 7 \ 8 \ 9 \ 9]$$

y se desea graficar únicamente tres rangos de clase.

Las instrucciones,

```
>>y = [ 1 1 4 6 6 3 7 7 7 8 9 9 ];
>>n = 3 ;
>>hist ( y , n ) ;
>>axis ( [ 0 10 0 7 ] ) ;
>>title ( 'Histograma de 3 rangos de clase para el
vector y' ) ;
```

Presenta en pantalla:



Las dos últimas opciones del comando *hist*, no despliegan ninguna gráfica pero sirven para obtener los vectores  $xx$  e  $yy$  que pueden ser graficados con una instrucción del tipo `bar ( xx , yy )`, desplegando la gráfica de barras correspondiente al histograma.