Práctica 2
Introducción a Matlab II

Objetivo. El objetivo de esta práctica es continuar con las herramientas más usuales de Matlab: el manejo de polinomios, el cálculo de raíces y las instrucciones de graficación en 2D y en 3D.

Introducción.

Manejo de Polinomios en Matlab.

Además de las matrices, otro de los objetos típicos en Matlab son los polinomios. Un polinomio en Matlab se representa por un vector fila conteniendo los coeficientes del dicho polinomio. En general, un polinomio de grado n arreglado en potencias descendentes de la variable x

$$P(x) = a_n x^n + a_{n-1} x^{n-1} + ... + a_1 x + a_0$$

Se representará en Matlab como el vector de n+1 componentes:

$$P = \begin{bmatrix} a_n & a_{n-1} & ... & a_1 & a_0 \end{bmatrix}$$

Matlab proporciona varias funciones que permiten trabajar con polinomios, en la siguiente tabla se describen brevemente las principales.

función	descripción
`roots(P)`	Obtiene las raíces del polinomio P
`poly(r)`	Obtiene el polinomio correspondiente a las raíces dadas en el vector r
`polyval(P,x)`	Evalúa el polinomio P en el valor x
`conv(A,B)`	Multiplica los polinomios A, B
`[Q,R]=deconv(B,A)`	Realiza la división de polinomios B/A calculando cociente Q y residuo R
`[R,P,K]=residue(B,A)`	Calcula la expansión en fracciones parciales de la división B/A

Ejemplo. Para los polinomios $A(x) = x^4 + x^3 + x^2 + x + 1$, $B(x) = (x-1)^2(x-2)(x-3)$. a) Calcular sus raíces. b) Escribir ambos en potencias descendentes. c) Verificar algunas de las raíces calculadas. d) Obtener el polinomio $C(x) = A(x)B(x)$. e) Obtener el cociente y el residuo de la división $A(x) / B(x)$. f) Obtener la expansión e fracciones simples de la división $A(x) / B(x)$.

Solución:

```
>> A=[1 1 1 1 1 ];       %%%Define el polinomio A
>> B=poly([1 1 2 3])     %%%Define el polinomio B a partir de sus raíces
B =
     1    -7    17   -17     6
%%% Es decir, B(x) = x^4 - 7x^3 + 17x^2 - 17x + 6

>> r=roots(A)                  %%%Calcula las cuatro raíces de A
```

```
r =
   0.3090 + 0.9511i
   0.3090 - 0.9511i
  -0.8090 + 0.5878i
  -0.8090 - 0.5878i

>> polyval(A,0.3090+0.9511i)  %%%Verifica que 0.3090+0.9511i es raíz de A
ans =
  1.1606e-005 -1.9824e-004i
%%% Obsérvese que no da cero porque se usaron solo 4 dígitos
%%% Sin embargo, el resultado es cercano a cero.
%%% Para obtener un resultado más cercano a cero usamos más dígitos:
%%% Se pueden usar todos los dígitos sin escribirlos, puesto
%%% que Matlab los almacena todos, pero solo despliega los que requiere
%%% el formato actual:
>> polyval(A,r(1))%%%Verifica la primer raíz calculada
ans =
  9.9920e-016 +3.3862e-015i
%%% Obsérvese que el resultado es casi cero.

>> polyval(B,1)   %%% Ahora verifica que 1 es una raíz de B
ans =
     0
%%% Obsérvese que en este caso el resultado es exacto,
%%% porque la raíz se conoce con exactitud.

>> C=conv(A,B)    %%%Multiplicación de polinomios
C =
     1    -6    11    -6     0    -1     6   -11     6
%%% Es decir, C(x) = A(x)B(x) = x^8 - 6x^7 + 11x^6 - 6x^5 - x^3 + 6x^2 - 11x + 6

>> [Q,R]=deconv(A,B)    %%%División de Polinomios
Q =
     1
R =
     0     8   -16    18    -5
%%% Es decir, A(x)/B(x) = 1 + (8x^3 - 16x^2 + 18x - 5)/B(x)

>> [R,P,K]=residue(A,B) %%%Expansión en fracciones parciales
R =
   30.2500
  -31.0000
    8.7500
    2.5000
P =
    3.0000
    2.0000
    1.0000
    1.0000
K =
     1
%%% Es decir, A(x)/B(x) = 30.25/(x-3) - 31/(x-2) + 8.75/(x-1) + 2.5/(x-1)^2 + 1
```

Gráficas en 2D

Los gráficos son una poderosa herramienta visual para la representación e interpretación de datos, por esta razón Matlab incorpora un poderoso conjunto de herramientas gráficas, el cual es demasiado extenso para tratarse en este curso, de manera que solamente se tratarán los comandos básicos para el trazado de gráficas en 2D y en 3D.

Ejemplo: Obtener la gráfica de la función $f(x)=\frac{1}{1+x^2}$ (curva de Agnesi) en el intervalo de valores de la variable independiente: $-5 \le x \le 5$.

Solución: Sólo hay que elegir un incremento adecuado de los valores de x. En este caso elegimos arbitrariamente 0.01 :

```
x=-5:0.01:5;       %%Vector de valores de x
>> y=1./(1+x.^2); %%Vector de valores de y
>> plot(x,y)       %%grafica
```

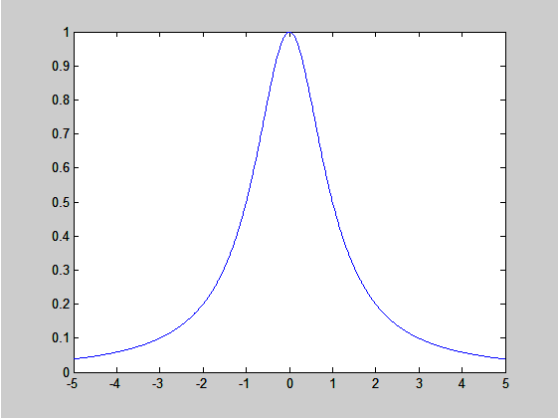

Figura 2.1 Curva de Agnesi

Al ejecutar el comando plot Matlab genera la gráfica de la figura 2.1.

Ejemplo: Usar la gráfica de la curva de Agnesi y de la parábola $f(x)=x^2$ para encontrar visualmente la intersección de ambas curvas.

Solución. Primeramente obtenemos ambas gráficas juntas.

```
>> x=-5:0.01:5;
>> y1=1./(1+x.^2);
>> y2=x.^2;
>> plot(x,y1,x,y2)
```

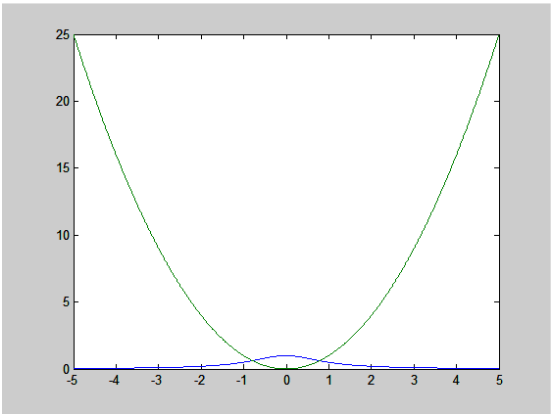

Figura 2.2 Parábola y curva de Agnesi

Con lo cual se obtiene la figura 2.2. Como puede verse, en la figura 2.2 es muy difícil apreciar a simple vista la ubicación exacta de la intersección buscada. Para mejorar la información que nos proporciona la gráfica usaremos los comandos `axis` y `grid`:

```
>> grid on  %%Activa el cuadriculado de la gráfica
>> axis([-1 1 0 1])
%%Reduce rango de visualización de la gráfica:
%%eje horizontal de -1 a 1 y eje vertical de 0 a 1
```

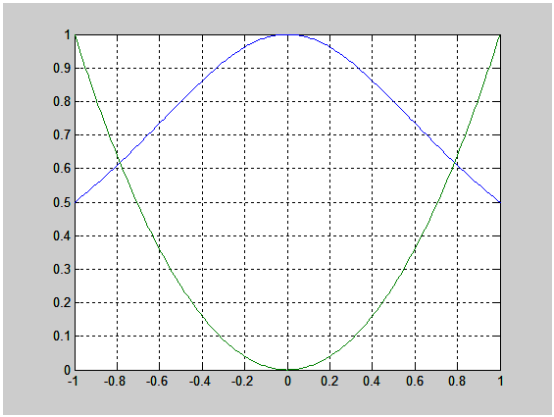

Figura 2.3 Cambio de rango en los ejes

Con esto se obtiene la gráfica de la figura 2.3. En esta figura resulta más fácil identificar a simple vista que las curvas se intersectan aproximadamente en $x=\pm 0.79$ lo cual está muy

cerca del valor exacto $x = \sqrt{\frac{\sqrt{5}-1}{2}}$.

Mejoramiento de gráficas.

Una gráfica bien hecha debe representar la información que se quiere realzar de manera que el usuario que observa la gráfica (no el que la creó) advierta a simple vista lo que se quiere resaltar. Para lograr una buena gráfica hay que considerar lo siguiente:

- La selección de los rangos de los ejes horizontal y vertical es fundamental.
- La selección del incremento de la variable independiente es fundamental.
- Los comandos `grid`, `hold`, `axis`, `xlim`, `ylim`, `zlim`, `title`, `xlabel`, `ylabel`, `zlabel`, `legend`, aunados a las propiedades de linea del comando `plot` pueden ayudar a mejorar considerablemente una gráfica.
- Matlab posee un editor de propiedades de la gráfica, que se puede usar de manera interactiva para mejorar el aspecto de una gráfica.
- Una gráfica se puede almacenar en formato .fig para su posterior edición.

Ejemplo: La gráfica de la figura 2.3 se puede mejorar mediante los siguientes comandos

```
plot(x,y1,'-b','LineWidth',3)
hold on  %%%permite graficar sin borrar la grafica actual
plot(x,y2,'-r','LineWidth',3)
grid on
title('Curva de Agnesi y Parabola','fontSize',16)
xlabel('Variable independiente x','fontSize',14)
ylabel('Variable dependiente y','fontSize',14)
legend('Agnesi','Parabola')
hold off  %%% permite que siguientes gráficas borren la actual
```

En la figura 2.4 se muestran ambas versiones de la gráfica (sin mejorar y mejorada).

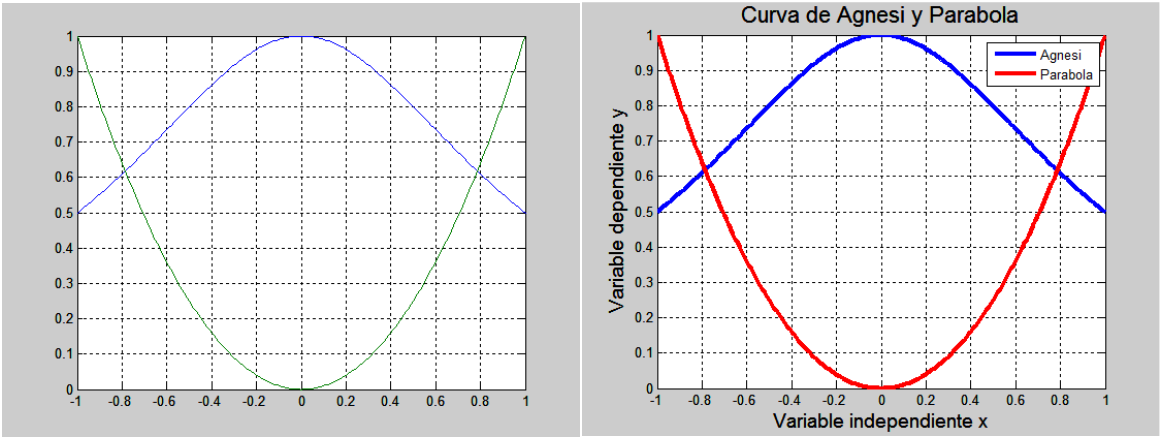

Figura 2.4 La misma figura 2.3 y su versión mejorada.

Se puede obtener una versión aún mejor usando el editor interactivo de Matlab. Al cual se puede acceder sobre la ventana de la gráfica la opción figure properies del menú Edit. En la figura 2.5 se muestra la versión mejorada de la figura 2.4 y su posterior mejora para ser visualizada en papel en tonos de gris.

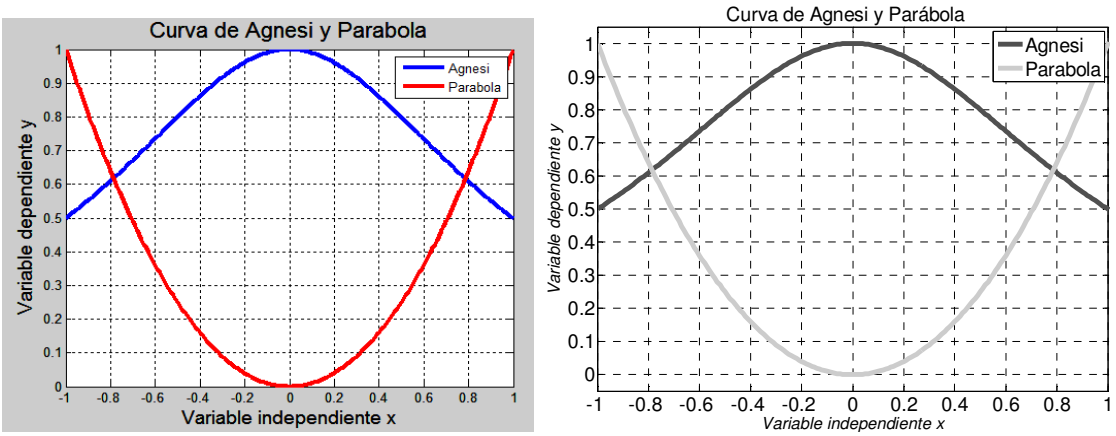

Figura 2.5 Versiones mejoradas para pantalla a color y para impresión en papel en blanco y negro

Modos de copiado de una gráfica.

La versión en tonos de gris mostrada en la figura 2.5 no solamente se ha mejorado mediante el editor de figuras de Matlab, sino que se ha copiado mediante la opción de tipo metafile, mientras que la grafica a color de la figura 2.5 fue copiada mediante la opción de tipo bitmap. Ambas opciones se pueden preestablecer desde el menu edit, en la opción copy options.

- **Copiado tipo bitmap**. Hace una captura de la pantalla con sus atributos de color y resolución.
- **Copiado tipo metafile**. Copia la información de objetos que aparecen en la gráfica, su resolución dependerá del medio final en que se visualice o imprima la figura. Permite la post-edición independiente de cada objeto que aparece en la gráfica.

Selección de la escala e incremento adecuados.

La selección de escala e incremento adecuados depende del tipo de gráfica que se quiere representar, y es un tema que excede los alcances de este instructivo, sin embargo, es fundamental tener presente que deben elegirse con cuidado y no esperar que cualquier elección arbitraria es buena, pues una mala elección puede esconder comportamientos importantes de la gráfica.

Ejemplo. Graficar el polinomio cuyas raíces son 1, 2, y 3.

```
>> p=poly([1 2 3]); %define el polinomio
>> x=0:0.1:10; %Elije un rango arbitrario de valores de x
>> y=polyval(p,x); %Evalúa el polinomio en ese rango
>> plot(x,y); %grafica
>> grid on
```

El resultado se muestra en la parte izquierda de la figura 2.6. En este caso se sabe que el polinomio tiene tres raíces, las cuales no se aprecian en esta figura, evidentemente se eligió mal el rango de valores de x. Se corrige este rango eligiendo ahora $0.5 \le x \le 3.5$ y se obtiene la figura 2.6 de la derecha, en la cual se aprecian claramente las raíces esperadas.

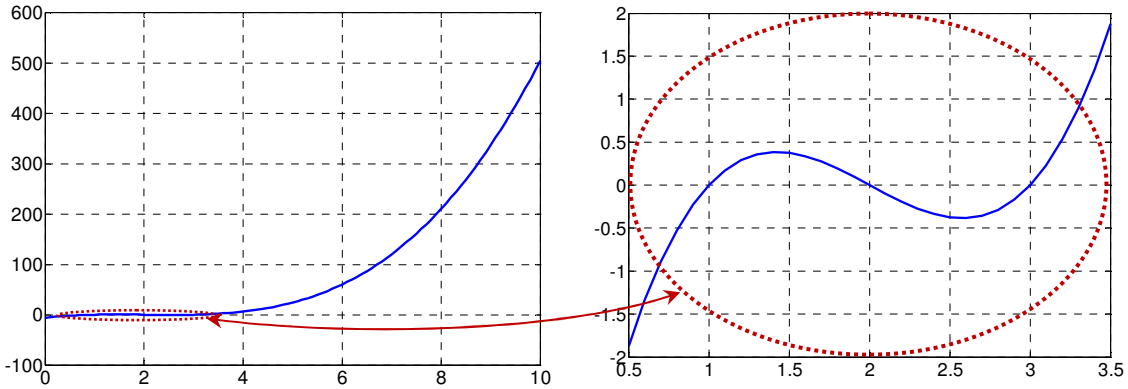

Figura 2.6 Gráfica del polinomio [1 2 3] y mejoramiento del rango horizontal para mostrar las raíces.

Gráficas paramétricas

El comando plot permite asignar valores a cada eje por separado, esto permite realizar el gráfico convencional de f(x) contra x, pero también se puede graficar f1(x) contra f2(x) convirtiéndose entonces x en un parámetro cuyo valor se puede asignar en el rango requerido.

Ejemplo: Obtener la figura de Lissajous correspondiente a la gráfica parámetrica de las funciones

$$y_1(x) = \sin(3x), \quad y_2(x) = \cos(5x)$$

```
>> x=0:0.01:2*pi;
>> y1=sin(3*x);
>> y2=cos(5*x);
>> plot(x,y1,x,y2,'--r');  %% Grafica de y1, y2 contra x
>> legend('y_1=sin(3x)','y_2=sin(5x)')
>> plot(y1,y2);      %% Obtiene figura de Lissajous
>> grid on
```

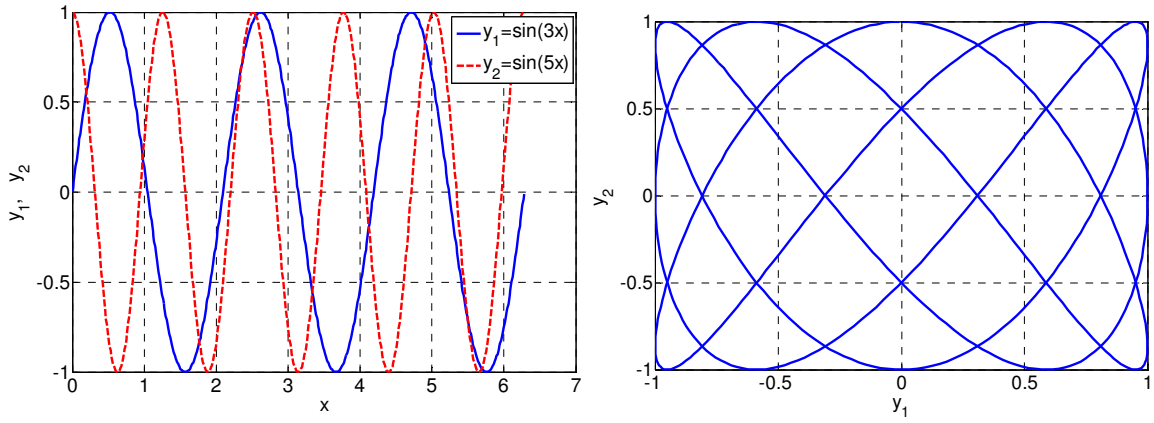

Figura 2.7 Gráficas de y1 y y2 contra x . Gráfica de y2 contra y1

Otras gráficas 2D.

A veces es importante graficar solamente los puntos de una gráfica, (sin una línea que conecte cada punto con el siguiente como lo hace el comando `plot` por default). En ese caso se tienen

opciones similares al comando `plot`:

- Comando `bar`.- En lugar de puntos y líneas usa barras.
- Comando `stairs`.- En lugar de unir puntos consecutivos por una línea, usa un escalón.
- Comando `stem`.- Usa un "tallo" vertical y un círculo pequeño para marcar cada punto.

Ejemplo. Graficar la función tangente trigonométrica en el intervalo de 0 a 2π radianes.

Solución 1: Usando un comando `plot` simple:

```
>> x=0:0.1:2*pi;
>> y=tan(x);
>> plot(x,y); grid on;
```

La grafica obtenida se muestra en la figura 2.8 en el lado izquierdo. La cual no parece ser la grafica de una tangente trigonométrica pues se espera una gráfica discontinua, además se espera que sea periódica cada π radianes, ya que $\tan(x) = \tan(x \pm \pi)$.

Solución 2: Usando un comando `plot` y marcadores punteados en lugar de líneas, Pero además restringiendo el rango vertical de -10 a 10.

```
>> plot(x,y,'o'); ylim([-10 10]);
```

En el lado derecho de la figura 2.8 se muestra el resultado. Obsérvese que ahora si se notan las discontinuidades y la periodicidad esperada cada pi radianes.

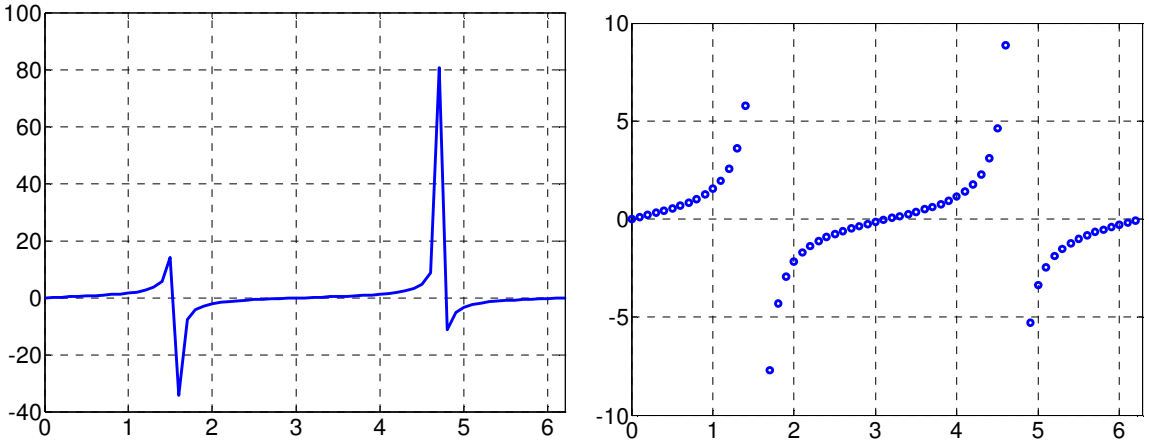

Figura 2.8 Gráfica de tan(x) con una mala elección de escala vertical y tipo de línea y su mejora.

Solución 3: Usando un comando `stem` y las misma restricción del rango vertical de -10 a 10:

```
>> stem(x,y); ylim([-10 10]);
```

En la parte izquierda de la figura 2.9 se muestra la gráfica obtenida con este comando. En la parte derecha de esa misma figura 2.9 se muestra el resultado utilizando el comando `stairs` como sigue:

```
>> stairs(x,y); ylim([-10 10]);
```

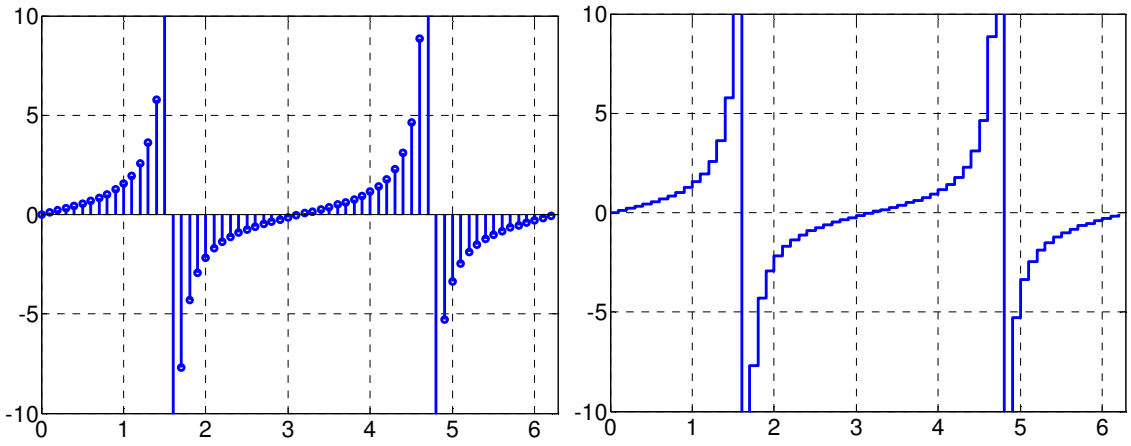

Figura 2.9 Graficas de tan(x) obtenidas mediante `stem` y `stairs`.

Arreglos de Subgráficas.

En ocasiones es conveniente representar en una misma ventana varias gráficas para ilustrar diferentes aspectos de un mismo tema o comportamientos distintos de un mismo sistema, etc. En Matlab una ventana de figura se puede subdividir en un arreglo de m renglones y n columnas de gráficas y cualquier subdivisión puede hacerse activa con el comando subplot (m,n,k), el cual establece como activa el área k-ésima. El número de área (k) se enumera de izquierda a derecha comenzando por el renglón superior y continuando con los renglones de arriba hacia abajo.

Ejemplo. Con los siguientes comandos se genera el arreglo de gráficas mostrado en la figura 2.10

```
>> x=0:0.01:2*pi; %%Primero genera tres funciones sinusoidales
>> y1=sin(x);
>> y2=cos(x);
>> y3=sin(x+pi/3); %% A continuación genera un arreglo de 2x2 gráficas
>> subplot(2,2,1); plot(x,y1); grid on
>> subplot(2,2,2); plot(x,y2); grid on
>> subplot(2,2,3); plot(y1,y2); grid on
>> subplot(2,2,4); plot(y1,y3); grid on
```

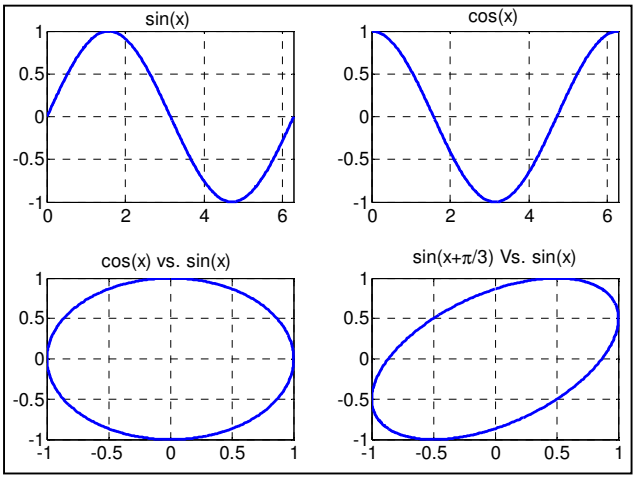

Figura 2.10. Ejemplo de arreglo de subgráficas de 2x2

Consideraciones adicionales:

El comando `plot` crea una gráfica de vectores o columnas de matrices. La forma de la orden es `plot(x1, y1, S1, x2, y2, S2, .....)` donde `(xn, yn)` son arreglos de coordenadas de puntos a graficar y `Sn` son cadenas opcionales que especifican color, marcas, grosor y/o estilos de línea. Algunas de estas cadenas se muestran en la siguiente tabla y se pueden combinar, por ejemplo la cadena `'-or'` especifica una línea roja con marcas circulares.

Símbolo	Color	Símbolo	Estilo de línea
y	amarillo	·	Marca punto
m	magenta	o	Marca círculo
c	cien	X	marca x
r	rojo	+	Marca +
g	verde	*	Marca *
		d	Marca diamante (♦)
b	azul	-	línea sólida (default)
w	blanco	:	línea punteada
k	negro	.-	línea punto-raya
		--	línea de trazos

- El comando `text(x,y,'string')` añade la cadena de caracteres *'string'* a la gráfica actual en las coordenadas especificadas `(x,y)`.
- El comando `gtext('string')` permite colocar texto 'string' interactivamente en la gráfica utilizando el ratón.
- Los comandos `axis, ylim, xlim` cambian los límites de los ejes y con ello la apariencia de las gráficas.
- Se pueden añadir gráficas a la gráfica actual sin borrar lo anterior mediante el comando `hold on`. Haciendo `hold off` se permite que la próxima orden de `plot` borre la ventana de la figura antes de hacer una nueva gráfica.
- Se pueden generar múltiples ventanas de figuras mediante el comando `figure`. El comando `figure(n)` establece la ventana de figura número n como la ventana de figura activa.
- Mediante el comando `zoom on`, la grafica de la figura activa se puede ampliar interactivamente usando el ratón, seleccionando el área de expansión o simplemente con un click del ratón `zoom off` lo desactiva
- El comando `loglog` es lo mismo que `plot`, excepto que se usan logarítmicas para ambos ejes.
- El comando `semilogx` es lo mismo que `plot` excepto que usa una escala logarítmica en el eje `x` y escala lineal en el eje `y`. `semilogy` es lo mismo solo que el eje `y` es el de la escala logarítmica y el eje `x` es lineal.

Gráficas en 3D.

Existen dos tipos básicos de gráficas en tres dimensiones: Líneas curvas en tres dimensiones y superficies en tres dimensiones.

Líneas Curvas en 3D. Estas se pueden trazar mediante el comando `plot3`. Para trazar una línea curva tridimensional, es suficiente con generar tres vectores `(x,y,z)` conteniendo las coordenadas de cada uno de los puntos a graficar. Lo normal es usar un parámetro para generar las coordenadas a partir de la variación de dicho parámetro en un rango de valores.

Ejemplo.

```
>> t=0:0.001:1;      %% Valores del parámetro t
>> x=sqrt(1-t.^2).*sin(24*pi*t); %% Coordenadas x(t)
>> y=sqrt(1-t.^2).*cos(24*pi*t); %% Coordenadas y(t)
>> z=t;  %% Coordenadas z(t)
>> plot3(x,y,z);  %% Grafica línea que une las coordenadas.
>> grid on
```

En este caso, las ecuaciones parámetricas usadas para generar los puntos de la línea corresponden a una espiral que se desenvuelve a lo largo de una esfera. La gráfica obtenida se muestra en la figura 2.11.

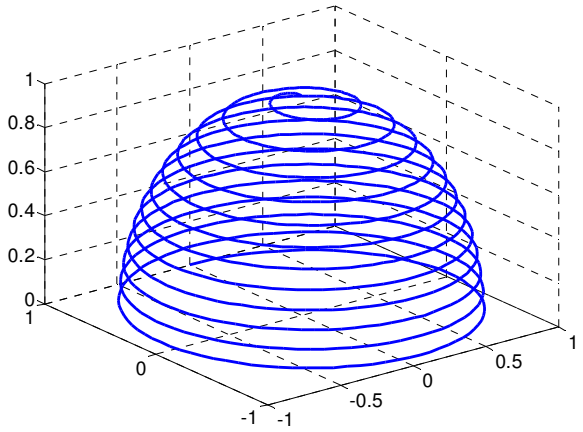

Figura 2.11. Ejemplo de línea tridimensional

Superficies en 3D. Las superficies se pueden trazar mediante los comandos `meshgrid`, `surf`, `meshc`, `meshz` o `waterfall`. Para generar la grafica de una superficie tridimensional primeramente se procede como siguei:

1. Elegir n valores de la variable independiente X,
2. Elegir m valores de la variable independiente Y.
3. Con estos vectores X, Y se generan dos arreglos matriciales x, y, de tamaño mxn conteniendo las coordenadas x, y de cada punto del plano en que se evaluará la función z=f(x,y) que define la superficie a graficar

Ejemplo. Con el siguiente código se genera la gráfica de la figura 2.12.

```
>> X=-3:0.3:3; %%Rango de valores de X (incrementos de 0.1)
>> Y=X;  %% Rango de valores de Y
>> [x,y]=meshgrid(X,Y);  %%Genera el plano de variables independientes
>> z=exp(-0.4*x.^2-0.4*y.^2).*cos(x.^2+y.^2); %% función a graficar
>> mesh(x,y,z)  %%Genera la gráfica
>> grid on
>> xlabel('eje X')
>> ylabel('eje Y')
>> zlabel('eje Z')
```

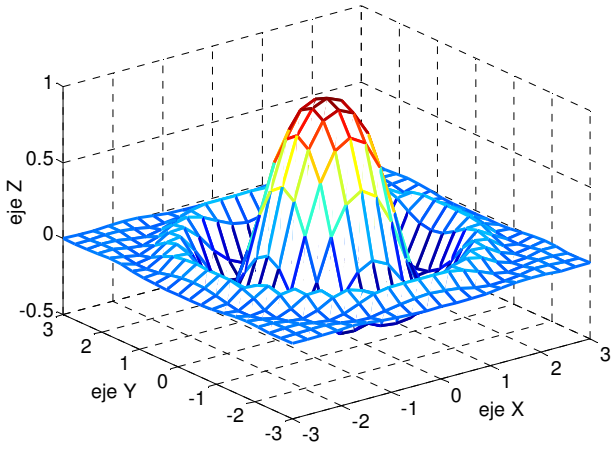

Figura 2.12. Ejemplo de superficie tridimensional

Obsérvese que la gráfica tiene la forma de una red o malla tridimensional, de ahí el nombre del comando (`mesh`). Una mejora visual se puede obtener mediante el comando `surf`, el cual colorea la cuadrícula dando un aspecto más sólido a la superficie. También se puede agregar la información de las ***líneas de nivel***, las cuales representan los cortes a diferentes alturas constantes de las superficies generadas. Esto último se hace mediante `meshc` o `surfc`.

Los diferentes resultados para la misma superficie tridimensional, se pueden observar en la figura. 2.13.

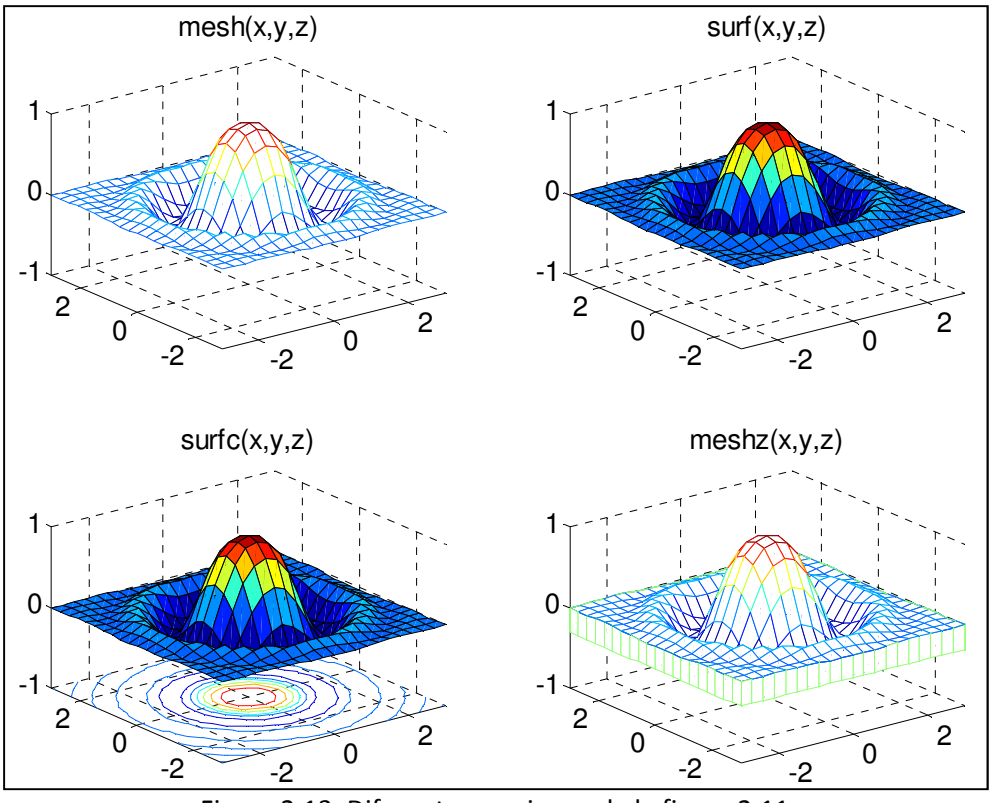

Figura 2.13. Diferentes versiones de la figura 2.11

Desarrollo de la Práctica.

Durante el desarrollo de esta práctica se presentan los comandos para el manejo de polinomios y para generar gráficas en 2D y 3D. En cada explicación se dan ejemplos del funcionamiento del ambiente de Matlab, y se proponen algunos ejercicios.

1. Probar todos los ejemplos propuestos por el profesor conforme los va explicando.
2. Escribir el código para generar la figura 2.13.
3. Contestar el cuestionario de evaluación de la práctica.

Reportar:

1. Reportar el código para generar la figura 2.13.
2. Investigar ¿Que son las líneas de nivel? Explicar en una a dos páginas.
3. Graficar las líneas de nivel etiquetadas mediante los comandos `clabel` y `contour` para la función del último ejemplo.
4. Calcular a mano la división de polinomios y la expansión en fracciones parciales para la expresión $\frac{x^2+1}{(x+1)(x+2)}$ y escribir los comandos de Matlab para verificar el cálculo.
5. Escribe el código para graficar una semiesfera de radio 1 y anexa la gráfica obtenida. Sugerencia: usa valores lógicos para multiplicar por cero los valores complejos que no se pueden graficar.