Práctica 2. Introducción a Scilab II

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

Introducción.

Manejo de Polinomios en Scilab.

Además de las matrices, otro de los objetos básicos en Scilab son los polinomios. Un polinomio en Scilab se puede representar 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 tiene la forma

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

Se representará en Scilab como el vector de n+1 coeficientes:

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

Scilab 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
`coeff(P)`	Obtiene los coeficientes del polinomio P
`poly(c,'x','r'|'c')`	Obtiene el polinomio en potencias de x cuyas raíces o coeficientes son los dados en el vector r
`horner(P,valor)`	Evalúa el polinomio P en el valor indicado.
`conv(A,B)`	Multiplica los polinomios A, B expresados por sus coeficientes
`[R,Q]=pdiv(B,A)`	Realiza la división de polinomios B/A calculando cociente Q y residuo R

Ejemplo 2.1. El símbolo usado para la variable del polinomio es arbitrario y se puede utilizar cualquier literal, por ejemplo, el polinomio los polinomios $A(x) = 4x^3 + 3x^2 + 2x + 1$, $B(x) = x^3 - 1$, $C(t) = t^2 + 3t + 2$ se define en Scilab como sigue:

```
A=poly([1 2 3 4],'x','c') //Define el polinomio A(x) a partir de sus coeficientes
A  =
                2   3
    1 + 2x + 3x + 4x
B=poly([1 0 0 1],'x','c') //Define el polinomio B(x) a partir de sus coeficientes
B  =
          3
    1 + x
C=poly([1 3 2],'t','c') //Define el polinomio C(t) a partir de sus coeficientes
```

```
C  =
                2
    1 + 3t + 2t
```

Dos polinomios en la misma variable se pueden sumar, restar, multiplicar y dividir, por ejemplo:

```
A+B
 ans  =
                2    3
    2 + 2x + 3x + 5x
A-B
 ans  =

           2    3
    2x + 3x + 3x
A*B
 ans  =
                2    3    4    5    6
    1 + 2x + 3x + 5x + 2x + 3x + 4x
A/B
 ans  =
                2    3
    1 + 2x + 3x + 4x
    ----------------
               3
         1 + x
```

Ejemplo 2.2. Para los polinomios $A(s)=s^4+s^3+s^2+s+1$, $B(s)=(s-1)^2(s-2)(s-3)$: a) Calcular sus raíces. b) Escribir ambos en potencias ascendentes. c) Verificar algunas de las raíces calculadas. d) Utilizar el comando `conv` para obtener el polinomio $C(x)=A(x)B(x)$. e) Obtener el cociente y el residuo de la división $A(x)/B(x)$.

Solución:

```
A=poly([1 1 1 1 1],'s','c')   //Define el polinomio A a partir de sus coeficientes
A  =
             2   3   4
    1 + s + s + s + s
// Es decir, A(s)=1+s+s^2+s^3+s^4
B=poly([1 1 2 3],'s','r')     //Define el polinomio B a partir de sus raíces
B  =
                  2    3   4
    6 - 17s + 17s - 7s + s
// Es decir, B(s)=6-17s+17s^2-7s^3+s^4
r=roots(A)              //Calcula las cuatro raíces de A
r =
    0.3090170 + 0.9510565i
    0.3090170 - 0.9510565i
  - 0.8090170 + 0.5877853i
  - 0.8090170 - 0.5877853i
horner(A,0.309017+0.9510565*%i) //Verifica que 0.3090170+0.9510565i es raíz de A
ans =
  - 1.337D-09 + 7.331D-08i
// Obsérvese que no da cero porque se usaron solo 7 decimales
// 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 Scilab
// los almacena todos, pero solo despliega los que requiere el formato actual:
horner(A,r(1))//Verifica la primer raíz calculada
ans =
  6.661D-16 + 1.388D-15i
// Obsérvese que el resultado es casi cero (muy cercano a %eps).

horner(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.

a=coeff(A); // Obtiene los coeficientes de los polinomios
b=coeff(B);
c=conv(a,b) //Obtiene los coeficientes de la Multiplicación de los polinomios
c =
    6.  - 11.    6.  - 1.    0.  - 6.    11.  - 6.    1.
C=poly(c,'s','c') //Obtiene el polinomio correspondiente
C  =
                2   3    5     6    7   8
    6 - 11s + 6s - s - 6s + 11s - 6s + s

[R,Q]=pdiv(A,B)   //División de Polinomios
Q  =
    1
 R  =
                  2    3
  - 5 + 18s - 16s + 8s
// Es decir, A(s)/B(s) = 1 + (8s^3 - 16s^2 + 18s - 5)/B(s)
```

Gráficas en 2D

Los gráficos son una poderosa herramienta visual para la representación e interpretación de datos, por esta razón Scilab 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 2.2. 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
//***obsérvese el espacio antes de ./
plot(x,y)    //grafica
```

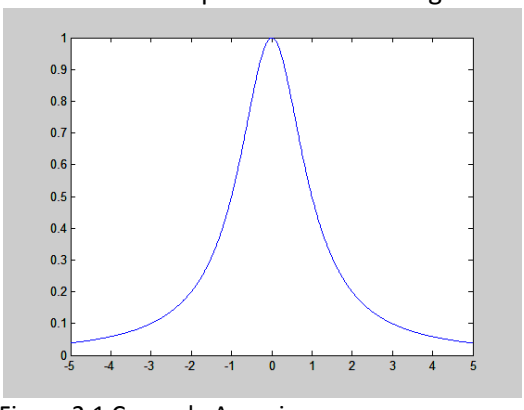

Figura 2.1 Curva de Agnesi

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

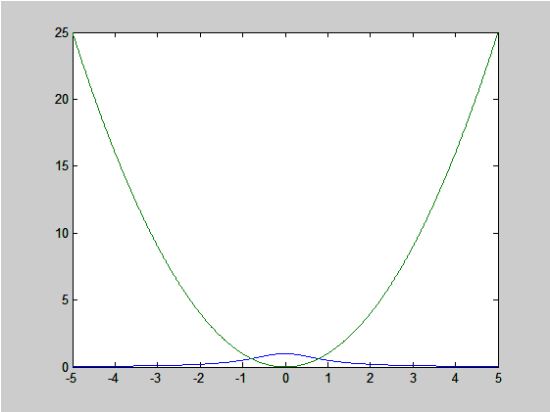

Figura 2.2 Parábola y curva de Agnesi

Ejemplo 2.3. 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;
clf // limpia grafica anterior
plot(x,y1,x,y2)
```

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 accederemos al manejador de propiedades de los ejes de la figura para modificar el rango vertical y horizontal.

```
xgrid(1);
//Activa rejilla de la gráfica en color negro
a=get("current_axes");
//Obtiene manejador de propiedades de los ejes
a.data_bounds=[-1,1,0,1];
//Reduce rango de visualización de la gráfica
//eje x de -1 a 1,eje y de 0 a 1
```

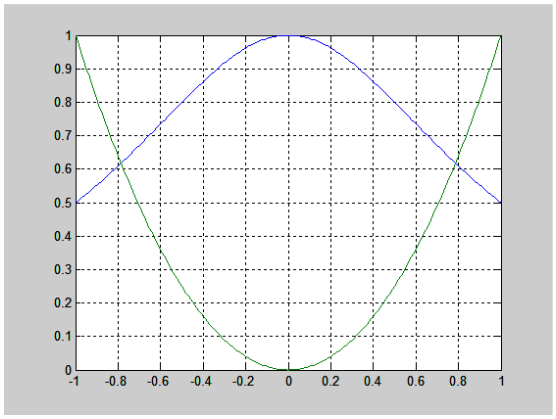

Figura 2.3 Cambio de rango en los ejes

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 mostrar de manera que el usuario que observa la gráfica (no quien 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.
- El uso de una rejilla (`xgrid`) es importante par ayudar a localizar visualmente los valores de interés.
- Los comandos `title, xlabel, ylabel y legend`, aunados a las propiedades de línea del comando `plot` pueden ayudar a mejorar considerablemente una gráfica.
- Se puede tener acceso a las propiedades de la rejilla, del tamaño y estilo de los letreros de los títulos y las etiquetas de los ejes, a través del manejador de los ejes, el cual se obtiene con el comando `get("current_axes")`.
- Scilab 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 .scg para su posterior edición.

Ejemplo 2.4: La gráfica de la figura 2.3 se puede mejorar mediante los siguientes comandos

```
a.auto_clear='on'; //Permite que la nueva grafica borre la anterior
plot(x,y1,'-b','LineWidth',4) //redibuja la curva de Agnesi
a.auto_clear='off'; // desactiva el modo de borrado automatico
plot(x,y2,'-r','LineWidth',4) //redibuja la parabola
xgrid(1); //Activa la rejilla color negro
a.data_bounds=[-1,1,0,1]; // reajusta el rango de los ejes
a.font_size=4; //aumenta el tamaño de las etiquetas de valores de los ejes
//A continuación pone títulos
title('Curva de Agnesi y Parabola','fontSize',5);
xlabel('Variable independiente x','fontSize',4);
ylabel('Variable dependiente y','fontSize',4);
legend('Agnesi','Parabola');
a.auto_clear='on'; //Permite que la nueva grafica borre la actual
```

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

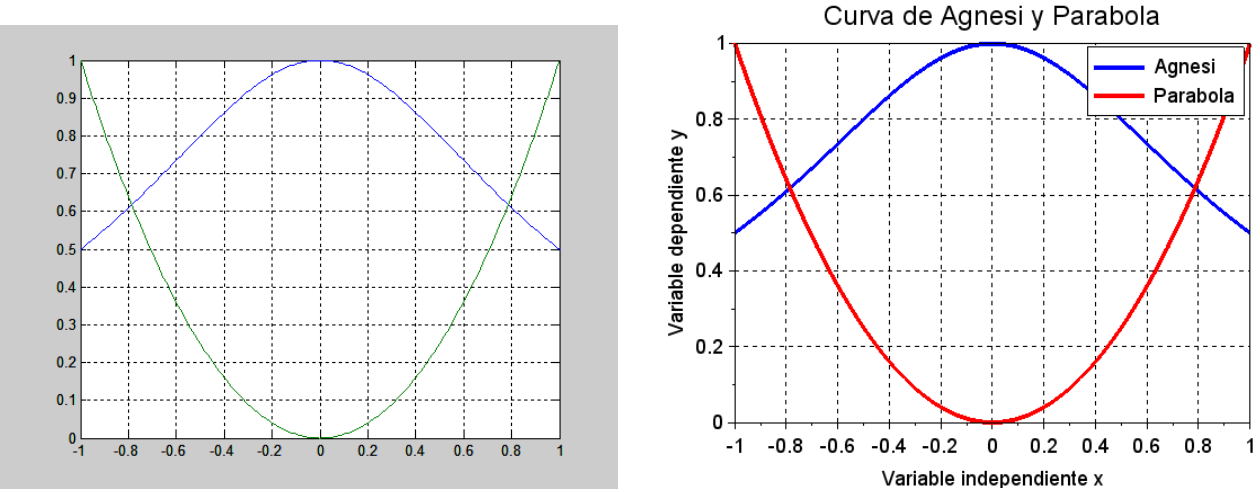

Figura 2.4 La misma figura 2.3 y su versión mejorada.

Normalmente ocurre que una excelente gráfica visualizada en la pantalla a colores no necesariamente resulta en una buena gráfica impresa en papel en blanco y negro o tonos de gris. Por esta razón es recomendable editar los colores de la gráfica de la figura 2.4 para obtener una versión para imprimirse en tonos de gris como la figura 2.5. Para lograr la figura 2.5 se requiere modificar el color de las curvas modificando la propiedad de color `'foreground'` mediante el comando `plot`. El valor de esta propiedad de color se especifica mediante un arreglo de tres componentes [R G B], donde la primera componente (R) es el valor de Rojo, la segunda (G) es valor de Verde y la tercera (B) es el valor de azul. Los valores son fracciones de 0 a 1, por ejemplo

[1 0 0]= rojo puro, [0 1 0]= verde puro, [0 0 1]= azul puro,
[0 0 0]= negro puro, [1 1 1]= blanco puro, [1 1 0]= amarillo puro,
[0.1 0.1 0.1]= gris oscuro, [0.5 0.5 0.5]= gris medio, [0.9 0.9 0.9]= gris claro, etc...

Ejemplo 2.5. El siguiente código es el usado para generar la figura 2.5 para lograr una buena visulización en tonos de gris.

```
clf // limpia grafica anterior
x=-5:0.01:5;
y1=1 ./(1+x.^2);
y2=x.^2;
plot(x,y1,'LineWidth',5,'foreground',[0.8 0.8 0.8]) //curva de Agnesi gris claro
plot(x,y2,'LineWidth',5,'foreground',[0.2 0.2 0.2]) //parábola gris oscuro
a=get("current_axes"); //Obtiene manejador de propiedades de los ejes
a.data_bounds=[-1,1,0,1]; //ajusta rangos: eje x de -1 a 1,eje y de 0 a 1
xgrid(1); //Activa la rejilla color negro
a.font_size=4; //aumenta el tamaño de las etiquetas de valores de los ejes
//A continuación pone títulos
title('Curva de Agnesi y Parabola','fontSize',5);
xlabel('Variable independiente x','fontSize',4);
ylabel('Variable dependiente y','fontSize',4);
legend('Agnesi','Parabola');
a.auto_clear='on'; //Permite que la nueva grafica borre la actual
```

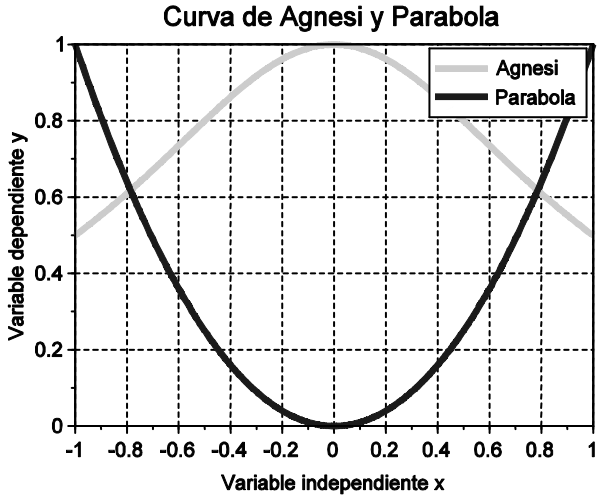

Figura 2.5 Versión mejorada para impresión en papel en tonos de gris

Modos de copiado de una gráfica.

La versión en tonos de gris mostrada en la figura 2.5, se ha exportado a un archivo mediante la opción [**export to**] al tipo encapsulated postScript (EPS) del menú [**File]** de la figura. mientras que la grafica a color de la figura 2.4 fue copiada mediante la opción [**copy to clipboard**], la cual genera un gráfica de tipo mapa de bits o bitmap.

- **Copiado tipo bitmap**. Hace una captura de la pantalla con sus atributos de color y resolución.
- **Copiado tipo encapsulated PostScript (EPS)**. 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 manual, 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 2.6. Graficar el polinomio cuyas raíces son 1, 2, y 3.

```
p=poly([1 2 3],'x','r'); //define el polinomio
x=0:0.1:10; //Elije un rango arbitrario de valores de x
y=horner(p,x); //Evalúa el polinomio en ese rango
clf; //borra la figura anterior
plot(x,y); //grafica
xgrid(1); //Activa la rejilla color negro
```

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 $-2 \le y \le 2$ obteniéndose la figura 2.6 de la derecha, en la cual se aprecian claramente las raíces esperadas.

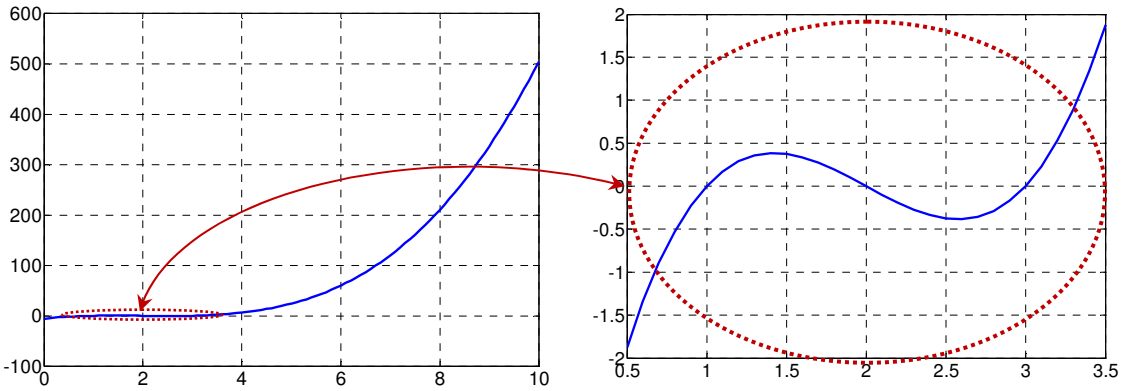

Figura 2.6 Gráfica del polinomio [1 2 3] y mejoramiento del rango horizontal para mostrar las raíces.

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 2.7. Obtener la figura de Lissajous correspondiente a la gráfica paramétrica de las funciones sinusoidales $y_1(x) = \sin(3x)$, $y_2(x) = \cos(5x)$

```
x=0:0.01:2*%pi;
y1=sin(3*x);
y2=cos(5*x);
scf(1);     //Genera nueva figura numerada como figura 1
plot(x,y1,x,y2,'--r');  // Grafica de y1, y2 contra x
legend('y_1=sin(3x)','y_2=sin(5x)');
xgrid(1); //Activa la rejilla color negro
//
scf(2); //Genera otra figura numerada como figura 2
plot(y1,y2);   // Dibuja figura de Lissajous
xgrid(1); //Activa la rejilla color negro
```

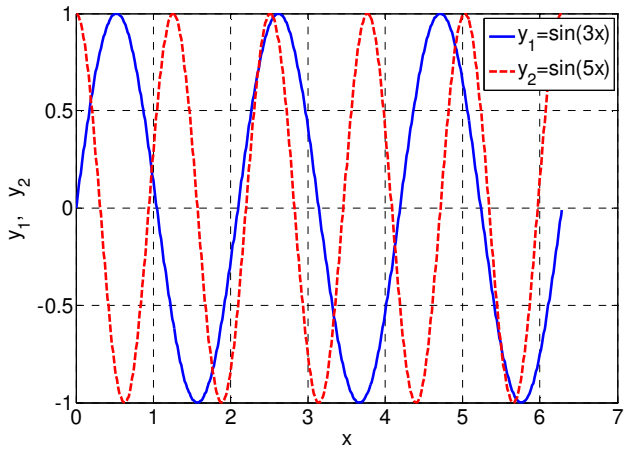

Figura 2.7 Gráficas de y1, y2 contra x .

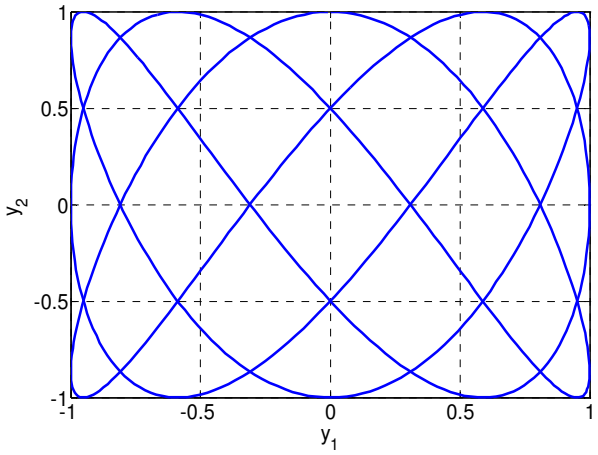

Figura 2.8 Gráfica de y2 contra y1

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 pueden usar los marcadores disponibles en el estilo de línea del comando `plot`:, o utilizar alguno de los siguientes comandos alternativos a `plot`:

- **`plot2d3`.** Gráfica de "tallos" es decir, dibuja líneas verticales desde el eje horizontal hasta los puntos de la gráfica.
- **`plot2d2`**. Gráfica escalonada, es decir, traza líneas horizontales a la altura de cada punto de la gráfica y une cada una con la siguiente mediante trazos verticales.

Ejemplo 2.8. 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);
xgrid(1); //Activa la rejilla color negro
```

La gráfica 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:

```
clf;
plot(x,y,'o');
a=get("current_axes"); //Obtiene manejador de propiedades de los ejes
a.data_bounds=[0,2*%pi,-10,10]; //ajusta rangos: eje x de -1 a 1,eje y de 0 a 1
xgrid(1); //Activa la rejilla color negro
```

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.

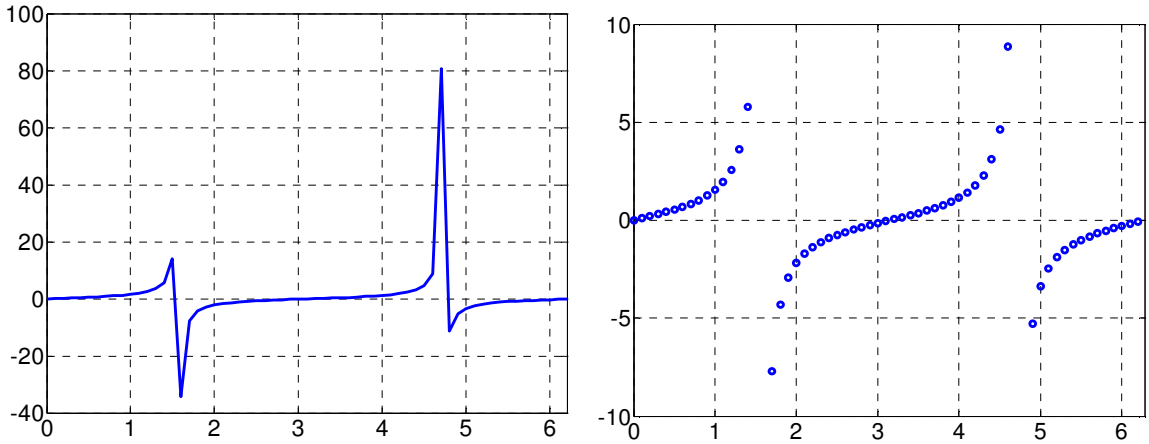

Figura 2.8 Gráfica de tan(x) con una mala elección de escala vertical y tipo de línea y su mejora.

Solución 3: Usando un comando `plot2d2` o `plot2d3` y la misma restricción del rango vertical de -10 a 10:

```
scf(1);
plot2d3(x,y,2); //Gráfica de "tallos"
a=get("current_axes"); //Obtiene manejador de propiedades de los ejes
a.data_bounds=[0,2*%pi,-10,10]; //ajusta rangos: eje x de -1 a 1,eje y de 0 a 1
a.children.children.thickness=2; //grosor de la línea =2
xgrid(1); //Activa la rejilla color negro

scf(2);
plot2d2(x,y,2); //Gráfica escalonada.
a=get("current_axes"); //Obtiene manejador de propiedades de los ejes
a.data_bounds=[0,2*%pi,-10,10]; //ajusta rangos: eje x de -1 a 1,eje y de 0 a 1
a.children.children.thickness=2; //grosor de la línea =2
xgrid(1); //Activa la rejilla color negro
```

En la parte izquierda de la figura 2.9 se muestra la gráfica obtenida con el comando `plot2d3` . En la parte derecha de esa misma figura 2.9 se muestra el resultado utilizando el comando `plot2d2`.

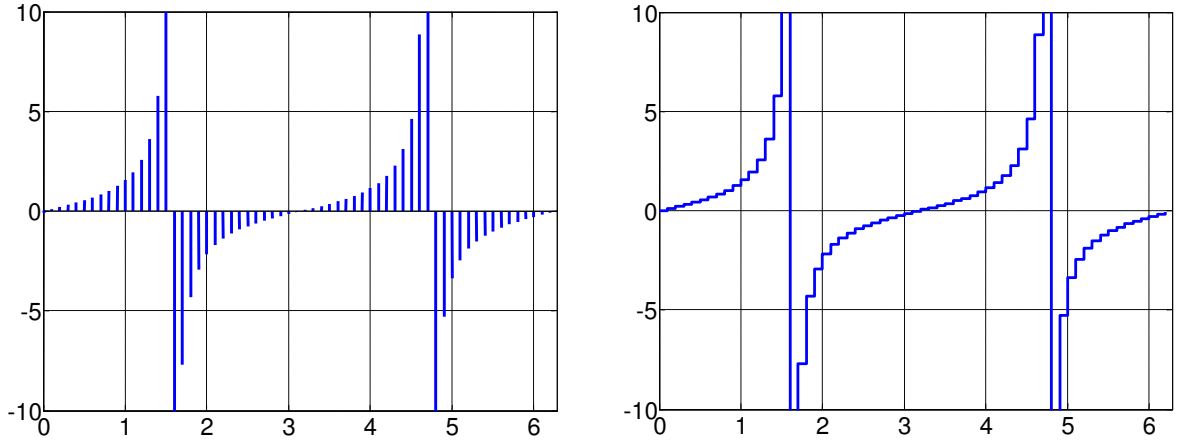

Figura 2.9 Graficas de tan(x) obtenidas mediante `stem` y `stairs`.

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 Scilab 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 2.9. 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
clf;
subplot(2,2,1); plot(x,y1);  xgrid(1); title('sin(x)');
subplot(2,2,2); plot(x,y2);  xgrid(1); title('cos(x)');
subplot(2,2,3); plot(y1,y2); xgrid(1); title('cos(x) vs. sin(x)');
subplot(2,2,4); plot(y1,y3); xgrid(1); title('sin(x+pi/3) Vs. sin(x)');
```

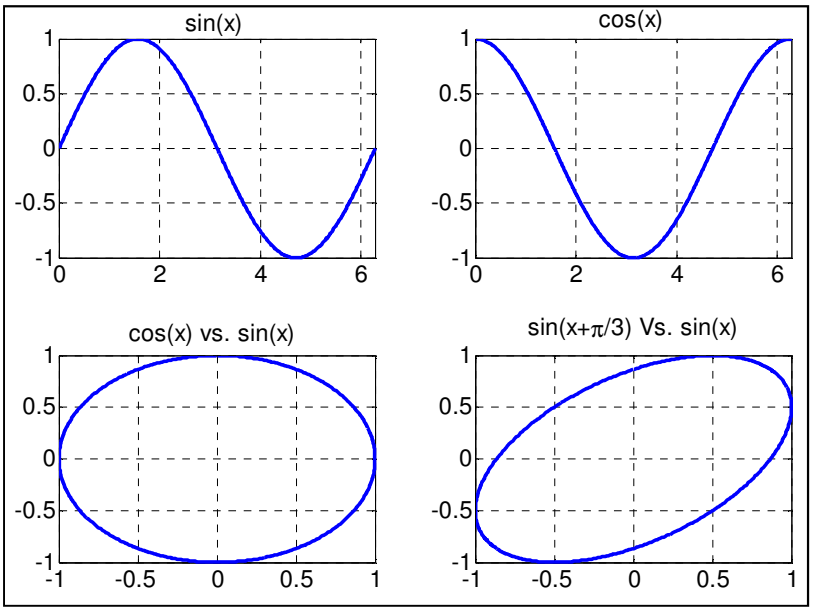

Figura 2.10. Ejemplo de arreglo de subgráficas de 2x2

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	cian	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 **`xstring`**`(x,y,'string')` añade la cadena de caracteres *'string'* a la gráfica actual en las coordenadas especificadas `(x,y)`.
- Por default la propiedad `auto_clear` está en `'off'`, por esta razón se pueden añadir gráficas a la gráfica actual sin borrar lo anterior. Si se desea que un comando gráfico como `plot` borre la ventana de la figura antes de hacer una nueva gráfica se debe poner la propiedad `auto_clear` en `'on'`.
- Se pueden generar múltiples ventanas de figuras mediante el comando `scf`. El comando `scf(n)` establece la ventana de figura número n como la ventana de figura activa.
- Mediante el comando **`zoom_rect`**, se puede ampliar una sección rectangular de la gráfica de la figura activa. El comando **`unzoom`** lo desactiva.
- El comando **`plot2d`** permite usar escalas logarítmicas logarítmicas para el eje vertical, el horizontal o para ambos ejes.

Gráficas en 3D.

En Scilab existen varias maneras de realizar gráficas en tres dimensiones. Sin embargo, en este manual solamente se describirán dos tipos básicos de gráficas en tres dimensiones: Líneas curvas en tres dimensiones y superficies en tres dimensiones.

Líneas Curvas en 3D.

Las líneas curvas en tres dimensiones se pueden expresar por las coordenadas de cada punto de la línea en términos de una variable denominada *parámetro*. Estas curvas se pueden trazar mediante el comando `param3d`. Para trazar una línea curva tridimensional, es suficiente con generar tres vectores `(x,y,z)` conteniendo las coordenadas paramétricas de dicha curva. Cada punto de la curva se obtiene por cada valor del parámetro en un intervalo de valores dado.

Ejemplo 2.10.

```
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)
param3d(x,y,z);    // Grafica línea que une las coordenadas.
a=get("current_axes"); //Obtiene manejador de propiedades de los ejes
a.children.thickness=2; //grosor de la línea =2
a.children.foreground=2; //color de linea = azul
xgrid(1);
```

En este caso, las ecuaciones paramétricas 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.

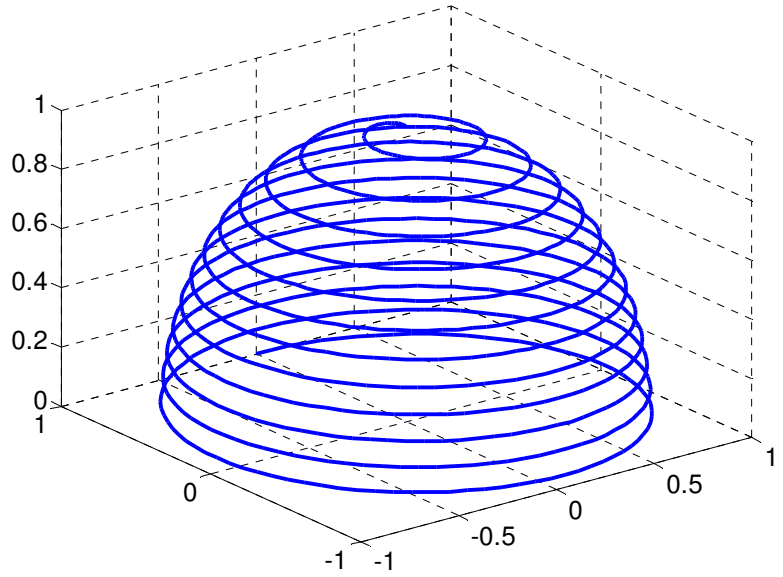

Figura 2.11. Ejemplo de línea tridimensional

Superficies en 3D.

Las superficies se pueden trazar mediante los comandos `meshgrid`, `surf` y `mesh`. Para generar la grafica de una superficie tridimensional primeramente se procede como sigue:

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 m x n 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 2.11. Con el siguiente código se genera la gráfica tridimensional de la figura 2.12. correspondiente a la función de dos variables

$$z = e^{-0.4(x^2+y^2)} cos(x^2 + y^2)$$

```
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
a=get("current_axes"); //Obtiene manejador de propiedades de los ejes
a.children.thickness=2; // grosor de la línea =2
a.children.foreground=2; // color de línea = azul
a.font_size=4; //aumenta el tamaño de las etiquetas de valores de los ejes
xgrid(1);
xlabel('eje X','font_size',4)
ylabel('eje Y','font_size',4)
zlabel('eje Z','font_size',4)
```

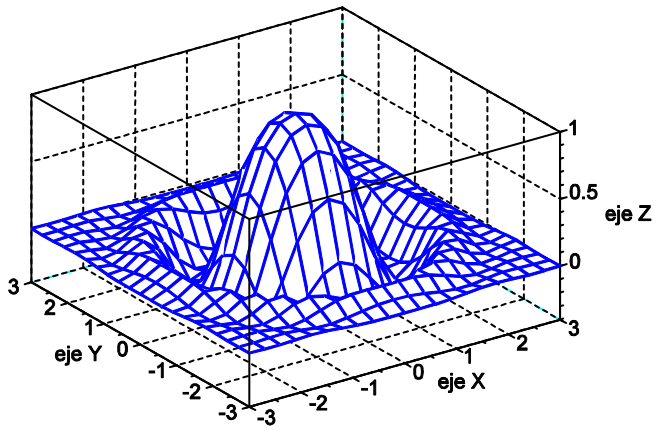

Figura 2.12. Ejemplo de superficie tridimensional

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 `contour`.

Ejemplo 2.12. Uso de los comandos `surf` y `contour`. En este ejemplo se generan cinco versiones (ver figura 2.13) de la gráfica correspondiente a la función de dos variables

$$z = e^{-0.3(x^2+y^2)} cos(x^2 + y^2)$$

```
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(-0.3*x.^2-0.3*y.^2).*cos(x.^2+y.^2); // función a graficar
scf(0); //Crea figura 0
surf(x,y,z);//Genera la superficie 3D con los valores por default
title('Figura con propiedades por default','font_size',4);
f1=scf(1); //genera nueva figura y su manejador
surf(x,y,z);//Genera la superficie 3D
f1.color_map = jetcolormap(32); //Cambia mapeo de colores
title('Figura con mapeo de colores jetcolormap','font_size',4);
f2=scf(2); //genera nueva figura y su manejador
surf(x,y,z,'facecol','interp');//Genera la superficie 3D con color interpolado
f2.color_map = jetcolormap(32); //Cambia mapeo de colores
title('Figura con mapeo de colores Interpolado jetcolormap','font_size',4);
f3=scf(3); //genera nueva figura y su manejador
surf(x,y,z,'facecol','interp');//Genera la superficie 3D con color interpolado
a=get("current_axes"); //Obtiene manejador de propiedades de los ejes
a.children.foreground=15; // color de línea
a.children.thickness=1; // grosor de la línea =1
a.font_size=4; //aumenta el tamaño de las etiquetas de valores de los ejes
f3.color_map = jetcolormap(32); //Cambia mapeo de colores
contour(X,Y,z',10,flag=[0 0 4]); //traza líneas de nivel a 10 niveles
// obsérvese que  se usan los vectores X,Y (no las matrices x,y) y la matriz z
title('Figura con mapeo de colores Interpolado y lineas de nivel','font_size',4);
xgrid(1); xlabel('eje X','font_size',4); ylabel('eje Y','font_size',4);
zlabel('eje Z','font_size',4)
f4=scf(4); //genera nueva figura y su manejador
```

```
a=get("current_axes"); //Obtiene manejador de propiedades de los ejes
a.font_size=4; //aumenta el tamaño de las etiquetas de valores de los ejes
xset("fpf","%.1f"); // Formato de visualización de valores a un decimal
contour(X,Y,z',10); //traza nuevamente las líneas de nivel en 2D
title('Vista superior de las Líneas de nivel','font_size',4);
xgrid(1); xlabel('eje X','font_size',4); ylabel('eje Y','font_size',4)
```

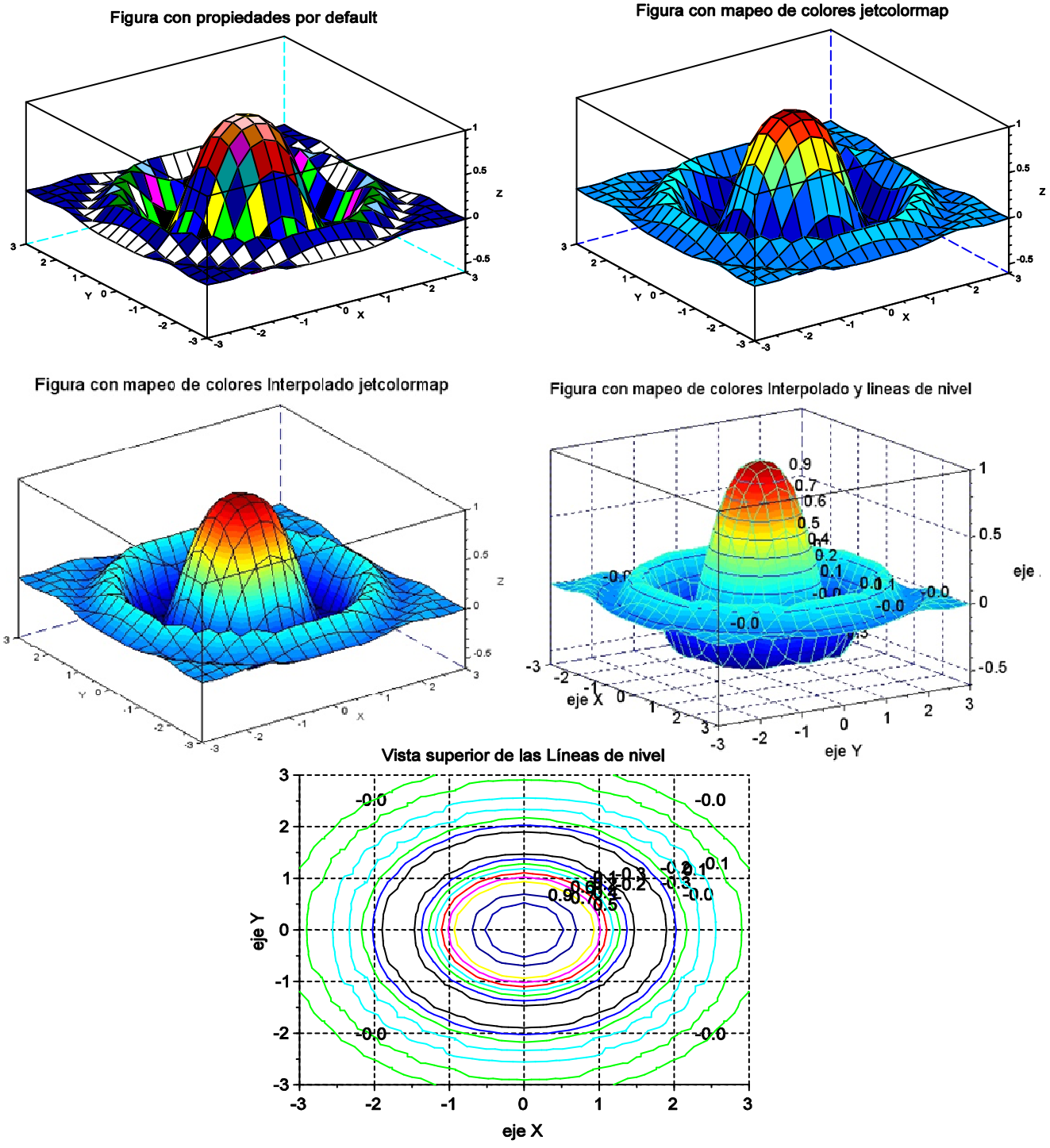

Figura 2.13. Diferentes aspectos de la gráfica de la función $z = e^{-0.3(x^2+y^2)} cos(x^2 + y^2)$

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 Scilab, 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. Investigar ¿Que son las líneas de nivel? Explicar en una a dos páginas.
2. Investigar las opciones para el mapeo de colores de una figura: `colormap`.
3. Obtener la superficie en 3D correspondiente a la función de dos variables $z = e^{-0.4x}\sin(x)cos(x^2 + y^2)$, en el rango $-2.5 \le x \le 2.5$, $-2.5 \le y \le 2.5$, usando el mapeo de color `hsvcolormap` y mostrando las líneas de nivel a 15 niveles sobre la gráfica 3D.
4. Graficar las líneas de nivel en 2D a 15 niveles, correspondientes a la figura anterior.
5. Calcular a mano la división de polinomios para la expresión $\frac{x^2+1}{(x+1)(x+2)}$ y escribir los comandos de Scilab para verificar el cálculo.
6. 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.