

# Prácticas de Matemáticas II

Ingeniería técnica industrial con intensificación en diseño  
industrial

UPNA - Campus de Tudela

## Sesiones I y II

VÍCTOR DOMÍNGUEZ BÁGUENA

Octubre 2009



# Prácticas Matlab 1

## Sesión 1 y 2: Dibujos bi y tridimensionales con Matlab

### 1.1. Introducción y objetivos

El objetivo de esta primera sesión de prácticas es doble, por un lado introducirnos al muy extenso mundo de las salidas gráficas<sup>1</sup> en Matlab y por otro lado visualizar lo que de palabra hemos visto en clase.

Se han adjuntado las gráficas resultantes de muchos de los ejemplos presentados en estas notas. Las puedes consultar al final de estas notas.

### 1.2. Gráficas de funciones de dos variables

Para empezar, introducimos la función que vamos<sup>2</sup> a dibujar

```
>>f=vectorize(inline('x^4-x*y^3+1'))
```

`inline`  
`vectorize`

Seguidamente vamos a crear *mall*a, donde dibujaremos la función. Este proceso es automático y sigue los siguientes pasos

```
>>x=linspace(-2,2,20); y=linspace(-3,3,20);  
>>[X,Y]=meshgrid(x,y);
```

En la primera instrucción estamos tomando 20 puntos entre  $-2$  y  $2$  en la dirección  $X$  y 20 puntos en la dirección  $Y$ , en este caso en  $[-3, 3]$ . El comando `meshgrid` se encarga de entrelazar  $X$  e  $Y$  para crear la malla (rejilla) sobre la que dibujaremos. Podéis visualizar  $X, Y$  simplemente ejecutando

```
>>X, Y
```

Podemos proceder a dibujar la función con el comando `surf`<sup>3</sup>

`surf`

---

<sup>1</sup>De facto, constituye un estándar en el ambiente científico en general e ingenieril en particular

<sup>2</sup>La orden `inline` define la función mientras que `vectorize` permite que ésta se pueda aplicar sobre listas de números.

<sup>3</sup>de *surface*

```
>>surf(X,Y,f(X,Y))
```

Se desplegará una ventana aparte con la gráfica de la función (ver Figura 1.1). Observa que en este dibujo  $x \in [-2, 2]$ ,  $y \in [-3, 3]$  y que se han utilizado 20 puntos para dibujar en cada dirección.

La ventana dispone de una serie de menús que permite editar los atributos del dibujo. No nos extenderemos demasiado. Es uno de esos casos en los que es mejor *trastear* que leer *sesudos* manuales. De todas formas, señalaremos que entre las opciones posibles está la de exportar el dibujo a diferentes formatos (jpg, eps, pdf,...), rotar la figura, realizar animaciones simples y editar diversos atributos del dibujo

Las instrucciones anteriores se pueden empaquetar en un fichero `script` que permite la edición y ejecución más simple de los comandos utilizados en el dibujo. Para ello, escribir primero

```
>> edit grafica
```

nos mostrará un aviso diciendo que el fichero `grafica.m` no existe, por tanto no se puede editar, y nos preguntará a continuación si deseamos crearlo. Decimos que sí y escribimos:

```
f=vectorize(inline('x^4-x*y^3+1'))
x=linspace(-2,2,40); y=linspace(-3,3,40);
[X,Y]=meshgrid(x,y);
surf(X,Y,f(X,Y))
```

Al ejecutar en la línea de comandos `grafica`, nos mostrará de nuevo la gráfica salvo que estará dibujada con mayor precisión dado que ahora tomamos 40 puntos en cada eje (línea 2). Maneja con cuidado esta opción dado que duplicar el número de puntos en cada eje supone multiplicar por cuatro el número de puntos que se utilizan al dibujar, por lo que si se toma un número moderadamente alto de puntos puedes *sobrecargar* al ordenador.

El color de la superficie se fija de acuerdo a la *altura* de cada punto, esto es, al valor de  $f(x, y)$ . Tecleando

```
>> colorbar
```

mostramos una barra lateral donde se informa de la correspondencia entre color y valor. Podemos modificar la situación de esta barra editando los atributos del dibujo o bien manejando alguna de los argumentos opcionales de esta instrucción. Por ejemplo

```
>> colorbar off; % eliminamos la barra de colores anterior
>> colorbar('SouthOutside')
```

dibuja la barra de colores debajo de la gráfica.

Quedan pendientes dos detalles más: la escala del dibujo y el patrón de colores utilizados en la gráfica. Para el primero contamos con el comando `axis` que admite diferentes opciones entre las que destacamos<sup>4</sup>

<sup>4</sup>Pruébalos...

```
auto tight equal image square normal vis3d
```

El patrón de colores se ajusta con `colormap`. Se puede hacer manualmente o bien utilizar algunos de los ya establecidos:

```
autumn    bone    colorcube  cool    copper    flag
hot       hsv     jet        lines   pink      prism
spring    summer  white     winter
```

Por ejemplo

```
g=vectorize(inline('sin(4*sqrt(x^2+y^2))./(sqrt(x^2+y^2))'))
x=linspace(-8,8,60); y=linspace(-8,8,60);
[X,Y]=meshgrid(x,y);
surf(X,Y,g(X,Y))
axis square
colormap('hot')
colorbar('westoutside')
title('Funcion sin(4*sqrt(x^2+y^2))./(sqrt(x^2+y^2))')
xlabel('OX')
ylabel('OY')
zlabel('OZ')
```

despliega la Figura 1.2

En este ejemplo hemos utilizado algunos comandos nuevos, `title`, para colocar un título al dibujo, `xlabel`, `ylabel`, `zlabel` para colocar tres etiquetas en los ejes.

```
title
xlabel
ylabel
zlabel
```

### Atributos opcionales

Es posible dar un aspecto más personal al dibujo recurriendo a comandos opcionales de `surf`. Por ejemplo

```
g=vectorize(inline('sin(4*sqrt(x^2+y^2))./(sqrt(x^2+y^2))'))
x=linspace(-8,8,60); y=linspace(-8,8,60);
[X,Y]=meshgrid(x,y);
surf(X,Y,g(X,Y),'edgecolor','none','facecolor','interp')
colormap(hot)
```

```
edgecolor
facecolor
```

indicamos que la rejilla (`edge`) no se dibuje y que el color sobre cada una de las caras que define el dibujo no sea uniforme (la misma en toda la *celda*) sino interpolado, lo que dota de mayor suavidad a la gráfica (compara las Figuras 1.2 y 1.3).

### Nota

Buena parte de estos atributos del dibujo se pueden modificar en la ventana. La ventaja de esta última aproximación es clara: no hay que memorizar

complicadas instrucciones y se puede ajustar el dibujo hasta obtener un resultado satisfactorio.

Por contra, modificar los atributos mediante comandos tiene una ventaja evidente: una vez fijado los aspectos del dibujo, se puede utilizar para producir múltiples gráficas según el mismo formato. No hay por tanto necesidad de modificar el aspecto manualmente a cada una de las gráficas.

Entre los comandos que configuran estos atributos destacamos

```
view    xlim    ylim    zlim    grid
```

`xlim`      Los comandos `xlim`, `ylim` y `zlim` establecen los límites en los ejes OX, OY y OZ del dibujo.

`ylim`  
`zlim`  
`view`      El comando `view` permite fijar el punto desde donde se observa el dibujo. En particular, `view(2)`, establece que el dibujo es bidimensional, mientras que `view(3)`, que se trata de un dibujo en 3D

`grid`      Finalmente, con `grid` se puede activar (`grid on`) o desactivar (`grid off`) la malla que aparece en el dibujo.

`get, set`      Además los comandos `get` y `set` permiten leer y fijar todos los atributos de la figura y del marco donde se dibuja. Una explicación más detallada de estos comandos excede los objetivos de unas prácticas como éstas. Existen, no obstante, manuales y libros que tratan este tema de forma adecuada.

## 1.3. Otros comandos de dibujo

Revisamos a continuación otras instrucciones ligadas con los dibujos y tres dimensiones. No son menos importantes que `surf`, sino que más bien comparten un núcleo esencial (la ventana gráfica) y en muchos casos parámetros similares que han sido ya expuestos en la sección anterior.

### 1.3.1. Gráficas y líneas de contorno de funciones de dos variables

**Líneas de contorno:** `contour`, `contourf`

Los comandos `contour` y `contourf`, en pocas palabras, dibujan curvas de nivel,. El primero de ellos, además, colorea el espacio entre ellos

```
contour
contourf
f=vectorize(inline('x^4-x*y^3+1'))
x=linspace(-2,2,40); y=linspace(-3,3,40);
clf %% borramos figuras
subplot(121) % dos graficas en 1x2. Ventana 1
[X,Y]=meshgrid(x,y);
contour(X,Y,f(X,Y))
colorbar % barra de colores
subplot(122) % dos graficas en 1x2. Ventana 2
[X,Y]=meshgrid(x,y);
contourf(X,Y,f(X,Y))
colorbar
```

En la línea anterior aparecen dos comandos nuevos, `clf` borra la figura completa, mientras que `subplot` permite crear varios *subdibujos* en la misma ventana gráfica. El resultado se muestra en la Figura 1.4.

`clf`  
`subplot`

Es posible colocar unas *etiquetas* sobre el dibujo. Para ello, hay que leer ciertos datos que devuelve `contour` (o `contourf`) para seguidamente colocar los datos con `clabel`:

`clabel`

```
f=vectorize(inline('x^4-x*y^3+1'))
x=linspace(-2,2,40); y=linspace(-3,3,40);
clf
[c,h]=contour(X,Y,f(X,Y))
clabel(c,h)
```

En la Figura 1.5 se muestra la salida obtenida. El número de líneas de nivel, y por tanto qué valores se toman las curvas de nivel, se fija de forma automática. De todas formas es posible seleccionar de forma explícita ambos valores (ver Ejercicio 1.3).

### Comando mesh

Esta instrucción es similar a `surf`, pero sólo dibuja la rejilla que conforma el dibujo.

`mesh`

### Comandos surfc y meshc

El comando `surfc` superpone la gráfica de la función con las líneas de contorno trazadas en la parte inferior de la gráfica.

`surfc`

Por su parte `meshc` es similar, pero con la superficie de la función sin colorear, es decir, tal y como la devuelve `mesh`.

`meshc`

## 1.3.2. Curvas en el plano y espacio: plot, plot3

Dibujan curvas en el plano y el espacio. Su funcionamiento es muy simple. Las líneas (observa cómo se especifica  $\pi$  en Matlab)

`plot3`  
`pi`

```
t=linspace(0,8*pi,200); % 200 puntos entre 0 y 8*pi
plot3(t.*cos(t),t.*sin(t),t) % Recuerda .* en vez de *
```

crean la Figura 1.6. De nuevo es posible editar el dibujo para darle un formato personal, vía la ventana gráfica o mediante comandos. A modo de ejemplo

```
>> plot3(t.*cos(t),t.*sin(t),t,'r:')
```

dibuja la curva en rojo y en lugar de con línea continua, punteada.

El comando `plot` es similar, pero en este caso dibuja curvas en  $\mathbb{R}^2$ , esto es, funciones de dos componentes y una variable real.

Los comandos `comet`, `comet3` permiten realizar animaciones donde se muestra el recorrido de una curva.

### 1.3.3. Superficies paramétricas: surf

El comando `surf` es mucho más flexible de lo que hemos visto hasta ahora. Una pequeña modificación permite dibujar superficies dadas en paramétricas. A modo de ejemplo, la superficie

$$(\cos^3(u) \operatorname{sen}(v), \operatorname{sen}(u) \operatorname{sen}(v), \cos(v)), \quad u, v \in [0, 2\pi] \times [0, \pi]$$

se puede dibujar con

```
u=linspace(0,2*pi,30); v=linspace(0,pi,30);
[U,V]=meshgrid(u,v); % U,V es la malla
surf(cos(U).*sin(V).^3, sin(U).*sin(V), cos(V)) %Recuerda .^
```

La gráfica así obtenida se puede ver en la figura 1.7. De nuevo se pueden utilizar atributos opcionales que modifiquen aspectos determinados del dibujo. Por ejemplo (observa el uso de “...” para saltar de línea),

`facecolor`  
`facealpha`

```
surf(cos(U).*sin(V).^3, sin(U).*sin(V), ...
     cos(V), 'facecolor', 'g', 'facealpha', 0.4)
```

fija un color verde uniforme para toda la superficie y especifica un nivel de transparencia para mostrar detalles ocultos de la superficie. El resultado se muestra en la Figura 1.8.

### Gráficas de funciones sobre superficies

La idea es dibujar, sobre una superficie, los valores de una función. Un ejemplo muy explicativo lo encuentras en estas líneas:

```
[x,y,z]=sphere(40); % x,y,z dan una esfera
subplot(211) % dos areas de dibujo, una sobre otra.
% Dibujamos en la primera
f=vectorize(inline('x^2*y^3*z^4-x^4*y^3*z^2'))
surf(x,y,z,f(x,y,z))
colorbar
axis equal
subplot(212) % dos areas de dibujo, una sobre otra.
% Dibujamos en la segunda
surf(x,y,z,f(x,y,z), 'facecolor', 'interp', 'edgecolor', 'none')
colorbar
axis equal
```

que dibujan la función  $f(x, y, z) = x^2 y^3 z^4$  evaluada sobre la esfera unidad. El resultado se muestra en la Figura 1.10. Es muy recomendable desplegar la barra de colores pues a priori es la única forma de obtener información sobre los valores de la función ya que ofrece una equivalencia entre colores y valores.

`edgecolor`

En la figura anterior, hemos utilizado una nueva instrucción `edgecolor`, que fija el color utilizado en la rejilla. El valor dado, `none`, simplemente la elimina del dibujo.

## 1.4. Campos vectoriales

Los comandos para dibujar campos vectoriales en Matlab son `quiver` (en 2D) y `quiver3`<sup>5</sup> (en 3D). Su funcionamiento es como sigue: se fija una malla y sobre ella se especifican los vectores. Por ejemplo

`quiver`  
`quiver3`

```
x=linspace(-2,2,16);
y=linspace(-2,2,16);
[X,Y]=meshgrid(x,y); % Malla en 2D
fx=vectorize(inline('3*x^2-4*y^2+1','x','y'))
fy=vectorize(inline('-8*y*x','x','y'))
quiver(X,Y,fx(X,Y),fy(X,Y)) % observa .^ y .*
```

dibuja el campo vectorial dado por

$$(3x^2 - 4y^2 + 1)i - (8yx)j.$$

Si superpones sobre este dibujo las curvas de nivel correspondientes a la función  $f(x, y) = x^3 - 4y^2x + x$ , observarás que el campo es ortogonal a las curvas de nivel<sup>6</sup>. Eso se consigue vía las instrucciones:

`hold`

```
hold on % activamos superposicion; se desconecta con hold off
c=contourf(X,Y,X.^3-4*Y.^2.*X+X,12), colorbar
clabel(c); % colocamos los valores sobre la curva de nivel.
```

El funcionamiento de `quiver3` es similar. Ambas funciones necesitan un mallado (variables X, Y y Z para `quiver3`) construido según los patrones de Matlab.

### Aplicación al cálculo de los extremos condicionados

Se puede visualizar el Teorema de los multiplicadores de Lagrange sin más que superponer el gradiente de la función de la que se quiere calcular los extremos sobre la curva que marca la restricción.

Veamos un ejemplo resolviendo (gráficamente) el siguiente problema:

Extremos de  $f(x) = x^2 + y^4 - 2x$  restringido a  $4 \exp(-x^4 - y^2)(x^2 + y^4) = 1$ .

Geoméricamente, las curvas de nivel marcan los puntos de  $f$  que toman igual valor. Por otro lado, la restricción  $g(x, y) = 0$  es equivalente a buscar máximos y mínimos de  $f$  cuando los puntos se mueven en en la curva de nivel de  $g$  correspondiente al 0. Es fácil comprobar que necesariamente, un punto es máximo o mínimo restringido de  $f$  si las curva de nivel de  $g(x, y) = 0$  es tangente a una curva de nivel de  $f$ . Ahora bien, tanto el gradiente de  $f$  como el de  $g$  son perpendiculares a sus curvas de nivel y por tanto la condición anterior se reescribe en los siguientes términos: en los máximos y mínimos de  $f$  condicionados a  $g$

<sup>5</sup> *quiver* es la palabra en inglés para carraj. ¿Sabes lo que significa?.

<sup>6</sup> ¿Por qué?

los vectores  $\nabla f$  y  $\nabla g$  son paralelos. Lo anterior es válido siempre y cuando la curva de nivel definida por  $g(x, y)$  sea *derivable*, esto es, que no tenga picos porque en este caso no se puede hablar de vector perpendicular. Precisamente, la condición de  $\nabla g(x, y, z) \neq \mathbf{0}$  fijada en el Teorema de multiplicadores de Lagrange evita esta situación y un extremo de  $f$  condicionado a  $g$  en un punto así no tiene porque satisfacer la situación geométrica anteriormente descrita.

Las siguientes líneas

`linewidth`

```

clf %borramos la pantalla
g=vectorize(inline('4*exp(-x^4-y^2)*(x^2+y^4)-1')); % restriccion
x=linspace(-2.5,2.5,30);
y=linspace(-2.5,2.5,30);
[X,Y]=meshgrid(x,y); % malla
contour(X,Y,g(X,Y),...
        [0 0], 'linewidth', 2); % Línea de nivel para el
                                % correspondiente a g(x,y)=0

hold on
fi=vectorize(inline('2*x', 'x', 'y')); % termino i del campo
fj=vectorize(inline('2*y-2', 'x', 'y')); % termino j del campo
quiver(X,Y,fi(X,Y),fj(X,Y)) % superponemos el campo
f=vectorize(inline('x^2+y^2-2*y'));
contour(X,Y,f(X,Y), ':-', 'linewidth', 2), colorbar
axis equal

```

generan la Figura 1.10. En la última línea hemos utilizado el atributo `linewidth` (disponible también en comandos como `plot` o `plot3` para realzar el trazado de las curvas de nivel de  $f$  y  $g$ . Se puede comprobar a simple vista que hay al menos cinco posibles extremos.

## 1.5. Figuras

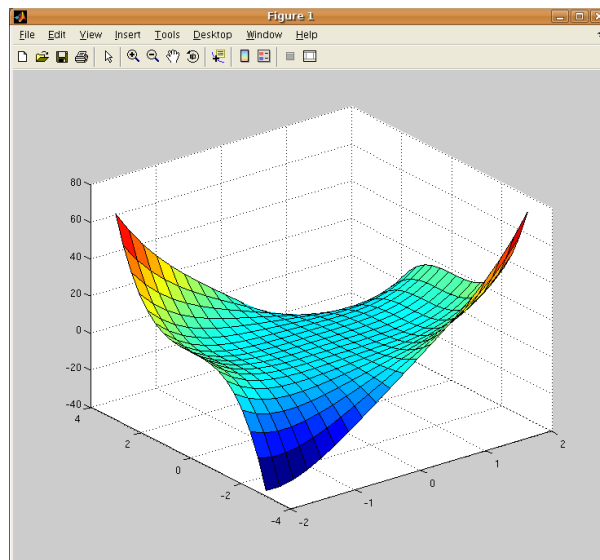


Figura 1.1: Primera gráfica en Matlab de una función de dos variables

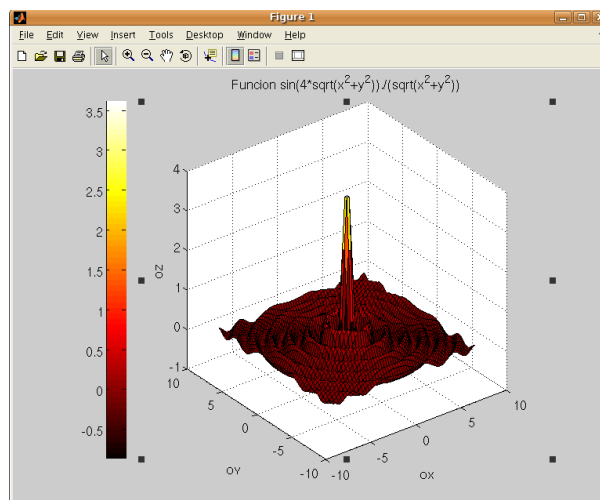


Figura 1.2: Otro ejemplo

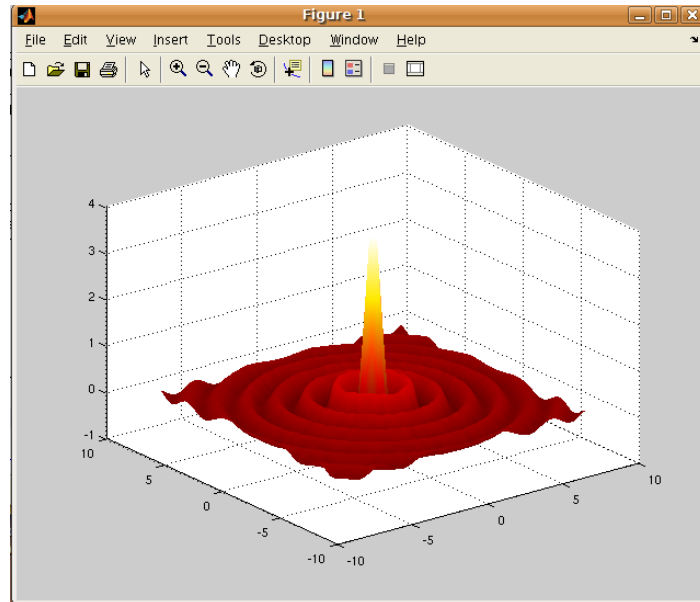


Figura 1.3: Superficie dibujada con algunos argumentos opcionales de surf

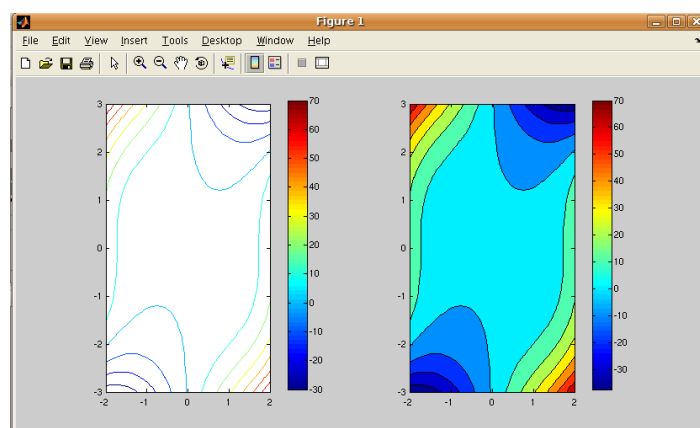


Figura 1.4: Curvas de contorno con contour (izqda) y contourf (dcha)

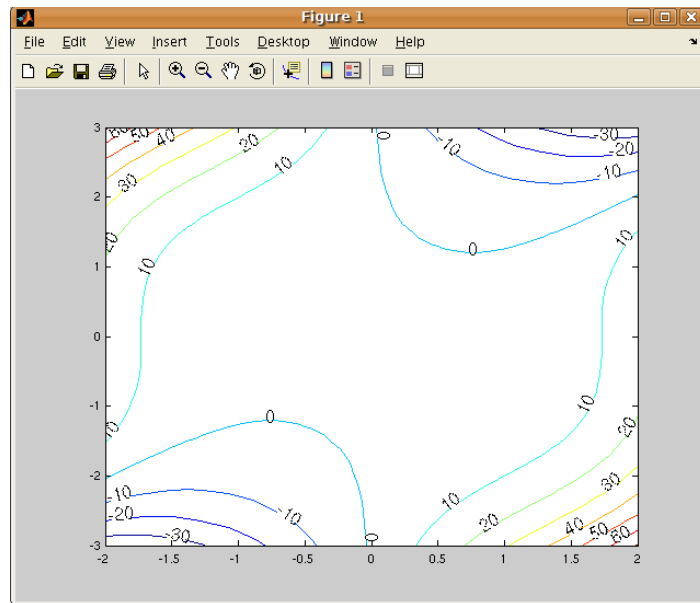


Figura 1.5: Curvas de contorno con *etiquetas*

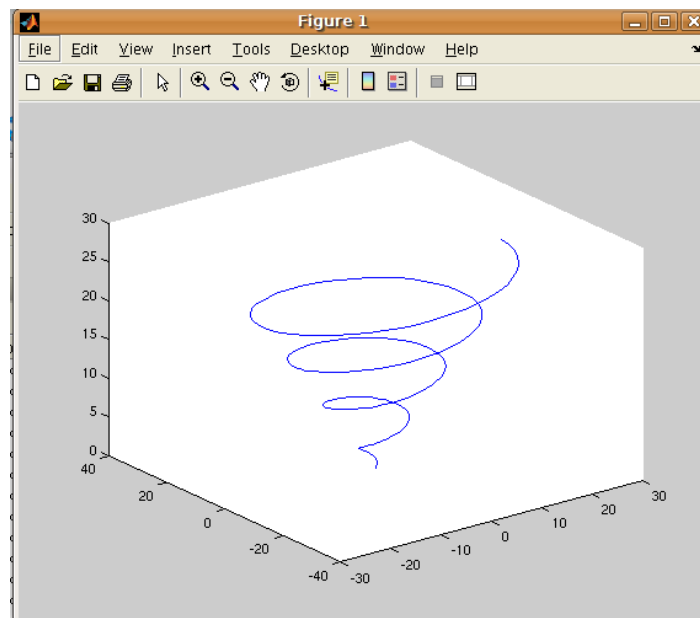


Figura 1.6: Curvas con `plot3`

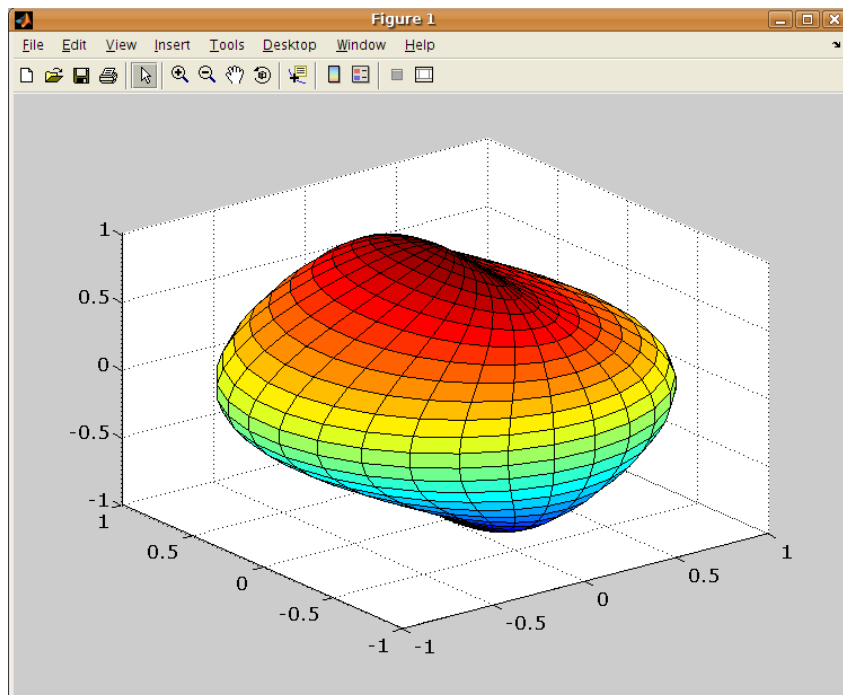


Figura 1.7: Una superficie en paramétricas

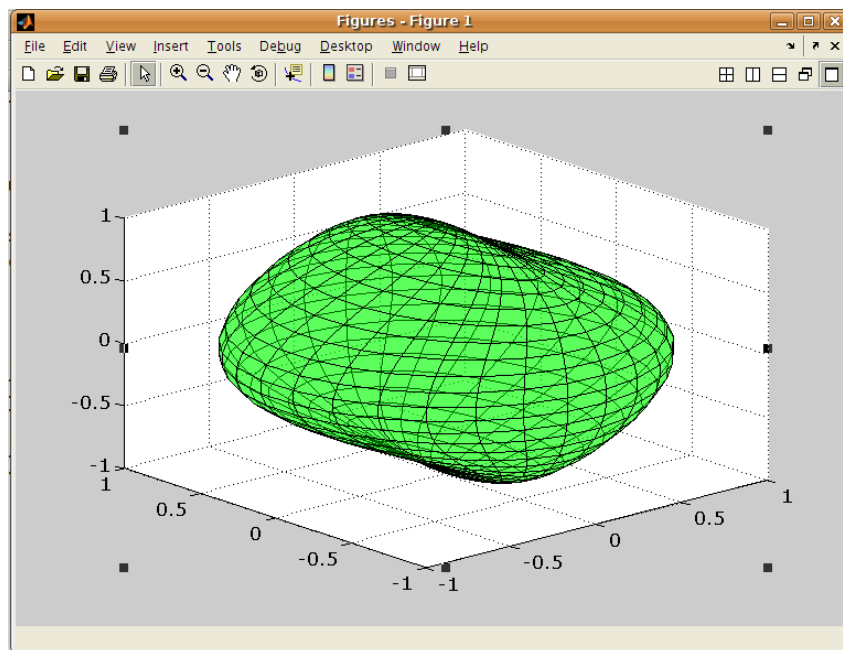


Figura 1.8: La superficie de la figura 1.7 con algunos retoques

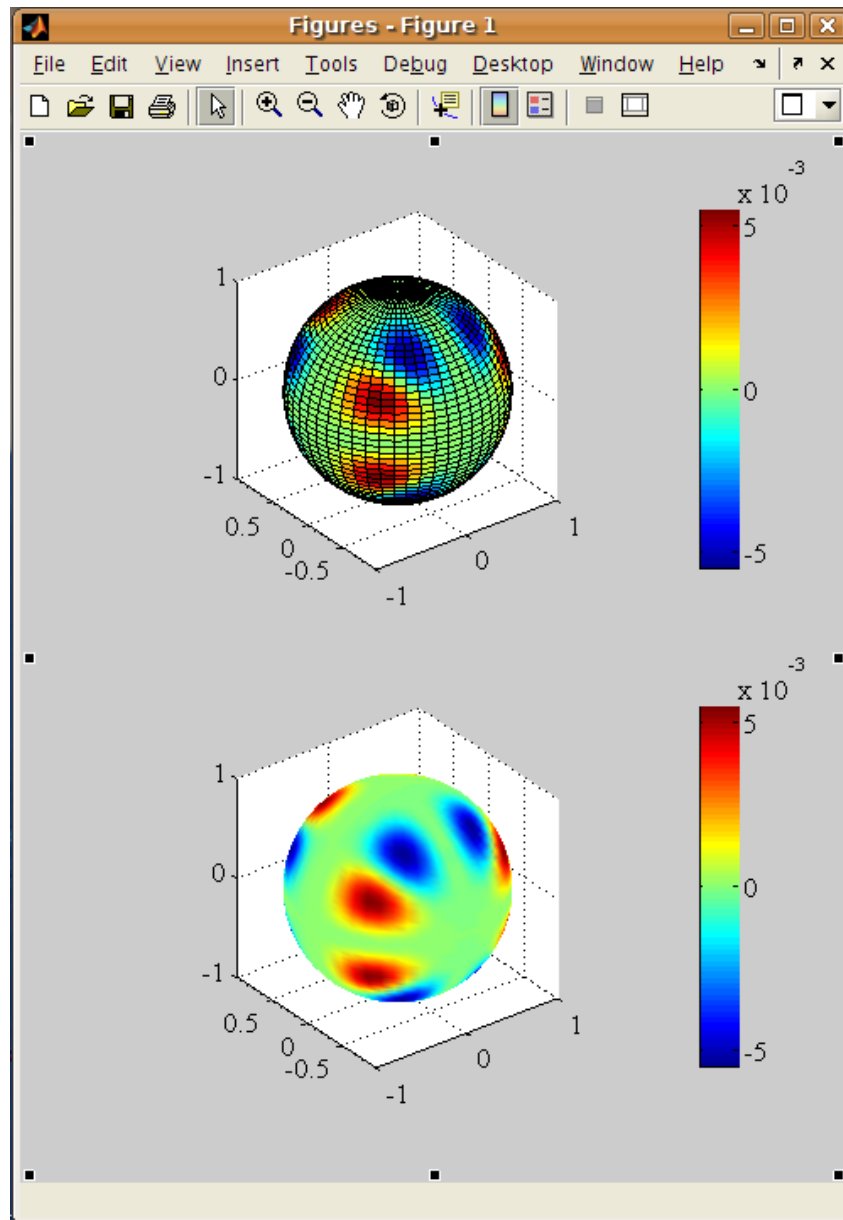


Figura 1.9: Funciones sobre superficies

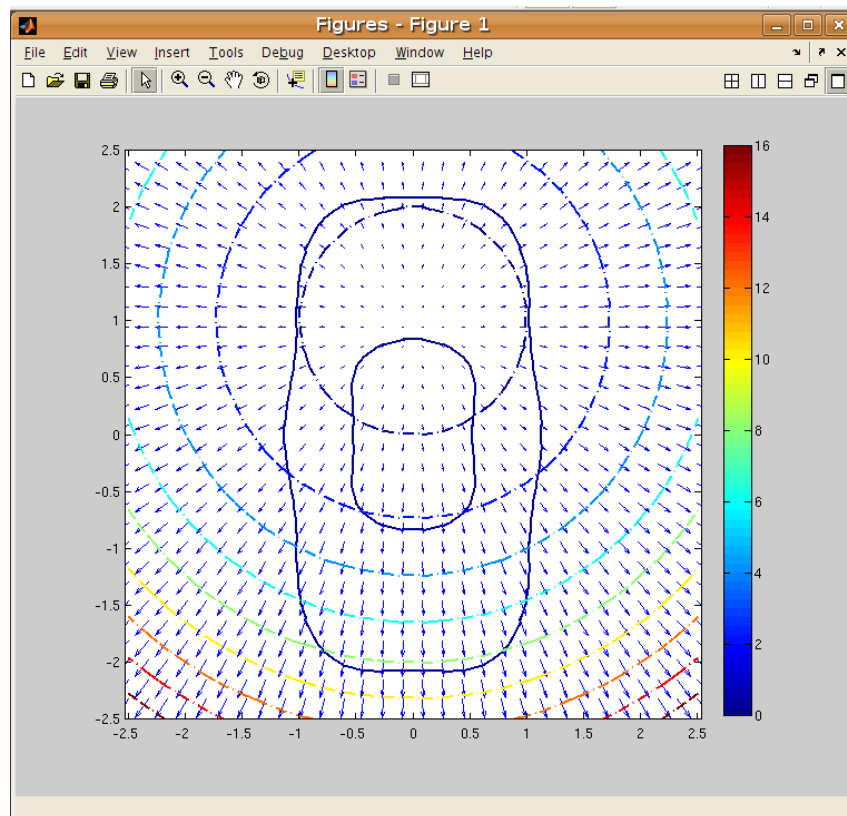


Figura 1.10: Situación geométrica correspondiente al problema de extremos condicionados (método de Lagrange)

# Ejercicios

## 1.1 Dibuja las siguientes funciones de dos variables

(a)  $x^2 + y^2$

(e)  $\sqrt{x^2 + y^2}$

(b)  $x^2 - y^2$

(f)  $\sqrt{x^2 + y^2 + 1}$

(c)  $\sqrt{16 - x^2 - y^2}$

(g)  $\sqrt{x^2 + y^2 - 1}$

(d)  $16 - x^2 - y^2$

Muestra también sus curvas de nivel  
(Ayuda: utiliza `real` en (c) y en (g))

## 1.2 Dibuja las gráficas de las siguientes funciones

(a)  $\frac{xy^2}{x^4 + y^2}$

(e)  $\exp(-x^2 - y^2) * (x + y)$

(b)  $\cos(x^2 - y^2)$

(f)  $x \cos(y) - y \cos(x)$

(c)  $\frac{\text{sen}(x + y)}{x^2 + y^2}$

(g)  $\log(1 + x^2) \cos(y)$

(d)  $\frac{\text{sen}(x^2 + y^2)}{|x| + |y|}$

(h)  $\frac{\sqrt{\text{sen}(\sqrt{|y + x^2|})}}{1 + \arctan(y^2)}$

1.3 Lee la ayuda de `contour` o `contourf` (`help`, `helpwin` o `doc`) y especifica de forma manual el número de líneas de nivel y los valores para las líneas de contorno en el problema anterior.

`help`  
`helpwin`  
`doc`

## 1.4 Dibuja las siguientes curvas en 2D

(a)  $(\cos(2\pi t), \text{sen}(2\pi t))$       (e)  $(\cos(t)(4 - \text{sen}(3t)), \text{sen}(t)(4 - \text{sen}(3t)))$

(b)  $(t \cos(t), t \text{sen}(t))$       (f)  $(\cos(t)(4 - \text{sen}(7t)), \text{sen}(t)(4 - \text{sen}(7t)))$

(c)  $(t - \text{sen}(t), 1 - \cos(t))$       (g)  $\left( (k+1) \left( \cos t - \frac{\cos((k+1)t)}{k+1} \right), \right.$   
 $\left. (k+1) \left( \text{sen } t - \frac{\text{sen}((k+1)t)}{k+1} \right) \right)$

para diferentes valores de  $k \geq 0$

(d)  $(2 \cos(t), \text{sen}(t))$       (f)  $(\text{senh}(t), \text{cosh}(t)).$

Escoge el intervalo de la forma que consideres oportuna. Utilizando el comando `comet`, obtened una animación del movimiento de la curva.

(Ayuda: Recuerda que el comando `hold on` activa la superposición en pantalla de forma que nuevos dibujos se superponen con los ya existentes en lugar de reemplazarlos. Para desconectar esta opción, utilizad `hold off`.)

### 1.5 Señala las diferencias que encuentres entre estas curvas

(a)  $(\cos(2\pi t), \text{sen}(2\pi t)), t \in [0, 1]$       (e)  $(\text{sen}(2\pi t), \cos(2\pi t)), t \in [0, 1]$

(b)  $(\cos(2\pi t^4), \text{sen}(2\pi t^4)), t \in [0, 1]$       (f)  $(\cos(t), -\text{sen}(t)), t \in [0, 2\pi]$

(c)  $(\cos(t), \text{sen}(t)), t \in [0, 6\pi]$       (d)  $(\cos((20t^3 - 30t^2 + 12t)\pi),$   
 $\text{sen}((20t^3 - 30t^2 + 12t)\pi)), t \in [0, 1]$

El comando `comet` te puede ser útil para este fin.

### 1.6 Haced lo mismo con estas curvas en 3D

(a)  $(\cos(2\pi t), \text{sen}(2\pi t), t), t \in [0, 4]$

(b)  $(t \cos(2\pi t), t \text{sen}(2\pi t), t), t \in [0, 4]$

(c)  $(4 \cos^2(t), 4 \cos(t) \text{sen}(t), 4\sqrt{(1 - \cos(t)) \cos(t)}), t \in [0, 4]$

(d)  $(4 \text{sen}(3t), 2 \text{sen}(7t), 4 \text{sen}(2t)), t \in [0, 2\pi]$

(e)  $(\cos(2\pi t) \text{sen}(\pi t), \text{sen}(2\pi t) \text{sen}(\pi t), \cos(\pi t)), t \in [0, 1]$

(f)  $(2 \cos(t), 2 \tan(t/2), 2 \sin(t)), t \in [-3.1, 3.1]$

(g)  $\left( \frac{3}{4} \cos(t) + \frac{1}{4} \cos(3t), \frac{3}{4} \text{sen}(t) - \frac{1}{4} \text{sen}(3t), \frac{2\sqrt{3}}{4} \text{sen}(2t) \right), t \in [0, 2\pi]$

Para los tres últimos ejemplos, es interesante que superpongas el dibujo con el de la esfera unidad. Para ello, simplemente ejecuta, antes de dibujar la curva, los siguientes comandos

sphere

```
clf % borra los dibujos
sphere(40) % dibuja una esfera
axis equal % fija misma proporcion en todos los ejes
hold on % el siguiente dibuja, se superpondrá a la esfera
```

Si quieres puedes utilizar comet3 para realizar una animación.

Muestra también sus curvas de nivel  
(Ayuda En Matlab,  $|x|$  se escribe abs(x))

**1.7 [Superficies en paramétricas]** Dibuja las siguientes superficies dadas en paramétricas

- (a)  $(u \cos(v), u \sin(v), u), (u, v) \in [-2, 2] \times [0, 2\pi]$
- (b)  $(u^2 \cos(v), u^2 \sin(v), u), (u, v) \in [-2, 2] \times [0, 2\pi]$
- (c)  $(\sin(v) \cosh(u), \cos(v) \cosh(u), \sinh(u)), (u, v) \in [-2, 2] \times [0, 2\pi]$
- (d)  $(\sin(v) \cos(u), \cos(v) \cos(u), \sin(u)), (u, v) \in [0, \pi] \times [0, 2\pi]$
- (e)  $(\cos(v)(2 + \cos(u)), \sin(v)(2 + \cos(u)), \sin(u)), (u, v) \in [0, \pi] \times [0, 2\pi]$
- (f)  $(\cos^3(u) \cos^3(v), \cos^3(u) \cos^3(v), \sin^3(v)), (u, v) \in [0, 2\pi] \times [0, 2\pi]$
- (g)  $\left( \cos(u)(3 + \sin(v) \cos(u/2) - \sin(2v) \sin(u)/2), \right.$   
 $\left. \sin(u)(3 + \sin(v) \cos(u/2) - \sin(2v) \sin(u/2)/2), \right.$   
 $\left. \sin(u/2) \sin(v) + \cos(u/2) \sin(2v)/2 \right), (u, v) \in [-\pi, \pi] \times [-\pi, \pi]$
- (h)  $\rho(u, v)(\cos(u) \sin(v), \sin(u) \sin(v), \cos(v))$

donde  $\rho(u, v) = \cos(mu) \sin(nv), (u, v) \in [0, \pi] \times [0, 2\pi]$  con  $m, n \in \mathbb{N} \cup \{0\}$ .

En algunas de estas superficies fijar la misma proporción en los ejes (axis equal) puede mejorar su aspecto.

**1.8** Dibuja sobre las gráficas de las funciones las funciones especificadas.

- (a) Sobre la superficie de la gráfica  $f(x, y) = xy^2 + 1$ , la función  $\cos(4\sqrt{x^2 + y^2 + z^2})$
- (b)  $\cos(\pi(x + y))$ , sobre la gráfica de la función  $xy \exp(-x^2 - y^2)$
- (c)  $\cos(\pi(x + y))$ , sobre la gráfica de la función  $xy \exp(-x^2 - y^2)$
- (b)  $(1 - z^2)^4(z^3 + 7z) \cos(4 \arctan(y/x))$ , sobre la esfera unidad

**1.9** Dibuja los siguientes campos vectoriales.

- (a)  $\left( -\frac{x}{(x^2 + y^2)^{3/2}}, -\frac{y}{(x^2 + y^2)^{3/2}} \right)$
- (b)  $\left( -\frac{y}{x^2 + y^2}, \frac{x}{x^2 + y^2} \right)$
- (c)  $(\cos(x^3 - y) - 2 \sin(y), \cos(4y)x)$
- (d)  $\left( -\frac{x}{(x^2 + y^2 + z^2)^{3/2}}, -\frac{y}{(x^2 + y^2 + z^2)^{3/2}}, -\frac{z}{(x^2 + y^2 + z^2)^{3/2}} \right)$
- (e)  $\left( -\frac{y}{(x^2 + y^2)}, -\frac{x}{(x^2 + y^2)}, 1 \right)$

**1.10** Proceded igual que en la sección 1.4 y mostrad la situación creada por los siguientes problemas de cálculo de extremos condicionados

- (a) Extremos de  $x^3/3 + x^2 + xy + y^2 - 4y^4/4$  condicionados a  $x^2 + y^2(x^8 + y^2) - 4$
- (b) Extremos de  $x^2y - y^3$  condicionados a  $(x^3 - y^3)^2 - 1$
- (c) Extremos de  $y \exp(-x^2) + y^2$  condicionados a  $\exp(x^2 + y)(x^4 + y^6) - 4x^2 + 1$
- (d) Extremos de  $x + yx^2 + \cos(xy)$  condicionados a  $\arccos(xy/(x^4 + y^2)) - 1$