PRÁCTICA 5. REPRESENTACIONES GRÁFICAS EN 2D Y 3D. CÁLCULO DIFERENCIAL EN VARIAS VARIABLES.

1 INTRODUCCIÓN.

El alumno debe ir realizando todos los EJEMPLOS que aparecen en esta práctica e ir anotando los resultados en la hoja que se le ha entregado. Además de estos ejemplos, hay planteados unos EJERCICIOS, que, a diferencia de lo que ocurre con los EJEMPLOS, no vienen desarrollados en el lenguaje Maxima, sino que tendrá que ser el alumno el que tenga que introducirlos para obtener el resultado.

2 GRÁFICOS EN EL PLANO CON plot2d.

2.1 Coordenadas cartesianas

El comando que se utiliza para representar la gráfica de una función de una variable real es plot2d que actúa, como mínimo, con dos parámetros: la función (o lista de funciones a representar), y el intervalo de valores para la variable x. Al comando plot2d se puede acceder también a través del menú Gráficos->Gráficos 2D

```
--> /*EJEMPLO 1*/
    plot2d(sin(2*x),[x,-2*%pi,2*%pi]);
```

```
--> /*EJEMPLO 2*/
    plot2d([x^2,sqrt(2*x)],[x,-2,2]);
```

Cuando pulsamos en el menú Gráficos->Gráficos 2D , aparece una ventana de diálogo con varios campos que podemos completar o modificar:
a) Expresión(es). La función o funciones que queramos dibujar. Por defecto, wxMaxima rellena este espacio con % para referirse a la salida anterior.
b) Variable x. Aquí establecemos el intervalo de la variable x donde queramos representar la función.
c) Variable y. Ídem para acotar el recorrido de los valores de la imagen.
d) Graduaciones. Nos permite regular el número de puntos en los que el programa evalúa una función para su representación en polares. Veremos ejemplos en la sección siguiente.

e) Formato. Maxima realiza por defecto la gráfica con un programa auxiliar. Si seleccionamos "en línea", dicho programa auxiliar es wxMaxima y obtendremos la gráfica en una ventana alineada con la salida correspondiente. Hay dos opciones más y ambas abren una ventana externa para dibujar la gráfica requerida: "gnuplot" es la opción por defecto que utiliza el programa Gnuplot para realizar la representación; también está disponible la opción "openmath" que utiliza el programa XMaxima. Prueba las diferentes opciones y decide cuál te gusta más.
f) Opciones. Aquí podemos seleccionar algunas opciones para que, por ejemplo, dibuje los ejes de coordenadas ("set zeroaxis;"); dibuje los ejes de coordenadas, de forma que cada unidad en el eje Y sea igual que el eje X ("set size ratio 1; set zeroaxis;"); dibuje una cuadrícula ("set grid;") o dibuje una gráfica en coordenadas polares ("set polar; set zeroaxis;"). Esta última opción la comentamos más adelante.
g) Gráfico al archivo. Guarda el gráfico en un archivo con formato Postscript.
Evidentemente, estas no son todas las posibles opciones. La cantidad de posibilidades que tiene Gnuplot es inmensa.
Observación. El prefijo “wx” añadido a plot2d o a cualquiera del resto de las órdenes que veremos en esta práctica (plot3d, draw2d, draw3d) hace que wxMaxima pase automáticamente a mostrar los gráficos en la misma ventana y no en una ventana separada. Es lo mismo que seleccionar en línea. Por ejemplo,

```
--> /*EJEMPLO 3*/
    wxplot2d(sin(2*x),[x,-2*%pi,2*%pi]);
```

```
--> /*EJEMPLO 4*/
    plot2d(x/(x^2-4),[x,-6,6],[y,-6,6],
    [gnuplot_preamble, "set zeroaxis;"])$
```

```
--> /*EJEMPLO 5*/
    plot2d(x/(x^2-4),[x,-6,6],[y,-6,6],
    [gnuplot_preamble, "set size ratio 1; set zeroaxis;"])$
```

```
--> /*EJEMPLO 6*/
    plot2d(x/(x^2-4),[x,-6,6],[y,-6,6],
    [gnuplot_preamble, "set grid;"])$
```

2.2 Coordenadas polares

Al representar una curva en coordenadas polares estamos escribiendo la longitud del vector como una función que depende del ángulo. En otras palabras, para cada ángulo fijo decimos cuál es el módulo del vector. El ejemplo más sencillo de función que a cualquiera se nos viene a la cabeza son las funciones constantes: La función f :[0,2pi]->R, f()=1 tiene como imagen aquellos vectores que tienen módulo 1 y argumento entre 0 y 2pi. Para ello, tenemos que seleccionar
"set polar; set zeroaxis;" en el campo Opciones de Gráficos 2D:

```
--> /*EJEMPLO 7*/
    plot2d([1], [ph,0,2*%pi],
    [plot_format, gnuplot],
    [gnuplot_preamble, "set polar; set zeroaxis;"],[x,-1,1])$
```

y si queremos que la gráfica sea proporcionada, hacemos

```
--> /*EJEMPLO 8*/
    plot2d([1], [ph,0,2*%pi],
    [plot_format, gnuplot],
    [gnuplot_preamble, "set polar;set size ratio 1; set zeroaxis;"],
    [x,-1,1])$
```

Otra gráfica viene dada por

```
--> /*EJEMPLO 9*/
    plot2d(ph,[ph,0,4*%pi],
    [gnuplot_preamble, "set polar; set zeroaxis;"],[x,-15,15])$
```

Observamos que la hélice resultante no es nada “suave”. Para conseguir el efecto visual de una línea curva como es esta hélice, añadimos el parámetro nticks. Por defecto, para dibujar una gráfica en paramétricas el programa evalúa en 10 puntos. Para aumentar este número de puntos, aumentamos dicho parámetro, por ejemplo nticks=30, o bien, podemos regularlo desde el campo Graduaciones dentro de de Gráficos 2D.

```
--> /*EJEMPLO 10*/
    plot2d(ph,[ph,0,4*%pi],
    [gnuplot_preamble, "set polar; set zeroaxis;"], [nticks,30],
    [x,-15,15])$
```

Si representamos la función
r(theta)=exp(cos(theta))-2cos(4theta)+sin(theta/12)^5
obtenemos algo parecido a una mariposa:

```
--> /*EJEMPLO 11*/
    r(ph):=exp(cos(ph))-2*cos(4*ph)+sin(ph/12)^5$
    plot2d(r(ph),[ph,0,2*%pi],
    [gnuplot_preamble, "set polar; set zeroaxis;"],[x,-5,5])$
```

2.3 Coordenadas paramétricas

El programa wxMaxima nos permite también representar curvas en forma paramétrica, es decir, curvas definidas como (x(t),y(t)), donde el parámetro t varía en un determinado intervalo compacto [a,b].
Para ello, dentro del comando plot2d añadimos "parametric" de la forma siguiente: plot2d([parametric,x(t),y(t),[t,a,b]])
Para acceder a esta opción de la función plot2d podemos hacerlo a través del botón Especial que aparece en la parte superior derecha de la ventana de diálogo Gráficos 2D.
Para terminar, aquí tienes algunas curvas planas interesantes:

Astroide: Es la curva trazada por un punto fijo de un círculo de radio r que rueda sin deslizar dentro de otro círculo fijo de radio 4r.
Sus ecuaciones paramétricas son:

```
--> /*EJEMPLO 12*/
    plot2d([parametric,cos(t)^3,sin(t)^3,[t,0,2*%pi],[nticks,50]])
```

Cardioide: Es la curva trazada por un punto fijo de un círculo de radio r que rueda sin deslizar alrededor de otro círculo fijo del mismo radio.
Sus ecuaciones paramétricas
son:

```
--> /*EJEMPLO 13*/
    plot2d([parametric,(1+cos(t))*cos(t),(1+cos(t))*sin(t),
    [t,0,2*%pi],[nticks,50]])
```

Lemniscata de Bernoulli:Es el lugar geométrico de los puntos P del plano, cuyo producto de distancias a dos puntos fijos F1 y F2, llamados focos, verifica la igualdad abs(P-F1)*abs(P-F2)=1/4*abs(F1-F2)^2
En coordenadas cartesianas esta curva viene dada por la ecuación (x^2+y^2)^2=x^2-y^2. Aquí tienes sus ecuaciones paramétricas:

```
--> /*EJEMPLO 14*/
    plot2d([parametric, (cos(t))/(1+sin(t)^2),
    (cos(t)*sin(t))/(1+sin(t)^2), [t,0,2*%pi],[nticks,70]])
```

Espiral equiangular: También llamada espiral logarítmica. Es aquella espiral en la que el radio vector corta a la curva en un ángulo constante alfa. Sus ecuaciones paramétricas son:

```
--> /*EJEMPLO 15*/
    %alpha:%pi/2-0.2$
    plot2d([parametric,exp(t*cot(%alpha))*cos(t),
    exp(t*cot(%alpha))*sin(t),[t,0,4*%pi],[nticks,90]])
```

Cicloide: También conocida como tautocrona o braquistocrona. Es la curva que describiría un punto de una circunferencia que avanza girando sin deslizar. Sus ecuaciones paramétricas son:

```
--> /*EJEMPLO 16*/
    plot2d([parametric,t-sin(t),1-cos(t),[t,0,6*%pi],[nticks,90]])
```

3 GRÁFICOS EN 3D.

Con Maxima se pueden representar funciones de dos variables de forma similar a como hemos representado funciones de una. La principal diferencia es que vamos a utilizar el comando plot3d en lugar de plot2d, pero igual que en el caso anterior, son obligatorios la función o funciones y el dominio que tiene que ser de la forma [a,b]x[c,d]:
plot3d(f(x,y),[x,a,b],[y,c,d]) gráfica de f(x,y) en [a,b]x[c,d]

```
--> /*EJEMPLO 17*/
    plot3d(cos(x*y),[x,-3,3],[y,-3,3]);
```

La gráfica que acabas de obtener se puede girar sin más que pinchar con el ratón sobre ella y deslizar el cursor.
plot3d también permite trazar la gráfica de una superficie definida en forma paramétrica; sin embargo, no permite gráficas de curvas en R3: Para ello usaremos
plot3d([x(u,v),y(u,v),z(u,v)],[u,umin,umax],[v,vmin,vmax])
Así, si queremos representar la esfera unitaria definida paramétricamente mediante (u,v)->(sen u sen v,cos u sen v,cos v), con u entre 0 y 2pi, y v entre -pi/2 y pi/2:

```
--> /*EJEMPLO 18*/
    plot3d([cos(u)*cos(v),sin(u)*cos(v),sin(v)],
    [u,0,2* %pi],[v,- %pi/2, %pi/2]);
```

Al comando plot3d se puede acceder a través del menú Gráficos->Gráficos 3D. Después de esto aparece una ventana con varios campos para rellenar:

a) Expresión. La función o funciones que vayamos a dibujar.
b) Variable. Hay dos campos para indicar el dominio de las dos variables.
c) Cuadrícula. Indica cuántas valores se toman de cada variable para representar la función. Cuanto mayor sea, más suave será la representación a costa de aumentar la cantidad de cálculos.
d) Formato. Igual que en plot2d, permite escoger qué programa se utiliza para representar la función. Se puede girar la gráfica en todos ellos salvo si escoges “en línea”.
e) Opciones. Las comentamos a continuación.
f) Gráfico al archivo. Permite elegir un fichero donde se guardará la gráfica.
Quizá la mejor manera de ver el efecto de las opciones es repetir el dibujo anterior. La primera de ellas es "set pm3d at b" que dibuja la superficie usando una malla y en la base añade curvas de nivel (como si estuviéramos mirando la gráfica desde arriba):

```
--> /*EJEMPLO 19*/
    plot3d(cos(x*y), [x,-5,5], [y,-5,5], [plot_format,gnuplot],
    [gnuplot_preamble, "set pm3d at b"])$
```

La segunda hace varias cosas, "set pm3d at s" nos dibuja la superficie coloreada, "unset surf" elimina la malla y "unset colorbox" elimina la barra que teníamos en la derecha con la explicación de los colores y su relación con la altura (el valor de la función):

```
--> /*EJEMPLO 20*/
    plot3d(cos(x*y), [x,-5,5], [y,-5,5], [plot_format,gnuplot],
    [gnuplot_preamble, "set pm3d at s; unset surf;
    unset colorbox"])$
```

La tercera, "set pm3d map", nos dibuja un mapa de curvas de nivel (gráfico de densidad) con alguna ayuda adicional dada por el color: Así, si queremos representar solamente el gráfico de densidad para esta superficie, haremos:

```
--> /*EJEMPLO 21*/
    plot3d(cos(x*y), [x,-5,5], [y,-5,5],
    [gnuplot_preamble,"set pm3d map"]);
```

La cuarta, "set hidden3d", sólo muestra la parte de la superficie que sería visible desde nuestro punto de vista. En otras palabras, hace la superficie sólida y no transparente:

```
--> /*EJEMPLO 22*/
    plot3d(cos(x*y), [x,-5,5], [y,-5,5], [plot_format,gnuplot],
    [gnuplot_preamble, "set hidden3d"])$
```

En el dibujo anterior (en el papel) es posible que no aprecie bien. A simple vista parece el mismo dibujo que teníamos dos salidas antes. Observa bien: hay una pequeña diferencia. El uso de pm3d hace que se coloree el dibujo, pero cuando decimos que no se muestra la parte no visible de la figura nos estamos refiriendo a la malla. Quizá es mejor dibujar la malla y el manto de colores por separado para que se vea la diferencia. Esta opción no viene disponible por defecto en wxMaxima. Ten en cuenta que las opciones que tiene Gnuplot son casi infinitas y sólo estamos comentando algunas.

```
--> /*EJEMPLO 23*/
    plot3d(x^2-y^2, [x,-5,5], [y,-5,5], [plot_format,gnuplot],
    [gnuplot_preamble, "set pm3d at b; set hidden3d"])$
```

```
--> /*EJEMPLO 24*/
    plot3d(x^2-y^2, [x,-5,5], [y,-5,5], [plot_format,gnuplot],
    [gnuplot_preamble, "set pm3d at b"])$
```

La quinta y la sexta opciones nos permiten dibujar en coordenadas esféricas o cilíndricas. Ya veremos ejemplos más adelante.

Si queremos representar solamente un gráfico de contorno (curvas de nivel), es posible mediante contour_plot, como vemos a continuación:

```
--> /*EJEMPLO 25*/
    contour_plot(cos(x*y), [x,-5,5], [y,-5,5])$
```

4 GRÁFICOS CON draw.

El paquete draw se distribuye conjuntamente con Maxima y constituye una interfaz que comunica de manera muy eficiente Maxima con Gnuplot. Este paquete incorpora una considerable variedad de funciones y opciones que permiten obtener la representación de un amplio número de objetos gráficos bidimensionales y tridimensionales. En este caso nos vamos a centrar solamente en GRAFICOS TRIDIMENSIONALES, ya que con draw vamos a poder realizar gráficas que no podemos realizar con plot3g, como por ejemplo, representar curvas en R3. No obstante, invitamos al lector a que profundice en este paquete (a menudo mucho más sencillo e intuitivo que plot) para ver otras opciones. Toda la información al respecto se puede ver fácilmente en el manual que se encuentra en: http://www.ugr.es/~dpto_am/docencia/Apuntes/Practicas_con_Maxima.pdf
Para poder utilizar el paquete draw es preciso cargarlo en la memoria, y para ello se utiliza la función load(draw).
Las funciones draw, draw2d y draw3d devuelven las salidas gráficas en una ventana de Gnuplot, aparte de la ventana de trabajo actual. No obstante el entorno gráfico wxMaxima permite utilizar las funciones wxdraw, wxdraw2d y wxdraw3d que sí devuelven las salidas en el cuaderno de trabajo actual. Debe tenerse presente que al utilizar la función wxdraw3d, el punto de vista de la gráfica obtenida no puede ser cambiado en tiempo real.
En esta práctica se van a mostrar los resultados mediante las función draw3d, pero todos los ejemplos mostrados pueden ser ejecutados, sin ningún problema, con la función wxdraw3d (lo mismo ocurre con draw y draw2d, como puede verse en el manual anteriormente citado).

Principales objetos gráficos tridimensionales incorporados en draw:
- points([x1,x2,...];[y1,y2,,,,],[z1,z2,...]) representa puntos en R3
- vector([p1,p2,p3],[v1,v2,v3]) nos da el vector (v1,v2,v3) con punto de aplicación (p1,p2,p3)
- explicit(f(x,y),x,xmin,xmax,y,ymin,ymax) representa a una función explícita f en el dominio dado.
- implicit(E(x,y,z),x,xmin,xmax,y,ymin,ymax,z,zmin,zmax) representa a una ecuación implícita E en el dominio dado
- parametric(fx,fy,fz,t,tmin,tmax) curva paramétrica tridimensional, cuyo parámetro t satisface tmin<t<tmax
- cylindrical(r(z,theta),z,zmin,zmax,theta,thetamin,thetamax), nos da una función cilíndrica r donde los parámetros varían en los intervalos que se indican
- spherical(r(fi;theta),fi,fimin,fimax,theta,thetamin,thetamax) nos da una función esférica r donde los parámetros varían en los intervalos que se indican
Veamos ejemplos de todos éstos comandos (y otros):

Aquí se muestra la grá□ca del vector cuyo punto de aplicación es el origen y cuya parte vectorial es (1,1,1): (Primero tenemos que llamar al paquete draw)

```
--> load("draw")$
```

```
--> /*EJEMPLO 26*/
    draw3d(vector([0,0,0],[1,1,1]));
```

Esto muestra la gráfica de la función f(x,y)=sen x + sen(xy), en [0,2pi]x[0,2pi]:

```
--> /*EJEMPLO 27*/
    draw3d(explicit(sin(x)+sin(x*y),x,0,2*%pi,y,0,2*%pi));
```

Aquí se muestra la gráfica de una superficie generada a partir de una ecuación implícita (se indica la ecuación y donde varían las 3 variables x, y, z):

```
--> /*EJEMPLO 28*/
    draw3d(implicit((sqrt(x^2+y^2)-4)^2+z^2=4,x,-6,6,
    y,-6,6,z,-2,2) );
```

Esta es la gráfica de la curva dada por la parametrización t->(cos t,sen t,t/8), con 0<t<4pi (estas curvas en R3 no podemos representarlas mediante plot3d):

```
--> /*EJEMPLO 29*/
    draw3d(parametric(cos(t),sin(t),t/8,t,0,4*%pi));
```

Aquí se muestra la gráfica de la superficie definida por la parametrización (u,v)->((2+cos v)cos u,(2+cos v)sin u,sin v), con 0<u<2pi y 0<v<2pi:

```
--> /*EJEMPLO 30*/
    draw3d(parametric_surface((2+cos(v))*cos(u),
    (2+cos(v))*sin(u),sin(v),u,0,2*%pi,v,0,2*%pi));
```

Esta es la gráfica de la superficie definida en coordenadas cilíndricas mediante (z,t)->cos z, con -2<z<2 y 0<t<2pi:

```
--> /*EJEMPLO 31*/
    draw3d(cylindrical(cos(z),z,-2,2,t,0,2*%pi));
```

He aquí la gráfica de la superficie definida en coordenadas esféricas mediante (a,t)->1, con 0<a<2pi y 0<t<pi:

```
--> /*EJEMPLO 32*/
    draw3d(spherical(1,a,0,2*%pi,t,0,%pi));
```

A continuación incluimos otros ejemplos donde aparecen determinadas opciones que podemos realizar:

Esto define una curva, algunos puntos sobre ésta y algunos vectores tangentes a la misma:

```
--> /*EJEMPLO 33*/
    a(t):=[cos(t),sin(t),t/8]$
    define("a'"(t),diff(a(t),t))$
    t0:create_list(i*%pi/4,i,0,16)$
    T:create_list( vector(a(i),"a'"(i)),i,t0)$
    P:map(a,t0)$
    objetos:([nticks=100,color=red,
    apply(parametric,append(a(t),[t,0,4*%pi]))
    color=blue,unit_vectors=true,T,
    color=green,point_type=7,points(P),
    user_preamble="set size ratio 1",
    title="Campo vectorial tangente",
    xyplane=0,axis_3d=false,
    xtics=false,ytics=false,ztics=false,
    xaxis=true,yaxis=true,zaxis=true,
    xlabel="x",ylabel="y",zlabel="z"])$
    draw3d(objetos);
```

La superficie implícita definida anteriormente pero con otras opciones:

```
--> /*EJEMPLO 34*/
    draw3d(x_voxel=17,y_voxel=17,z_voxel=15,
    palette=[12,5,27],enhanced3d=true,
    implicit((sqrt(x^2+y^2)-4)^2+z^2=4,x,-6,6,
    y,-6,6,z,-2,2));
```

5 Cálculo diferencial en varias variables.

5.1 Derivadas parciales de funciones de varias variables

Vamos a ver como calcular las derivadas parciales de funciones de varias variables, tanto de primer orden como de orden superior.

```
--> /*EJEMPLO 35: Para calcular la primera derivada parcial respecto
    de x, de f(x,y,z)=x+sin(x*y)+log(x*y*z), y la llamamos fx*/

    f(x,y,z):=x+sin(x*y)+log(x*y*z)$
    fx:diff(f(x,y,z),x);
```

```
--> /*EJEMPLO 36: Y lo mismo para las derivadas parciales respecto
    de z e y*/

    fz:diff(f(x,y,z),z);
    fy:diff(f(x,y,z),y);
```

```
--> /*EJEMPLO 37: Podemos evaluar fx en un punto
    (%pi/2,y=1/2,z=1)*/

    fx,x=%pi/2,y=1/2,z=1;
```

```
--> /*EJEMPLO 38: Podemos calcular derivadas sucesivas, como por
    ejemplo, fxx, fxy o fxzy*/

    fxx:diff(f(x,y,z),x,2);
    fxy:diff(f(x,y,z),x,1,y,1);
    fxzy:diff(f(x,y,z),x,1,z,1,y,1);
```

```
--> /*EJEMPLO 39: Comprobamos que la función z(x,t)=sen(x-c*t),
    con c constante, satisface la ecuación de ondas
    ftt=c^2 fxx*/

    z(x,t):=sin(x-c*t)$
    is(ratsimp(diff(z(x,t),t,2)=c^2*diff(z(x,t),x,2)));
```

EJERCICIO 1.
a) Sea f(x,y)=5x^2-2xy+3y^3 x^2. Calcular las siguientes derivadas: fxy, fyx, fxx(3,4).

b) Probar que la función f(x,y)=e^x*sin(y) es armónica, es decir, verifica la ecuación fxx+fyy=0.

c) Calcular la pendiente de la recta tangente a la gráfica de f(x,y)=x*y^3+x^3*y en P(1,-1,-2) en la dirección del eje x y en la del eje y.

5.2 Extremos relativos de funciones de dos variables

Vamos a ver como podemos calcular los puntos críticos de una función con dos variables, calcular su hessiano, y evaluar éste en los puntos críticos.

```
--> /*EJEMPLO 40: Vamos a calcular los puntos críticos de la función
    f(x,y)=x*y*exp(-x^2-y^2)*/

    f(x,y):=x*y*%e^(-x^2-y^2)$
    puntoscriticos:solve([diff(f(x,y),x),diff(f(x,y),y)],[x,y]);
```

Ahora definimos el hessiano de una función f(x,y) con Maxima y lo llamamos Hessiano:

```
--> /*EJEMPLO 41: Definicion general de Hessiano*/

    Hessiano(x,y):=(diff(f(x,y),x,2)*diff(f(x,y),y,2))-
    (diff(f(x,y),x,1,y,1)*diff(f(x,y),x,1,y,1));
```

```
--> /*EJEMPLO 42: Vamos a obtener el Hessiano para la función anterior
    (al que llamaremos hessianof) y vamos a evaluar éste en los puntos
    críticos obtenidos anteriormente*/

    Hessiano(x,y)$
    hessianof(x,y):=ev(Hessiano(x,y),diff)$
    hessianof(x,y);
```

```
--> /*EJEMPLO 42 (continuación)*: Vamos a ver que en los puntos
    críticos (3) y (4) de la lista puntoscriticos obtenida
    anteriormente la función presenta mínimos relativos*/

    is(at(hessianof(x,y),puntoscriticos[3])>0);
    is(at(hessianof(x,y),puntoscriticos[4])>0);
```

```
--> /*EJEMPLO 42 (continuación): Ahora evaluamos fxx en estos puntos
    críticos*/

    fxx(x,y):=diff(f(x,y),x,2)$
    fxx(x,y);
    is(at(fxx(x,y),puntoscriticos[3])>0);
    is(at(fxx(x,y),puntoscriticos[4])>0);
```

Por tanto la función tiene dos mínimos en los puntos críticos (3) y (4) obtenidos anteriormente.

```
--> /*EJEMPLO 42 (continuación): Ahora representaremos gráficamente
    la superficie y los puntos donde se alcanzan estos mínimos*/

    load("draw")$
    draw3d(color=green,explicit(f(x,y),x,-2,2,y,-2,2),color=black,
    point_size=2,point_type=filled_circle,
    points([[1/sqrt(2),-1/sqrt(2),
    f(1/sqrt(2),-1/sqrt(2))],[-1/sqrt(2),1/sqrt(2),
    f(-1/sqrt(2),1/sqrt(2))]]));
```

```
--> /*EJEMPLO 43: Ahora hacemos lo mismo con los puntos críticos
    (2) y (5) de la lista puntoscriticos anterior*/

    is(at(hessianof(x,y),puntoscriticos[2])>0);
    is(at(hessianof(x,y),puntoscriticos[5])>0);
    is(at(fxx(x,y),puntoscriticos[2])<0);
    is(at(fxx(x,y),puntoscriticos[5])<0);
```

```
--> /*EJEMPLO 43 (Continuación): Ahora representamos la superficie
    y los puntos máximos*/

    load("draw")$
    draw3d(color=green,explicit(f(x,y),x,-2,2,y,-2,2),color=black,
    point_size=2,point_type=filled_circle,
    points([[1/sqrt(2),1/sqrt(2),f(1/sqrt(2),1/sqrt(2))],
    [-1/sqrt(2),-1/sqrt(2),f(-1/sqrt(2),-1/sqrt(2))]]));
```

Por lo que f tendrán en dichos puntos máximos relativos.

Veamos por último que ocurre con el punto crítico (0,0):

```
--> /*EJEMPLO 44: Podemos ver que en (0,0) hay un punto de silla
    por el criterio de ser el hessiano negativo*/

    is(at(hessianof(x,y),puntoscriticos[1])<0);
```

5.3 Derivada de la función compuesta

Veremos ahora como aplicar la regla de la cadena para calcular derivadas de funciones compuestas de varias variables. Con la orden depends podemos declarar dependencias de funciones.
Veremos que en el caso de tener funciones compuestas, la orden diff aplica la regla de la cadena. Además calcularemos derivadas de funciones compuestas cuyas funciones componentes vienen definidas de forma explícita.

```
--> /*EJEMPLO 45: Sea la función de dos variables f(x,y), donde
    x e y son funciones de una variable independiente t: x=x(t)
    e y=y(t). Para calcular la derivada df/dt, declaramos las
    dependencias con la orden depends y derivamos con diff:*/

    depends(f,[x,y],[x,y],t);
    diff(f,t);
    remove(all,dependency)$
```

Notemos que en este último ejemplo, la orden diff, aplica la regla de la cadena para calcular la derivada:
df/dt = @f/@x dx/dt + @f/@y dy/dt
La orden remove(all,dependency) se usa para cancelar las dependencias de las variables.

```
--> /*EJEMPLO 46: Consideramos el caso de una función de dos
    variables x e y, f(x,y), que a su vez dependen de otras dos
    variables independientes s y t, x(s,t), y(s,t). Queremos calcular
    las derivadas parciales @f/@s y @f/@t:*/

    depends(f,[x,y],[x,y],[s,t]);
    diff(f,s);
    diff(f,t);
    remove(all,dependency)$
```

```
--> /*EJEMPLO 47: Sea f(x) tal que x=x(t). Calculamos la derivada
    segunda de f respecto de t:*/

    depends(f,x,x,t)$
    diff(f,t,2);
    remove(f,dependency)$
```

Suele ser habitual que la relación de dependencia nos venga dada en forma explícita, como veremos en los siguientes ejemplos:

```
--> /*EJEMPLO 48: Sea f(x,y)=xy+y^2, tal que x(t)=sen(t), y(t)=e^t.
    Calcularemos df/dt:*/

    f(x,y):=x*y+y^2$
    diff(f(sin(t),%e^t),t);
    kill(f)$
```

```
--> /*EJEMPLO 49: Sea f(x,y)=xy+x^2+y^2, donde x(t)=rcos(t),
    y(t)=rsen(t). Calculamos @f/@r y @f/@t:*/

    f(x,y):=x*y+x^2+y^2$
    diff(f(r*cos(t),r*sin(t)),r);
    diff(f(r*cos(t),r*sin(t)),t);
```

```
--> /*EJEMPLO 49 (cont.): Y si queremos evaluar @f/@t en el punto
    (pi,0):*/

    diff(f(r*cos(t),r*sin(t)),t)$
    %,r=%pi,t=0;
    kill(f)$
```

5.4 Derivación implícita

Para usar la derivación implícita, es más aconsejable derivar directamente en la ecuación que nos den, aunque previamente hemos de definir la dependencia entre las variables, como vemos en los ejemplos siguientes:

```
--> /*EJEMPLO 50: Sea la ecuación 2sin(x)cos(y)=1. Calculamos
    y'(x) por derivación implícita:*/

    depends(y,x)$
    ec:2*sin(x)*cos(y)=1;
    diff(ec,x);
    der:solve(%,'diff(y,x));
```

Notemos que hemos declarado la dependencia con depends, hemos definido la ecuación y la hemos llamado ec, y hemos derivado y despejado la derivada. La salida es una lista de un solo elemento que guardamos en una variable que llamamos der. Este valor es, del primer (único) elemento de esa lista, la parte que está a la derecha del igual. Podemos usar para extraer esa parte la orden second(exp) y lo guardamos en una variable que llamamos dydx.

```
--> /*EJEMPLO 50 (cont.):*/

    dydx:second(der[1]);
```

```
--> /*EJEMPLO 50 (cont.): Y si ahora calculamos la derivada de la
    función en el punto x=pi/4, y=pi/4, y'(pi/4)*/

    dydx,x=%pi/4,y=%pi/4;
    remove(all,dependency)$
    kill(all)$
```

```
--> /*EJEMPLO 51: Calculamos las derivadas parciales primeras de
    z(x,y)  tal que z está definida implícitamente por la ecuación
    x^2+y^2+z^2=25*/
    /*En primer lugar definimos la dependencia de z y a la ecuación
    la llamamos ec*/

    depends(z,[x,y])$
    ec:x^2+y^2+z^2=25$
```

```
--> /*EJEMPLO 51 (cont.): Para calcular dz/dx derivamos toda la
    ecuación con respecto a x y despejamos. Guardamos la derivada
    dz/dx en una variable llamada dzdx (para ello usamos la orden
    second como anteriormente)*/

    diff(ec,x);
    solve(%,diff(z,x));
    dzdx:second(%[1]);
```

```
--> /*EJEMPLO 51 (cont.): Algo similar hemos de hacer para calcular
    dz/dy. En este caso guardamos la derivada parcial en la
    variable dzdy:*/

    diff(ec,y)$
    solve(%,diff(z,y))$
    dzdy:second(%[1]);
```

```
--> /*EJEMPLO 51 (cont.): Para calcular la segunda derivada de z
    respecto a dos veces x, derivamos la ecuación dos veces con
    respecto a x*/

    diff(ec,x,2);
```

```
--> /*EJEMPLO 51 (cont.): Observamos que en la ecuación derivada
    tenemos tanto dz/dxx como dz/dx. Antes de despejar, tendremos
    que sustituir la derivada dz/dx que hemos calculado antes y que
    hemos guardado en dzdx. Para hacer esto, usamos la orden
    subst(exp1,exp2,exp), que sustituye en la expresión exp,
    la exp2 por exp1. Por último, despejamos dz/dxx:*/

    diff(ec,x,2);
    subst(dzdx,'diff(z,x),%);
    solve(%,diff(z,x,2));
```

```
--> /*EJEMPLO 51 (cont.): Para calcular dz/dyy:*/

    diff(ec,y,2);
    subst(dzdy,'diff(z,y),%);
    solve(%,diff(z,y,2));
```

```
--> /*EJEMPLO 51 (cont.): Calculamos dz/dxdy derivando la ecuación
    primero con respecto a y y luego con respecto a x:*/

    diff(ec,x,1,y,1);
    subst(dzdy,'diff(z,y),%);
    subst(dzdx,'diff(z,x),%);
    solve(%,diff(z,x,1,y,1));
```

EJERCICIO 2.
a) Calcular la rectan tangente a la curva x^2(x^2+y^2)=y^2 en el punto (1/(2^1/2),1/(2^1/2)).

b) Calcular las derivadas y', y'', y''', donde x^2+xy+y^2=3.

c) Calcula dx/dt y dy/dt si x+y+t=0, x^2+y^2+t^2 = 1.