Prácticas con wxMaxima: cálculo diferencial de funciones de varias variables

BENITO J. GONZÁLEZ RODRÍGUEZ (bjglez@ull.es)
DOMINGO HERNÁNDEZ ABREU (dhabreu@ull.es)
MATEO M. JIMÉNEZ PAIZ (mjimenez@ull.es)
M. ISABEL MARRERO RODRÍGUEZ (imarrero@ull.es)
ALEJANDRO SANABRIA GARCÍA (asgarcia@ull.es)

Departamento de Análisis Matemático
Universidad de La Laguna

Índice

PRÁCTICA 4: CÁLCULO DIFERENCIAL DE FUNCIONES DE VARIAS VARIABLES.

Duración estimada: 3 horas.

Objetivo: trabajar algunos conceptos relacionados con el cálculo diferencial de funciones de varias variable (derivación parcial, vector gradiente, matriz hessiana, puntos críticos, plano tangente, etc.).

1 Declaración y representación de funciones de varias variables.

Las funciones de varias se declaran de modo análogo al caso de funciones de una variable.

Para el caso de funciones de dos variables reales f(x,y), podemos representar gráficamente la superficie z=f(x,y) con las órdenes "plot3d" o "draw3d". Como veremos, este último comando permite obtener representación más elaboradas.

```
--> f(x,y):=sin(x-y);

--> f(%pi/2,0);
```

Si generamos una gráfica GNU podemos girarla automáticamente desde la ventana abierta.

```
--> plot3d(f(x,y),[x,-5,5],[y,-5,5]);

--> plot3d(f(x,y),[x,-5,5],[y,-5,5],[colorbox,true]);

--> plot3d(f(x,y),[x,-5,5],[y,-5,5],[grid, 100,100],[legend,false], [palette, false],[color, red]);

--> plot3d(f(x,y),[x,-5,5],[y,-5,5],[box,false],[colorbox,true]);
```

Con plot3d también podemos representar superficies definidas paramétricamente.

```
--> expr_1: cos(y)*(2+cos(x))$
    expr_2: sin(y)*(2+cos(x))$
    expr_3: -sin(x)$

    plot3d([expr_1, expr_2, expr_3], [x, 0, 2*%pi], [y, 0, 2*%pi], [grid, 40, 40]);
```

Podemos definir funciones de varias variables a partir de expresiones en varias variables usando la orden "ev" (evaluar).

```
--> f(x,y):=ev(exp(4-x^2-y),x=x,y=y);

--> f(0,0); %,numer;

--> f(x,y,z):=ev(1+x^3-y^2+z,x=x,y=y,z=z);

--> f(1,1,1);

--> plot3d(f(x,y,0),[x,-5,5],[y,-5,5]);

--> plot3d(f(0,y,z),[y,-5,5],[z,-5,5]);

--> f(x,y):=sin(3-x-y);

--> plot3d(f(x,y),[x,-3,3],[y,-3,3]);
```

Si sólo conocemos la ecuación implícita de la superficie F(x,y,z)=0 es más recomendable usar el paquete "draw" con una gráfica implícita.

```
--> f(x,y,z):=(x-1)^2-(y+1)^2+z^2-4;

--> load(draw);

--> grafica:implicit(f(x,y,z)=0,x,-6,6,y,-5,3,z,-5,5);

--> draw3d(grafica);
```

En el caso de funciones implícitas, podemos modificar la resolución usando los parámetros "x_voxel" e "y_voxel".

```
--> draw3d(x_voxel=25,y_voxel=25,grafica);

--> draw3d(enhanced3d=true,cbrange=[-5,8],grafica);

--> draw3d(enhanced3d=true,palette =gray,grafica);

--> grafica:implicit(x^2-y^2=z^2,x,-3,3,y,-3,3,z,-3,3);
    draw3d(color=green,x_voxel=15,y_voxel=15,grafica);
```

También podemos crear gráficas animadas en 3D con la sentencia "with_slider_draw3d".

En el caso de funciones explícitas o paramétricas, podemos modificar la resolución usando los parámetros "xu_grid" e "yv_grid".

```
--> with_slider_draw3d(k,makelist(i/4,i,-8,8),xu_grid=50,yv_grid=50,enhanced3d=true,zrange=[-1,1],contour=base,explicit(cos(x^2-
    k*y^2),x,-2,2,y,-2,2),surface_hide=true);
```

2 Derivadas parciales.

Dada una función (o expresión) de varias variables reales, podemos obtener sus derivadas parciales análogamente al caso unidimensional con la orden "diff" indicando las variables e índices de derivación:

diff(f(x1,...xn),x1,m1,x2,m2,...,xn,mn)

```
--> f(x,y):=x^4+x^2*y^2+y^4;

--> diff(f(x,y),x); diff(f(x,y),y) /* Derivadas primeras respecto de x e y */;

--> diff(f(x,y),x,x); diff(f(x,y),x,2) /* Observar el siguiente error al intentar calcular la derivada segunda respecto de x */;

--> diff(f(x,y),x,y); diff(f(x,y),x,1,y,1) /* Si se deriva respecto de dos o más variables, es necesario indicar los índices de derivación en cada
    variable */;

--> diff(diff(f(x,y),x),y);
```

Si deseamos evaluar una derivada parcial, podemos generar una función a partir de ellas con la orden "declare".

```
--> f(x,y):=sin(x-y);

--> f(1,0);

--> fx(x,y):=diff(f(x,y),x);

--> fx(1,0);

--> define(fx(x,y),diff(f(x,y),x));
```

```
--> fx(1,0);

--> plot3d([f(x,y),fx(x,y),[x,-%pi,%pi],[y,-%pi,%pi]]);

--> define(f(x,y,z),z*tan(x*y));

--> define(fx(x,y,z),diff(f(x,y,z),x)); define(fy(x,y,z),diff(f(x,y,z),y)); define(fz(x,y,z),diff(f(x,y,z),z));

--> fx(%pi,1,1); fy(%pi,1,1); fz(%pi,1,1);
```

Veamos algunos ejemplos adicionales sobre el cálculo de derivadas de orden superior.

```
--> f(x,y,z):=x^3+x*y^2+z^2*cos(x);

--> define(fxx(x,y,z),diff(f(x,y,z),x,2));

--> fxx(1,1,1);

--> define(fxy(x,y,z),diff(f(x,y,z),x,1,y,1));

--> define(fyx(x,y,z),diff(f(x,y,z),y,1,x,1));

--> define(fxyz(x,y,z),diff(f(x,y,z),x,1,y,1,z,1));

--> define(fxxz(x,y,z),diff(f(x,y,z),x,2,z,1));

--> fxxz(1,0,1);

--> define(fxxxx(x,y,z),diff(f(x,y,z),x,4));

--> kill(all);
```

Como podemos observar, por defecto wxMaxima asume que las derivadas cruzadas de una función genérica son iguales.

```
--> diff(f(x,y),x,1,y,1); diff(f(x,y),y,1,x,1); diff(f(x,y),x,1,y,1)-diff(f(x,y),y,1,x,1);

--> diff(f(x,y),x,4,y,2); diff(f(x,y),x,1,y,1,x,3,y,1); diff(f(x,y),x,4,y,2)- diff(f(x,y),x,1,y,1,x,3,y,1);
```

3 Vectores gradientes y matrix Hessiana. Puntos críticos.

Para el cálculo de vectores gradientes y matrices hessianas usaremos, respectivamente, las órdenes "jacobian" y "hessian" tal como indicamos a continuación.

Recordamos que el vector gradiente de una función es el vector cuyas componentes son las derivadas parciales primeras de dicha función. Asimismo, la matriz hessiana es la formada por las derivadas parciales segundas.

```
--> kill(all);

--> f(x,y):=x*sin(y);

--> gradiente:jacobian([f(x,y)],[x,y]);

--> matrizhess:hessian(f(x,y),[x,y]);
```

El determinante hessiano que permite clasificar los posibles puntos criticos se obtiene sin más que calcular el determinante de la matriz correspondiente.

```
--> define(hessiano(x,y),ev(determinant(matrizhess),x=x,y=y));
```

Finalmente, determinamos los posibles puntos críticos resolviendo el sistema (por lo general, no lineal) que surge al anular el vector gradiente.

```
--> load(to_poly_solve);

--> nicedummies(%solve([gradiente[1,1]=0,gradiente[1,2]=0],[x,y]));
```

Observamos que los puntos críticos son aquellos de la forma (0,k*%pi), con k entero cualquiera. Evaluemos finalmente el determinante hessiano en dichos puntos.

```
--> declare(k,integer);

--> hessiano(0,k*%pi);
```

Resulta entonces que la función de partida tiene infinitos puntos de silla de la forma (0,k*%pi), con k entero. Veamos su comportamiento gráficamente.

```
--> plot3d(f(x,y),[x,-3*%pi,3*%pi],[y,-3*%pi,3*%pi],[grid, 50,50]);
```

Veamos ahora otros ejemplos con funciones polinómicas.

```
--> f(x,y):=x*y-x^4-y^4;

--> gradiente:jacobian([f(x,y)],[x,y]);

    matrizhess:hessian(f(x,y),[x,y]);

    define(hessiano(x,y),ev(determinant(matrizhess),x=x,y=y));

--> nicedummies(%solve([gradiente[1,1]=0,gradiente[1,2]=0],[x,y]));
```

Observamos que en este caso el sistema resultante de anular el gradiente posee también soluciones complejas. Podemos forzar la devolución de únicamente puntos críticos reales del modo siguiente:

```
--> realonly:true;

--> puntoscriticos:nicedummies(%solve([gradiente[1,1]=0,gradiente[1,2]=0],[x,y]));
```

Obtenemos entonces únicamente tres puntos críticos reales. Finalmente evaluamos el determinante hessiano en dichos puntos.

```
--> part(puntoscriticos,1);
    hessiano(x,y),part(puntoscriticos,1);
    subst(part(puntoscriticos,1),hessiano(x,y));
```

Evaluamos la derivada parcial segunda fxx en el punto crítico.

```
--> subst(part(puntoscriticos,1),diff(f(x,y),x,2));
```

Luego, comprobamos que el punto crítico (-1/2,-1/2) es un máximo relativo.

```
--> part(puntoscriticos,2);
    subst(part(puntoscriticos,2),hessiano(x,y));
```

El punto crítico (0,0) es un punto de silla.

```
--> part(puntoscriticos,3);
    hessiano(x,y),part(puntoscriticos,3);
```

```
--> subst(part(puntoscriticos,3),diff(f(x,y),x,2));
```

Por tanto, el punto crítico (1/2,1/2) es también un máximo relativo.

```
--> plot3d(f(x,y),[x,-1,1],[y,-1,1],[grid, 50,50]);
```

Observar que al tener únicamente tres puntos críticos podríamos haber evaluado el hessiano directamente. No obstante, laas anteriores instrucciones resultan de mayor generalidad y evitan considerar casos concretos.

```
--> hessiano(-1/2,-1/2); hessiano(0,0); hessiano(1/2,1/2);
```

Finalizamos esta sección indicando otro modo de obtener los puntos críticos en el caso de funciones polinómicas.

```
--> kill(all);

--> f(x,y):=x^3-y^2+2*y;

--> gradiente:jacobian([f(x,y)],[x,y]);

    matrizhess:hessian(f(x,y),[x,y]);

    define(hessiano(x,y),ev(determinant(matrizhess),x=x,y=y));
```

Podemos usar la orden "algsys" en el caso algebraico.

```
--> puntoscriticos:algsys([gradiente[1,1]=0,gradiente[1,2]=0],[x,y]);
```

Finalmente, evaluamos el hessiano en los elementos de la lista anterior.

```
--> hessiano(x,y),puntoscriticos[1];
```

El criterio del hessiano no resulta concluyente. Sin embargo a la vista de la gráfica en un entorno del punto crítico comprobamos que se trata de un punto de silla.

```
--> plot3d(f(x,y),[x,-1,1],[y,-1,1],[grid, 50,50]);
```

4 Plano tangente a una superficie z=f(x,y) en un punto.

Veamos finalmente cómo obtener y representar el plano tangente a una superficie z=f(x,y) en un punto (x,y)=(a,b).

La ecuación cartesiana de dicho plano es: z=f(a,b)+fx(a,b)*(x-a)+fy(a,b)*(y-b).

```
--> f(x,y):=x^3-y^2+2*y;

--> a:0; b:1;

--> define(fx(x,y),diff(f(x,y),x)); define(fy(x,y),diff(f(x,y),y));

--> define(planotang(x,y),f(a,b)+fx(a,b)*(x-a)+fy(a,b)*(y-b));

--> plot3d([f(x,y),planotang(x,y),[x,-1,1],[y,0,2]]);

--> f(x,y):=x^2+4*x+y^2;

--> a:-2; b:0;

--> define(fx(x,y),diff(f(x,y),x)); define(fy(x,y),diff(f(x,y),y));
```

```
define(planotang(x,y),f(a,b)+fx(a,b)*(x-a)+fy(a,b)*(y-b));
plot3d([f(x,y),planotang(x,y),[x,-3,-1],[y,-1,1]]);
--> a:1; b:1;
--> define(planotang(x,y),f(a,b)+fx(a,b)*(x-a)+fy(a,b)*(y-b));
plot3d([f(x,y),planotang(x,y),[x,-4,4],[y,-4,4]]);
```

Si se desea representar cada superficie con un color distinto debemos usar la orden "draw3d".

```
--> graficaf:explicit(f(x,y),x,-4,4,y,-4,4);
graficaplano:explicit(2*(y-1)+6*(x-1)+6,x,-4,4,y,-4,4);
--> draw3d(xu_grid=40,yv_grid=40,graficaf,color=red,graficaplano,color=green);
--> draw3d(surface_hide=true,xu_grid=40,yv_grid=40,graficaf,color=red,graficaplano,color=green);
```

Created with wxMaxima.