Actividad 8: Iniciandose en Computo Simbolico con Maxima

Luisa Fernanda Orci Fernandez.

01 de Abril del 2016

Descripción de la actividad

Para esta actividad se nos pidió leer y explorar un manual de Maxima.

Maxima es un sistema de álgebra computcional, se especializa en operaciones simbólicas. [1] En Maxima se pueden realizar expansiones de series de Taylor, transformadas de Laplace, también puede resolver matrices, sistemas de ecuaciones líneales, ecuaciones diferenciales, entre otros.
Para saber utilizar Maxima fue necesario hacer los ejercicios que venían propuestos en el manual, solo que le modificamos los colores y las funciones.

A continuación se muestran los resultados obtenidos.

1. Geometría en tres dimensiones

1.1. Vectores y álgebra líneal

En esta sección aprendemos a realizar operaciones con vectores; desde como es la notación de

Maxima es un sistema de álgebra un vector en Maxima, hasta como operarlos entre si, ya sea sumandolos, multiplicandolos, calculando el producto punto o el producto cruz. A continuación se muestran los resultados obtenidos:

```
(%i1) a: [3, 6, 8];
(%o1)                                [3, 6, 8]
(%i2) b: [9, 10, 15];
(%o2)                               [9, 10, 15]
(%i3) 3*a;
(%o3)                               [9, 18, 24]
(%i4) b + a;
```

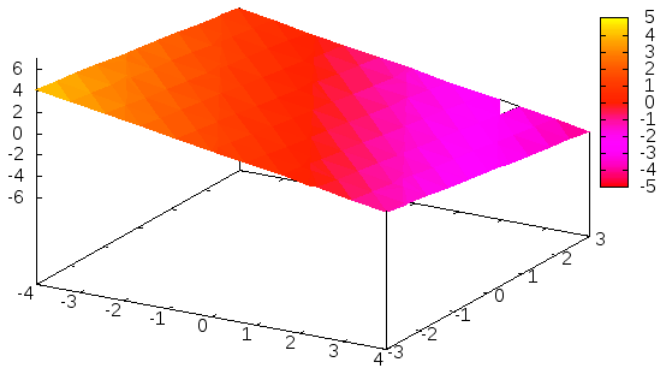
```
(%o4)                              [12, 16, 23]
(%i5)  a.b;
(%o5)                                   207
(%i6) load(vect);
(%o6)           /usr/share/maxima/5.21.1/share/vector/vect.mac
(%i7) a~b;
(%o7)                         [3, 6, 8] ~ [9, 10, 15]
(%i8) express(a~b);
(%o8)                              [10, 27, - 24]
(%i9)
```

1.2. Líneas, planos y superficies cuadráticas

Aquí se muestra como utilizar el comando *draw*3*d*, que es para gráficar superficies y el comando *palette* que es para modificar los colores. A continuación se muestra como utilizar Maxima para graficar un plano, así como un hiperboloide.

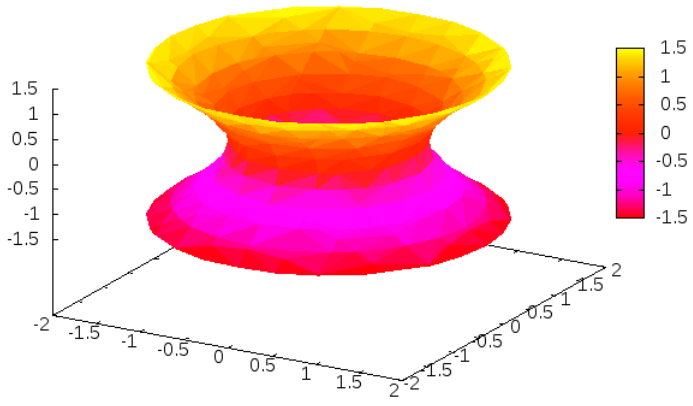
Plano:

```
(%i1) plane1: 4*x + 3*y + 6*z = 0;
(%o1)                           6 z + 3 y + 4 x = 0
(%i2) load(draw);
(%o2)             /usr/share/maxima/5.21.1/share/draw/draw.lisp
(%i3) draw3d(enhanced3d=true,implicit(plane1, x,-4,4, y,-3,3, z,-7,7),palette=[2,5,15]
(%o3)                            [gr3d(implicit)]
```



Hiperboloide:

```
                                  2    2    2
(%o1)                           - z  + y  + x  = 1
(%i2) load(draw);
(%o2)             /usr/share/maxima/5.21.1/share/draw/draw.lisp
(%i3) draw3d(enhanced3d=true, implicit(hyperboloid, x,-2,2, y,-2,2, z,-1.5,1.5), palet
(%o3)                            [gr3d(implicit)]
```

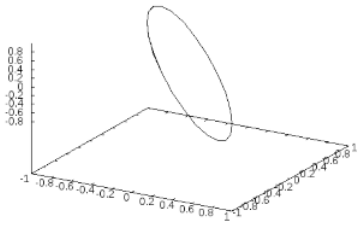


1.3. Funciones vectoriales evaluadas

Aquí propusimos una función, se evaluó en un punto, calculamos su límite y su derivada y se graficó, todo esto fue posible gracias a que se siguió el manual de Maxima.

```
(%i1) r(t) := [t, cos(t), sin(t)];
(%o1)                      r(t) := [t, cos(t), sin(t)]
(%i2) r(2);
(%o2)                           [2, cos(2), sin(2)]
(%i3) float(r(2));
(%o3)              [2.0, - 0.41614683654714, 0.90929742682568]
(%i4) load(draw);
(%o4)             /usr/share/maxima/5.21.1/share/draw/draw.lisp
(%i5)  draw3d(parametric(cos(t), -cos(t), sin(t), t, -4, 4));
(%o5)                           [gr3d(parametric)]

(%i6)limit(r(t), t, 2);
(%o6)                           [2, cos(2), sin(2)]
(%i7) float(limit(r(t), t, 2));
(%o7)              [2.0, - 0.41614683654714, 0.90929742682568]
(%i8) limit(r(t), t, 2, plus);
(%o8)                           [2, cos(2), sin(2)]
(%i9) limit(r(t), t, 3, minus);
(%o9)                           [3, cos(3), sin(3)]
(%i10) diff(r(t), t);
(%o10)                        [1, - sin(t), cos(t)]
(%i11)
```



1.4. Longitud de arco y curvatura

En esta sección se realizó un ejemplo de curvatura que fue propuesto por el manual de Maxima, y se utilizó la fórmula de la curvatura. En este ejercicio Maxima se utilizo como herramienta, ya que el software no es capaz de calcular la curvatura, este ejecuta los pasos para poderla calcular, la curvatura se define de la siguiente forma:

$$k = \frac{||r'(t) \times r''(t)||}{||r'(t)^3||}$$

Utilizando esta definición, pudimos obtener los componentes para calcularla:

```
(%i1) f(t) := [t, cos(t), sin(t)];
(%o1)                      f(t) := [t, cos(t), sin(t)]
(%i2) fp(t) := [1, -sin(t), cos(t)];
(%o2)                    fp(t) := [1, - sin(t), cos(t)]
(%i3) Tp(t) := [0, -cos(t), sin(t)]/sqrt(2);
                                  [0, - cos(t), sin(t)]
(%o3)                    Tp(t) := ---------------------
                                          sqrt(2)
(%i4) sqrt(Tp(t).Tp(t))/sqrt(rp(t).rp(t));
                                     2         2
                                  sin (t)   cos (t)
                             sqrt(------- + -------)
                                     2         2
(%o4)                        ----------------------
                                            <2>
                                 sqrt(rp(t)   )
(%i5) trigsimp(sqrt(Tp(t).Tp(t))/sqrt(rp(t).rp(t)));
                                       1
(%o5)                        ---------------------
                                              <2>
                             sqrt(2) sqrt(rp(t)   )
(%i6) define(kappa(t), sqrt(Tp(t).Tp(t))/sqrt(rp(t).rp(t)));
                                          2         2
```

```
                                          sin (t)   cos (t)
                                       sqrt(------- + -------)
                                              2         2
(%o6)                kappa(t) := ------------------------
                                                  <2>
                                           sqrt(rp(t)   )
(%i7) integrate(f(t), t);
                                2
                               t
(%o7)                         [--, sin(t), - cos(t)]
                               2
(%i8) g(t):= [2*t, 3*sin(t), 3*cos(t)];
(%o8)                   g(t) := [2 t, 3 sin(t), 3 cos(t)]
(%i9) define(gp(t), diff(g(t), t));
(%o9)                  gp(t) := [2, 3 cos(t), - 3 sin(t)]
(%i10) integrate(trigsimp(sqrt(gp(t).gp(t))), t, 0, 2*%pi);
(%o10)                          2 sqrt(13) %pi
(%o11)                         22.65434679827795
(%i12)
```

2. Funciones de varias variables

Derivadas parciales

Se propuso una función de varias variables y calculamos su derivada parcial con el comando utilizado para calcular una derivada total, la diferencia fue que ahora se le especificó a Maxima con respecto a que variable se quería operar.

```
(%i1) diff(f(x,y), x);
                                d
(%o1)                           -- (f(x, y))
                                dx
(%i2) diff(diff(f(x,y), x), x);
                                 2
                                d
(%o2)                           --- (f(x, y))
                                  2
                                dx
(%i3) diff(diff(f(x,y), x), y);
                                 2
                                d
(%o3)                           ----- (f(x, y))
```

```
                                    dx dy
(%i4) G: x^6 * y^5 ;
                                         6  5
(%o4)                                   x  y
(%i5) diff(G, x, 1, y, 2, x, 3);
                                           2  3
(%o5)                                  7200 x  y
(%i6)
```

2.1. Aproximación líneal

Aquí se utilizó un polinomio de Taylor para aproximar una función. Maxima facilitó las coas, ya que cuenta con un comando para realizar este tipo de aproximaciones.

```
(%i1) f(x, y) := exp(x^2) * sin(y);
                                                2
(%o1)                        f(x, y) := exp(x ) sin(y)
(%i2) taylor(f(x,y), [x,y], [1,2], 1);
(%o2)/T/ sin(2) %e + (2 sin(2) %e (x - 1) + cos(2) %e (y - 2)) + . . .
(%i3) diff(f(x,y));
                     2                          2
                    x                          x
(%o3)              %e   cos(y) del(y) + 2 x %e   sin(y) del(x)
(%i4)
```

2.2. Regla de la cadena

Aquí volvimos a utilizar la función anterior, pero esta vez la diferenciamos por la regla de la cadena en Maxima. Fue necesario parametrizar esta función y diferenciarla con respecto a cada una de las variables nuevas:

```
(%i1) f(x,y) := exp(x^2) * sin(y);
                                                2
(%o1)                        f(x, y) := exp(x ) sin(y)
(%i2) [x, y] : [s^2 * t, s*t^2];
                                      2       2
(%o2)                                [s  t, s t ]
(%i3) diff(f(x,y), s);
                            4  2                    4  2
                  3  2   s  t          2     2   s  t          2
(%o3)          4 s  t  %e      sin(s t ) + t  %e      cos(s t )
(%i4) diff(f(x,y), t);
                         4  2                     4  2
```

```
                4     s  t         2            s  t         2
(%o4)         2 s  t %e      sin(s t ) + 2 s t %e      cos(s t )
(%i5) diff(f(x,y), x);

                                                    2
diff: second argument must be a variable; found s  t
 -- an error. To debug this try: debugmode(true);
(%i6) diff(f(u,v), u);
                                      2
                                     u
(%o6)                          2 u %e   sin(v)
(%i7) kill(x,y);
(%o7)                               done
(%i8) diff(f(x,y), x);
                                      2
                                     x
(%o8)                          2 x %e   sin(y)
(%i9)
```

2.3. Derivada direccional y gradiente

Se propuso una función, la cual se convirtió en función vectorial para así poder calcular su gradiente con Maxima, ya que cuenta con un comando especifico para este calculo. Una vez que se obtiene el gradiente, este se evalua en un punto y se calcula la derivada direccional con la una formula previamente aprendida en el curso de Cálculo III:

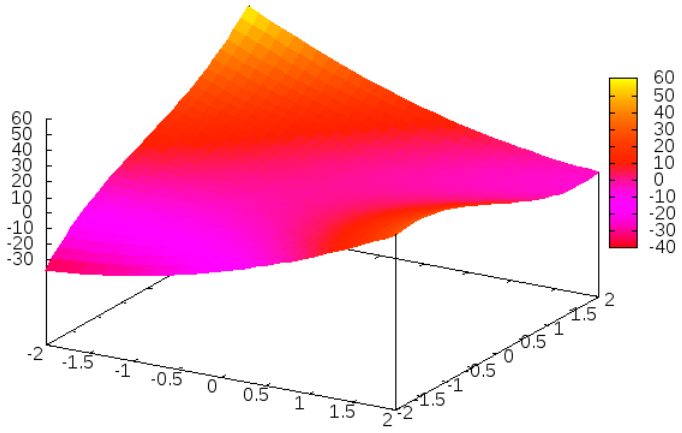
```
(%i1) f(x,y) := exp(x^2) * cos(y);
                                          2
(%o1)                    f(x, y) := exp(x ) cos(y)
(%i2) load(vect);
(%o2)          /usr/share/maxima/5.21.1/share/vector/vect.mac
(%i3) scalefactors([x,y]);
(%o3)                               done
(%i4) gdf: grad(f(x,y));
                                      2
                                     x
(%o4)                        grad (%e   cos(y))
(%i5) ev(express(gdf), diff);
                              2             2
                             x             x
(%o5)                 [2 x %e   cos(y), - %e   sin(y)]
(%i6) define(gdf(x,y), ev(express(gdf), diff));
```

```
                                      2              2
                                     x              x
(%o6)            gdf(x, y) := [2 x %e   cos(y), - %e   sin(y)]
(%i7) v: [3, 4];
(%o7)                                [3, 4]
(%i8) (gdf(1,2).v)/sqrt(v.v);
                           6 %e cos(2) - 4 %e sin(2)
(%o8)                      ------------------------
                                       5
(%i9) ev((gdf(1,2).v)/sqrt(v.v), diff);
                           6 %e cos(2) - 4 %e sin(2)
(%o9)                      ------------------------
                                       5
(%i10) float(ev((gdf(1,2).v)/sqrt(v.v), diff));
(%o10)                        - 3.334826598112032
(%i11) sqrt(gdf(1,2).gdf(1,2));
                              2    2          2    2
(%o11)                  sqrt(%e  sin (2) + 4 %e  cos (2))
(%i12) float(ev((gdf(1,2).v)/sqrt(v.v), diff));
(%o12)                        - 3.334826598112032
(%i13)
```

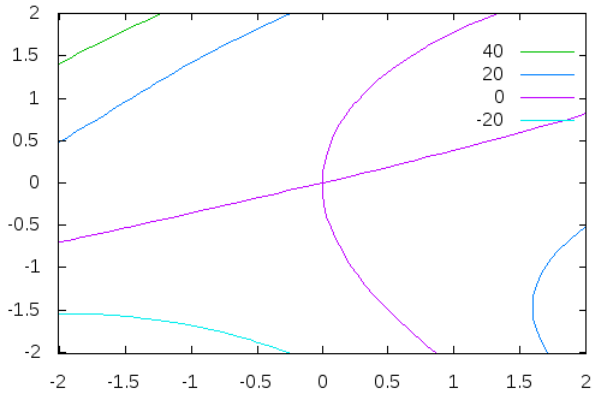
2.4. Optimización y Extremos locales

De nuevo se propuso una función y se graficó y se siguió un procedimiento para optimizar propuesto por el manual, se utilizaron las derivadas parciales y una matriz a la cual se le calculó el determinante, todo esto para calcular el punto crítico de una función:

```
(%i1) f(x,y) := 3*x^2 + 2*y^3 - 8*x*y;
                                    2      3
(%o1)                 f(x, y) := 3 x  + 2 y  + (- 8) x y
(%i2) load(draw);
(%o2)            /usr/share/maxima/5.21.1/share/draw/draw.lisp
(%i3) draw3d(enhanced3d = true, explicit(f(x,y), x, -2,2, y,-2,2), palette=[2,5,15]);
(%o3)                          [gr3d(explicit)]
```




```
(%i4)draw3d(explicit(f(x,y), x,-2,2, y,-2,2), contour=map, palette=[2,5,15]);
(%o4)                          [gr3d(explicit)]
```




```
(%i5) fx: diff(f(x,y), x);
(%o5)                              6 x - 8 y
(%i6) fy : diff(f(x,y), y);
                                     2
(%o6)                              6 y  - 8 x
(%i7) solve([fx, fy], [x,y]);
                           64      16
(%o7)                [[x = --, y = --], [x = 0, y = 0]]
                           27      9
(%i8) H: hessian(f(x,y), [x,y]);
                                  [  6    - 8  ]
(%o8)                             [            ]
                                  [ - 8   12 y ]
(%i9) determinant(H);
(%o9)                              72 y - 64
```

```
(%i10) subst([x = -1, y=-1], diff(fx, x));
(%o10)                                6
(%i11) subst([x = -1, y = -1], determinant(H));
(%o11)                              - 136
(%i12) f(-1, 1);
(%o12)                                13
(%i13)
```

2.5. Multiplicador de Lagrange

En esta sección se calcularon los valores extremos de una superficie que estaba limitada por otra y para lograr esto, se propusieron dos funciones, estas se diferenciaron y se resolvió el sistema de ecuaciones diferencles para poder encontrar los valores extremos y al final se evaluó esta función en esos puntos para poder encontrar el máxio y mínimo:

```
%i1) f(x,y) := 4*x^2 + y^2;
                                            2    2
(%o1)                        f(x, y) := 4 x  + y
(%i2)  g: x^2 + y^2;
                                          2    2
(%o2)                                    y  + x
(%i3) eq1: diff(f(x,y), x) = h*diff(g,x);
(%o3)                                 8 x = 2 h x
(%i4) eq2: diff(f(x,y), y) = h*diff(g,y);
(%o4)                                 2 y = 2 h y
(%i5) eq3: g=1;
                                       2    2
(%o5)                                  y  + x  = 1
(%i6) solve([eq1, eq2, eq3], [x, y, h]);
(%o6) [[x = 1, y = 0, h = 4], [x = - 1, y = 0, h = 4],
                                  [x = 0, y = - 1, h = 1], [x = 0, y = 1, h = 1]]
(%i7) [f(1,0), f(-1,0), f(0,-1), f(0,1)];
(%o7)                                 [4, 4, 1, 1]
(%i8)
```

Conclusiones

Esta actividad, como todas las anteriores, se me hizo de gran interes, no solo por que ahora sabemos como graficar espacios fase en *Python*, si no por que en un futuro será de gran ayuda.

Referencias

[1] Wikipedia, *Maxima*, (2016, 01 de Abril).Desde: `https://es.wikipedia.org/wiki/Maxima`