

# Actividad 8: Iniciandose en Computo Simbólico con Maxima

Paulina Valenzuela Coronado

Abril de 2016

## 1. Introducción

Maxima es un sistema para la manipulación de expresiones simbólicas y numéricas, incluyendo diferenciación, integración, expansión en series de Taylor, transformadas de Laplace, ecuaciones diferenciales ordinarias, sistemas de ecuaciones lineales, vectores, matrices y tensores. Maxima produce resultados de alta precisión usando fracciones exactas, números enteros de precisión arbitraria y números de coma flotante con precisión variable. Adicionalmente puede graficar funciones y datos en dos y tres dimensiones.

El código fuente de Maxima puede ser compilado en varios sistemas operativos incluyendo Windows, Linux y MacOS X. El código fuente para todos los sistemas y los binarios pre-compilados para Windows y Linux están disponibles en el Administrador de archivos de SourceForge.

Maxima es un descendiente de Macsyma, el legendario sistema de álgebra computacional desarrollado a finales de 1960 en el Instituto Tecnológico de Massachusetts (MIT). Este es el único sistema basado en ese programa que está todavía disponible publicamente y con una comunidad activa de usuarios, gracias a la naturaleza del software abierto. Macsyma fue revolucionario en sus días y muchos sistemas posteriores, tales como Maple y Mathematica, se inspiraron en él. [1]

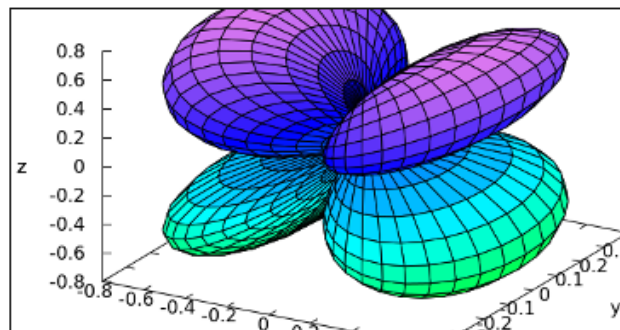


Figura 1: Gráficos realizados en Maxima  
[1]

## Actividad

En esta actividad nos familiarizamos con las funciones básicas de Maxima, usando como referencia el manual de Jay Kerns[2], se repite un ejercicio de cada sección, modificando colores, variables etc.

A continuación se presentan dichos ejemplos:

## 2. Geometría en R3

### 2.1. Vectores y Álgebra Lineal

Podemos expresar los vectores entre corchetes y realizar distintas operaciones matemáticas con ellos.

```
(%i1) a: [2,3,4];  
      b: [5,6,7];  
      sqrt(a.b)/(a.a)*b;  
(%o1) [2, 3, 4]  
(%o2) [5, 6, 7]  
(%o3) [ $\frac{10\sqrt{14}}{29}$ ,  $\frac{12\sqrt{14}}{29}$ ,  $\frac{14\sqrt{14}}{29}$ ]
```

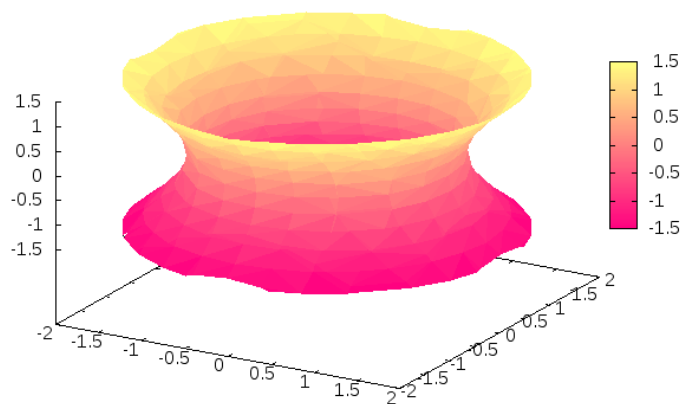
### 2.2. líneas, planos y superficies cuadráticas

Podemos graficar planos con el paquete *implicitplot* de la librería DRAW.

```
hyperboloid: x^2 + y^2 - z^2 = 2;
```

```
load(draw);
```

```
draw3d(enhanced3d = true, palette=[2,3,1], implicit(hyperboloid, x,-2,2, y,-2,2, z,-1
```



## 2.3. Funciones Vectoriales

Podemos definir una función vectorial como cualquier otra función y el valor dado será otro vector.

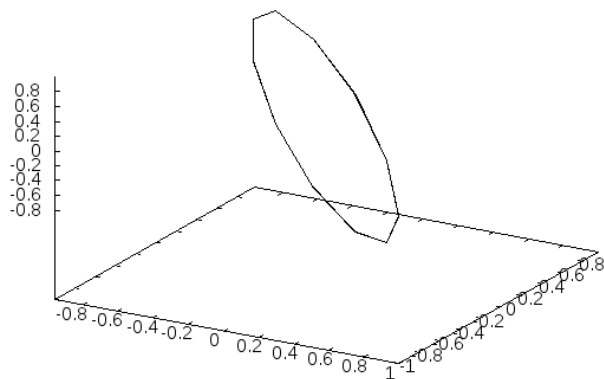
```
r(t) := [t, cos(t), sin(t)];
```

```
r(2);
```

```
float(%);
```

```
load(draw);
```

```
draw3d(parametric(cos(t), -cos(t), sin(t), t, -8, 8));
```



## 2.4. Curvatura y longitud de arco

No existe una función especial para la curvatura en maxima, pero sin embargo podemos calcularla con las operaciones matemáticas aplicadas a los vectores

```
g(t) := [2* t, 3* cos(t), 3* sin(t)];

define(gp(t), diff(g(t), t));

integrate(trigsimp(sqrt(gp(t) . gp(t))), t, 0, %pi/2);
```

[Link to solution](#)
[Export to Image](#)
[Export to wxMaxima file](#)

```
(%i1) g(t) := [2* t, 3* cos(t), 3* sin(t)];
(%o1) g(t) := [2 t, 3 cos(t), 3 sin(t)]
(%i2) define(gp(t), diff(g(t), t));
(%o2) gp(t) := [2, - 3 sin(t), 3 cos(t)]
(%i3) integrate(trigsimp(sqrt(gp(t) . gp(t))), t, 0, %pi/2);
(%o3)
      sqrt(13) %pi
      -----
             2
(%i4)
```

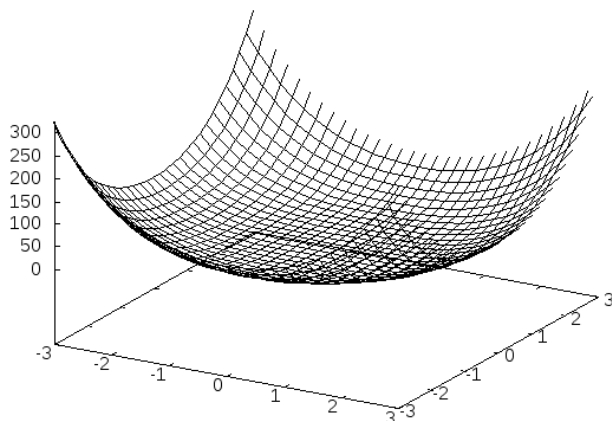
## 3. Funciones de Varias Variables

```
f(x,y) := (x^2 + y^2)^2;

load(draw);

draw3d(palette=[2,3,1], explicit(f(x,y), x, -3, 3, y, -3, 3));
```

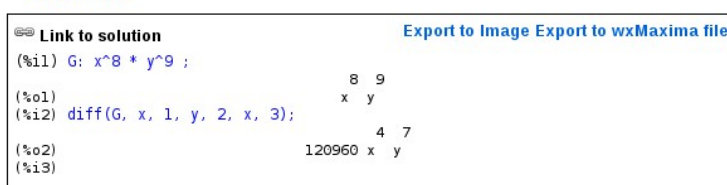
```
draw3d(explicit(f(x,y), x, -5, 5, y, -5, 5),
contour_levels = 15,
contour
= map);
```



### 3.1. Derivadas Parciales

Podemos realizar derivadas parciales usando el comando *diff*.

```
G: x^8 * y^9 ;
diff(G, x, 1, y, 2, x, 3);
```



### 3.2. Aproximación Lineal y Diferenciales

Maxima encuentra una aproximación a cualquier función mediante series de Taylor.

```
f(x,y) := x^2 * sin(y);
taylor(f(x,y), [x,y], [1,2], 1);
```

```

Link to solution Export to Image Export to wxMaxima file
(%i1) f(x,y) := x^2 * sin(y);
(%o1) f(x, y) := x^2 sin(y)
(%i2) taylor(f(x,y), [x,y], [1,2], 1);
(%o2) T/ sin(2) + (2 sin(2) (x - 1) + cos(2) (y - 2)) + . . .
(%i3)

```

### 3.3. Regla de la Cadena y Diferenciales Implícitas

Maxima realiza la regla de la cadena automáticamente usando el comando *diff* y para realizar diferenciales implícitas solamente usamos el comando *Fn* antes de derivar, donde *n* es la variable con respecto a la cual derivar.

```

f(x,y) := x^2 * sin(y);
[x,y] : [s^2 * t, s * t^2];
diff(f(x,y), s);
diff(f(x,y), t);
diff(f(u,v), u);
kill(x, y);
diff(f(x,y), x);

```

```

Link to solution Export to Image Export to wxMaxima file
(%i1) f(x,y) := x^2 * sin(y);
(%o1) f(x, y) := x^2 sin(y)
(%i2) [x,y] : [s^2 * t, s * t^2];
(%o2) [s^2 t, s t^2]
(%i3) diff(f(x,y), s);
(%o3) 4 s^2 t sin(s t) + s^4 t^4 cos(s t)
(%i4) diff(f(x,y), t);
(%o4) 2 s^4 t sin(s t) + 2 s^5 t^3 cos(s t)
(%i5) diff(f(u,v), u);
(%o5) 2 u sin(v)
(%i6) kill(x, y);
(%o6) done
(%i7) diff(f(x,y), x);
(%o7) 2 x sin(y)
(%i8)

```

### 3.4. Derivadas Direccionales y Gradiente

Para encontrar una derivada direccional máxima requiere que la función esté dada en forma de vector, para esto usamos el comando *load(vect)*; para el gradiente usamos *gdf(a,b)*.

```

f(x,y) := (x^2) * cos(y);

load(vect);

scalefactors([x,y]);

gdf: grad(f(x,y));

```

```
ev(express(gdf), diff);
```

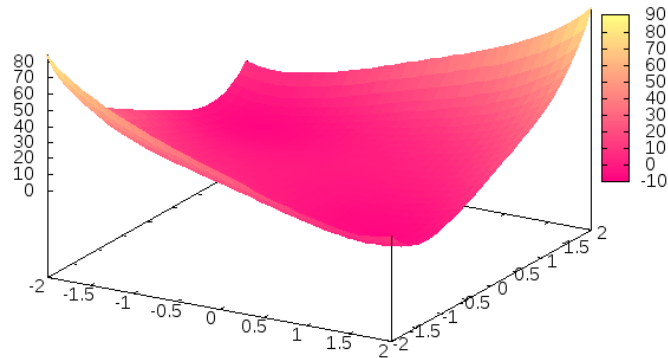
**Calculate**

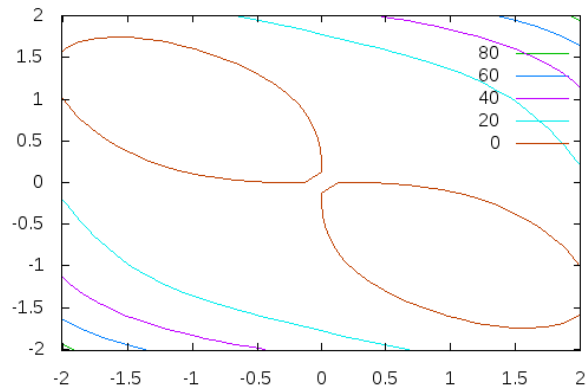
[Link to solution](#) [Export to Image](#) [Export to wxMaxima file](#)

```
(%i1) f(x,y) := (x^2) * cos(y);
(%o1) f(x, y) := x^2 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));
(%o4) grad (x^2 cos(y))
(%i5) ev(express(gdf), diff);
(%o5) [2 x cos(y), - x^2 sin(y)]
(%i6)
```

### 3.5. Optimización y los extremos locales

```
f(x,y) := x^4 + 2 * y^4 + 9 * x * y;
load(draw);
draw3d(palette=[2,3,1], enhanced3d = true, explicit(f(x,y), x, -2, 2, y, -2, 2));
draw3d(explicit(f(x,y), x, -2, 2, y, -2, 2), contour= map);
```





### 3.6. Multiplicadores de Lagrange

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

```
g: x^2 - y^2;
```

```
eq1: diff(f(x,y), x) = h * diff(g, x);
```

```
eq2: diff(f(x,y), y) = h * diff(g, y);
```

```
eq3: g = 1;
```

```
solve([eq1, eq2, eq3], [x, y, h]);
```

```
[f(1,0), f(-1,0), f(0,-1), f(0,1)];
```

```
(%i26) f(x,y) := x^2 + 2*y^2;
      g: x^2 - y^2;
      eq1: diff(f(x,y), x) = h * diff(g, x);
      eq2: diff(f(x,y), y) = h * diff(g, y);
      eq3: g = 1;
      solve([eq1, eq2, eq3], [x, y, h]);
      [f(1,0), f(-1,0), f(0,-1), f(0,1)];
(%o26) f(x,y):=x^2+2 y^2
(%o27) x^2-y^2
(%o28) 2 x=2 h x
(%o29) 4 y=-2 h y
(%o30) x^2-y^2=1
(%o31) [[x=1,y=0,h=1],[x=-1,y=0,h=1],[x=0,y=%i,h=-2],[x=0,y=-%i,h=-2]]
(%o32) [1,1,2,2]
```



## 4. Integración Múltiple

### 4.1. Integrales Dobles

```
f(x,y) := 3*x^3 - x*y;  
integrate(integrate(f(x,y), y), x);  
integrate(integrate(f(x,y), y, x^2, x), x, 0, 1);
```

[Link to solution](#) [Export to Image](#) [Export to wxMaxima file](#)

```
(%i1) f(x,y) := 3*x^3 - x*y;  
  
(%o1) f(x, y) := 3 x^3 - x y  
(%i2) integrate(integrate(f(x,y), y), x);  
  
(%o2) 4 2 2  
3 x y x y  
-----  
4 4  
(%i3) integrate(integrate(f(x,y), y, x^2, x), x, 0, 1);  
  
(%o3) 7  
120  
(%i4)
```

### 4.2. Integración en Coordenadas Polares

```
f(x,y) := x^2 + y^2;  
  
[x,y]: [r * cos(theta), r * sin(theta)];  
  
integrate(integrate(f(x,y) * r, r, 0, 2*cos(theta)), theta, 0, %pi/2);
```

[Link to solution](#) [Export to Image](#) [Export to wxMaxima file](#)

```
(%i1) f(x,y) := x^2 + y^2;  
  
(%o1) f(x, y) := x^2 + y^2  
(%i2) [x,y]: [r * cos(theta), r * sin(theta)];  
  
(%o2) [r cos(theta), r sin(theta)]  
(%i3) integrate(integrate(f(x,y) * r, r, 0, 2*cos(theta)), theta, 0, %pi/2);  
  
(%o3) 3 %pi  
4  
(%i4)
```

### 4.3. Integrales Triples

```
integrate(integrate(integrate(x*y^2*z, z, x, 2*x+y), y, 2, -x), x, 0, 4);
```

[Link to solution](#) [Export to Image](#) [Export to wxMaxima file](#)

```
(%i1) integrate(integrate(integrate(x*y^2*z, z, x, 2*x+y), y, 2, -x), x, 0, 4);  
  
(%o1) 144128  
-----  
105  
(%i2)
```

## 4.4. Integrales en Coordenadas Cilíndricas y Esféricas

```
f(x,y,z) := x*y*z;
[x,y,z] : [r*cos(theta), r*sin(theta), z];
integrate(integrate(integrate(f(x,y,z)*r, z,2,8), r,1,4), theta,0,%pi/2);

kill(f,x,y,z);
f(x,y,z) := x*y*z;
[x,y,z] : [rho*sin(phi)*cos(theta), rho*sin(phi)*sin(theta), rho*cos(theta)];
integrate(integrate(integrate(f(x,y,z)*rho^2*sin(phi),rho,0,1),theta,0,%pi),
phi,0,%pi/2);
```

[Link to solution](#) [Export to Image](#) [Export to wxMaxima file](#)

```
(%i1) f(x,y,z) := x*y*z;
(%o1) f(x, y, z) := x y z
... (%i2) [x,y,z] : [r*cos(theta), r*sin(theta), z];
(%o2) [r cos(theta), r sin(theta), z]
(%i3) integrate(integrate(integrate(f(x,y,z)*r, z,2,8), r,1,4), theta,0,%pi/2);
(%o3) 3825
      ----
      4
(%i4)
```

[Link to solution](#) [Export to Image](#) [Export to wxMaxima file](#)

```
(%i1) kill(f,x,y,z);
(%o1) done
(%i2) f(x,y,z) := x*y*z;
(%o2) f(x, y, z) := x y z
(%i3) [x,y,z] : [rho*sin(phi)*cos(theta), rho*sin(phi)*sin(theta),
rho*cos(theta)];
(%o3) [sin(phi) rho cos(theta), sin(phi) rho sin(theta), rho cos(theta)]
(%i4) integrate(integrate(integrate(f(x,y,z)*rho^2*sin(phi),rho,0,1),theta,0,%pi),phi,0
,%pi/2);
(%o4) 2
      --
      27
);
(%i5)
```

## 4.5. Cambio de Variable

```
f(x,y) := 2*x + 3*y;
[x,y]: [u^2 - 2*v^4, 5 * u * v];
J: jacobian([x,y], [u,v]);
J: determinant(J);
integrate(integrate(f(x,y)*J,u,1,2),v,3,4)
```

[Link to solution](#)
[Export to Image](#)
[Export to wxMaxima file](#)

```

(%i1) f(x,y) := 2*x + 3*y;
(%o1) f(x, y) := 2 x + 3 y
(%i2) [x,y]: [u^2 - 2*v^4, 5 * u * v];
(%o2) [u^2 - 2 v^4, 5 u v]
(%i3) J: jacobian([x,y], [u,v]);
(%o3) [2 u - 8 v^3, 5 v]
      [5 v, 5 u]
nt(m1));
(%i4) J: determinant(J);
(%o4) 40 v^4 + 10 u^2
(%i5) integrate(integrate(f(x,y)*J,u,1,2),v,3,4);
(%o5) 136393069
4*%5) -----
      36
(%i6)

```

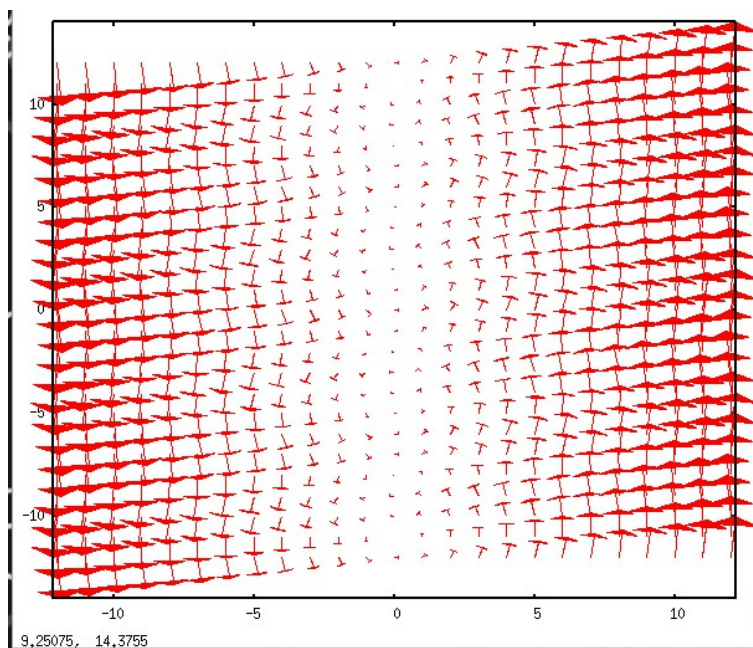
## 5. Cálculo Vectorial

### 5.1. Campos Vectoriales

```

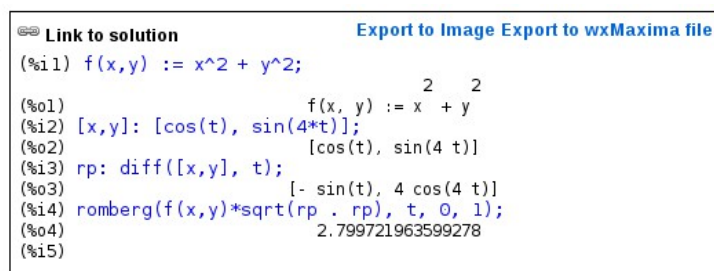
load ( draw );
coord : setify ( makelist ( k ,k , -12 ,12));
points2d : listify ( cartesian_product ( coord , coord ));
vf2d (x , y ):= vector ([ x , y ] ,[ cos ( y ) , x ]/6);
vect2 : makelist ( vf2d ( k [1] , k [2]) , k , points2d );
apply ( draw2d , append ([ color = red ] , vect2));

```



## 5.2. Integrales de Línea

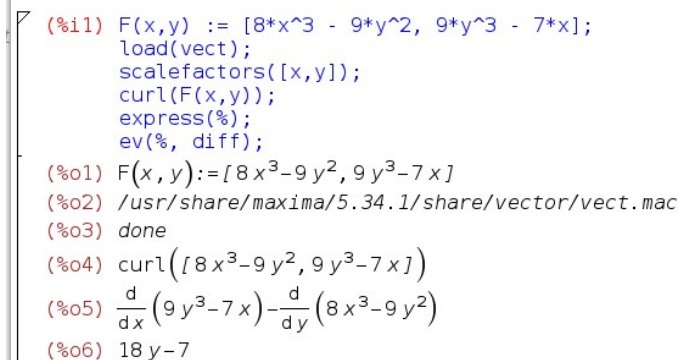
```
f(x,y) := x^2 + y^2;
[x,y]: [cos(t), sin(4*t)];
rp: diff([x,y], t);
romberg(f(x,y)*sqrt(rp . rp), t, 0, 1);
```



```
(%i1) f(x,y) := x^2 + y^2;
(%o1) f(x, y) := x^2 + y^2
(%i2) [x,y]: [cos(t), sin(4*t)];
(%o2) [cos(t), sin(4 t)]
(%i3) rp: diff([x,y], t);
(%o3) [- sin(t), 4 cos(4 t)]
(%i4) romberg(f(x,y)*sqrt(rp . rp), t, 0, 1);
(%o4) 2.799721963599278
(%i5)
```

## 5.3. Campos Vectoriales Conservativos y Potenciales Escalares

```
F(x,y) := [8*x^3 - 9*y^2, 9*y^3 - 7*x];
load(vect);
scalefactors([x,y]);
curl(F(x,y));
express(%);
ev(%, diff);
```



```
(%i1) F(x,y) := [8*x^3 - 9*y^2, 9*y^3 - 7*x];
load(vect);
scalefactors([x,y]);
curl(F(x,y));
express(%);
ev(%, diff);
(%o1) F(x, y) := [8 x^3 - 9 y^2, 9 y^3 - 7 x]
(%o2) /usr/share/maxima/5.34.1/share/vector/vect.mac
(%o3) done
(%o4) curl([8 x^3 - 9 y^2, 9 y^3 - 7 x])
(%o5) d/dx (9 y^3 - 7 x) - d/dy (8 x^3 - 9 y^2)
(%o6) 18 y - 7
```

## Referencias

- [1] *Maxima*. <http://maxima.sourceforge.net/es/>
- [2] *Multivariable Calculus with Maxima by G.Jay Kerns* <http://gkerns.people.ysu.edu/maxima/maximaintro/>