Universidad de Sonora
División de Ciencias Exactas y Naturales
Licenciatura en Física
Física computacional
2016-1

Actividad 8: Bitácora de ejercicios realizados con Maxima

Carolina Valenzuela Córdova

5 de Marzo de 2016

Índice

1. Descripción de la actividad

Para esta actividad, se nos solicitó explorar un manual de uso para Maxima, el cual ün 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."[1]
Fue necesario repetir uno de los ejemplos o ejercicios propuestos por este manual en cada sección, modificando funciones y colores de gráficas. A continuación se presentan los resultados obtenidos y una breve explicación.

2. Geometría en tres dimensiones

2.1. Vectores y álgebra lineal

En esta sección se muestra cómo con Maxima se pueden realizar operaciones con vectores como las conocemos habitualmente. Estos fueron los resultados:

```
i1) a:[1,2,3];
(%o1)                                [1, 2, 3]
(%i2) b:[2,-1,4];
(%o2)                               [2, - 1, 4]
(%i3) a+b;
(%o3)                                [3, 1, 7]
(%i4) a.b;
(%o4)                                   12
(%i5) load(vect);
(%o5)           /usr/share/maxima/5.21.1/share/vector/vect.mac
(%i6) a~b;
(%o6)                        [1, 2, 3] ~ [2, - 1, 4]
(%i7) express(a~b);
(%o7)                             [11, 2, - 5]
(%i8) sqrt(a.a);
(%o8)                               sqrt(14)
```

```
(%i9)
```

2.2. Líneas, planos y superficies cuadráticas

Aquí se graficó un hiperboloide con el comando draw3d, y se cambiaron las combinaciones de colores con el comando palette. Utilizamos una combinación de números al azar que nos resultó en colores diferentes a los mostrados en el manual. A continuación se presentan las instrucciones hechas a Maxima y la gráfica obtenida.

```
hyperboloid:x**2+y**2-z**2=1;

load(draw);

draw3d(enhanced3d= true, implicit
(hyperboloid, x,-2,2, y,-2,2, z,-1.5,1.5),palette=[29,9,20]);
```

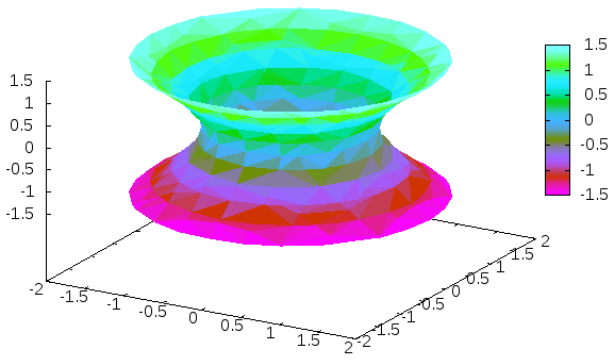
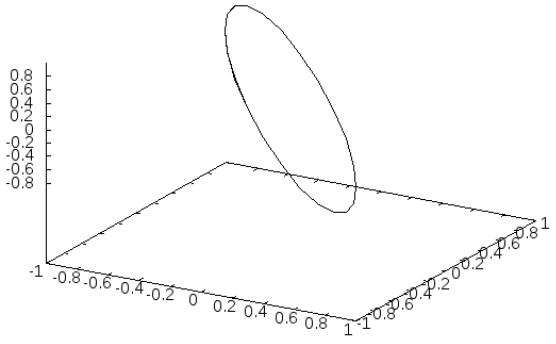


Figura 1: Hiperboloide

2.3. Funciones vectoriales evaluadas

Aquí se propuso una función, a la cual evaluamos en un punto, graficamos y calculamos su límite y su derivada. Utilizamos los comandos del manual para realizar todas estas operaciones a nuestra función.

```
contenidos...%i1) r(t):= [t, cos(t), sin(t)];
(%o1)                      r(t) := [t, cos(t), sin(t)]
(%i2) r(1);
(%o2)                           [1, cos(1), sin(1)]
(%i3) float(r(1));
(%o3)               [1.0, 0.54030230586814, 0.8414709848079]
(%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) diff(r(t),t);
(%o8)                          [1, - sin(t), cos(t)]
(%i9)
```



2.4. Longitud de arco y curvatura

Se realizó un ejemplo de curvatura propuesto por el manual y se utilizó la fórmula de curvatura. Cabe destacar que Maxima fue utilizado en este ejercicio como una herramienta, pues en realidad el software mismo no es capaz de calcular la curvatura, sino de ejecutar los pasos para calcularla. La curvatura está definida de la siguiente manera:

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

De esta forma, obtuvimos sus componentes con Maxima para calcularla:

```
r(t) := [t, cos(t), sin(t)];
(%o1)                      r(t) := [t, cos(t), sin(t)]
(%i2) rp(t) := [1 ,-sin(t), cos(t)];
(%o2)                    rp(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         2
sqrt(sin (t) + cos (t) + 1)
(%i5) trigsimp(sqrt(Tp(t) . Tp(t))/sqrt(rp(t) . rp(t)));
1
(%o5)                                    -
2
(%i6) define(kappa(t),sqrt(Tp(t) . Tp(t))/sqrt(rp(t)
. rp(t)))integrate(r(t), t);
incorrect syntax: INTEGRATE is not an infix operator
Space.Spacerp(t)))integrate(
^
(%i6) integrate(r(t), t);
```

```
2
t
(%o6)                          [--, sin(t), - cos(t)]
2
(%i7)
```

3. Funciones de varias variables

3.1. Derivadas parciales

Propusimos una función multivariable y calculamos sus derivadas parciales con el mismo comando con el cual calculamos derivadas totales, pero ahora especificando con respecto a cuál variable queríamos que Maxima operara y cuántas veces.

```
(%i1) G: x^7 * y^8;
7  8
(%o1)                                  x  y
(%i2) diff(G, x, 1, y, 2, x, 3);
3  6
(%o2)                               47040
```

Aquí se le solicitó que derivara con respecto a x 4 veces y con respecto a y dos veces.

3.2. Aproximación lineal

Se utilizó un polinomio de Taylor para aproximar una función propuesta. Fue sencillo ya que Maxima tiene un comando específico para este tipo de aproximación.

```
f(x,y) := exp(x^2) * sin(y);
(%o1)
f(x, y) := expx2siny
(%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)
```

3.3. Regla de la cadena

Se utilizó la misma función para diferenciarla mediante la regla de la cadena con Maxima. Antes de darle esta instrucción al programa, fue necesario parametrizar la función y diferenciarla con respecto a las nuevas dos variables. Maxima lo hizo de la siguiente manera:

```
f(x,y) := exp(x^2) * sin(y);
[x,y] : [s^2 * t, s * t^2];
diff(f(x,y), s);
diff(f(x,y), t);
```

```
(%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)
```

3.4. Derivadas direccionales y Gradiente

Propusimos una función, a la cual debimos convertirla en vectorial, para poder calcular su gradiente con un comando específico de Maxima para esta operación. Una vez obtenido el gradiente, se evalúa en un punto y calculamos la derivada direccional con la fórmula aprendida en el curso de Cálculo III. Se obtuvo de la siguiente manera:

```
(%i1) f(x,y) := exp(x^2) * sin(y);
2
(%o1)                        f(x, y) := exp(x ) sin(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   sin(y))
(%i5) ev(express(gdf), diff);
2            2
x            x
(%o5)                    [2 x %e   sin(y), %e   cos(y)]
(%i6) define(gdf(x,y),ev(express(gdf), diff));
2            2
x            x
(%o6)              gdf(x, y) := [2 x %e   sin(y), %e   cos(y)]
(%i7) v: [3,4];
(%o7)                                 [3, 4]
(%i8) (gdf(1,2) . v)/sqrt(v . v);
6 %e sin(2) + 4 %e cos(2)
(%o8)                       -------------------------
5
(%i9) ev((gdf(1,2) . v)/sqrt(v . v), diff);
6 %e sin(2) + 4 %e cos(2)
(%o9)                       -------------------------
5
(%i10) float(ev((gdf(1,2) . v)/sqrt(v . v), diff));
(%o10)                          2.061108499400332
(%i11)
```

3.5. Optimización y Extremos locales

Se propuso una función y se graficó. Después de ello se siguió un procedimiento de optimización que el manual propone, utilizando las derivadas parciales y una matriz a la cual se le calculó su determinante. Esto para calcular un punto crítico de la función. Obtuvimos los siguientes resultados:

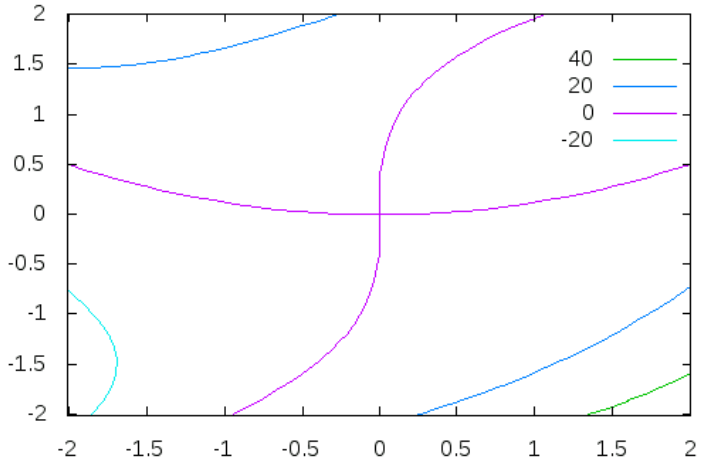
```
f(x,y) :=  x^3 + y^4 - 8 * x * y;
```

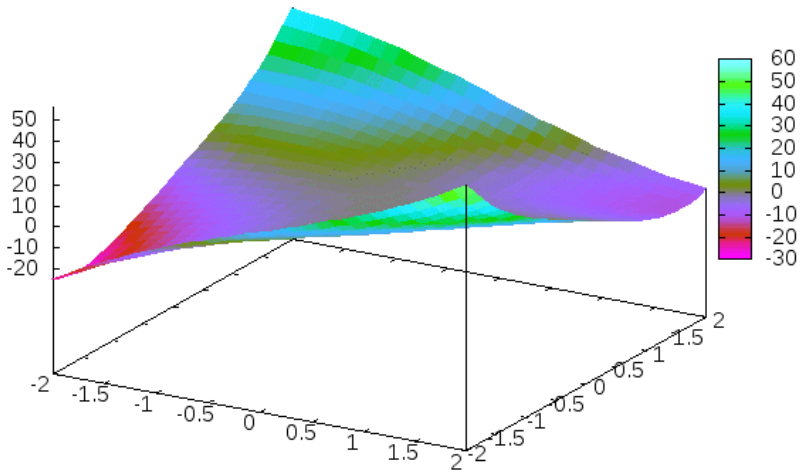
```
load(draw);
draw3d(enhanced3d = true,
explicit(f(x,y), x, -2, 2, y, -2, 2),
palette=[29,9,20]);
draw3d(explicit(f(x,y), x, -2, 2, y, -2, 2), contour = map);
fx : diff(f(x,y), x);
fy : diff(f(x,y), y);
solve([fx,fy], [x,y]);
H: hessian(f(x,y), [x,y]);
determinant(H);
subst([x = -1, y = -1], diff(fx, x));
subst([x = -1, y = -1], determinant(H));
f(-1,-1);
f(x,y) :=  x^3 + y^4 - 8 * x * y;
3    4
(%o1)                    f(x, y) := x  + 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=[29,9,20]);
(%o3)                           [gr3d(explicit)]


diff(f(x,y), x);
2
(%o5)                               3 x  - 8 y
(%i6) fy : diff(f(x,y), y);
3
(%o6)                               4 y  - 8 x
(%i7) solve([fx,fy], [x,y]);
(%o7) [[x = 2.069127516778524, y = 1.605483271375465],
[x = 1.216202589577063 %i - 1.673959255924093,
y = 0.49612156935 - 1.526905186426034 %i],
[x = - 1.216202589577063 %i - 1.673959255924093,
y = 1.526905186426032 %i + 0.49612156935],
[x = 1.967857127141327 %i + 0.63939553998052,
y = 0.94367930280978 %i - 1.298863131110227],
```

```
[x = 0.63939553998052 - 1.967857127141327 %i,
y = - 0.94367930280978 %i - 1.298863131110226], [x = 0, y = 0]]
(%i8) H: hessian(f(x,y), [x,y]);
[ 6 x   - 8  ]
(%o8)                              [            ]
[          2 ]
[ - 8  12 y  ]
(%i9) determinant(H);
2
(%o9)                               72 x 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)                                 - 8
(%i13)
```

Y las siguientes gráficas:





3.6. Multiplicadores de Lagrange

Se necesitó calcular los valores extremos de una superficie limitada por otra; para ello se propusieron dos funciones, las cuales fueron diferenciadas y se resolvió el sistema de ecuaciones diferenciales para encontrar los valores extremos. Después se evaluó la función en estos puntos para obtener el máximo y mínimo de la misma. Aquí se muestra el procedimiento:

```
(%i1) f(x,y) := x^2 + 2*y^2;
2      2
(%o1)                          f(x, y) := x  + 2 y
(%i2) g: 3*x^2 + 3*y^2;
2      2
(%o2)                                3 y  + 3 x
(%i3) eq1: diff(f(x,y), x) = h * diff(g, x);
(%o3)                                2 x = 6 h x
(%i4) eq2: diff(f(x,y), y) = h * diff(g, y);
(%o4)                                4 y = 6 h y
(%i5) eq3: g = 1;
```

```
2      2
(%o5)                           3 y  + 3 x  = 1
(%i6) solve([eq1, eq2, eq3], [x, y, h]);
1                1             1                 1
(%o6) [[x = -------, y = 0, h = -], [x = - -------, y = 0, h = -],
sqrt(3)             3          sqrt(3)             3
1         2                    1         2
[x = 0, y = -------, h = -], [x = 0, y = - -------, h = -]]
sqrt(3)      3                 sqrt(3)      3
(%i7) [f(1,0), f(-1,0), f(0,-1), f(0,1)];
(%o7)                             [1, 1, 2, 2]
(%i8)
```

4. Integración múltiple

4.1. Integrales dobles

Se propuso una función, para cual calculamos su doble integral, simplemente doblando el comando predeterminado por Maxima. Obtuvimos los siguientes resultados:

```
f(x,y) := 3*x^3 - 5*x*y;
integrate(integrate(f(x,y), y), x);
```

```
(%i1) f(x,y) := 3*x^3 - 5*x*y;
3
(%o1)                        f(x, y) := 3 x  - 5 x y
(%i2) integrate(integrate(f(x,y), y), x);
4        2  2
3 x  y   5 x  y
(%o2)                            ------ - -------
4         4
(%i3)
```

4.2. Integración con coordenadas polares

Para este ejercicio se propuso una función de dos variables, para la cual se utilizó una parametrización general en coordenadas polares. Esto lo pudimos hacer con los conocimientos adquiridos en nuestros cursos de Cálculo, y utilizamos a Maxima como una herramienta para ello.
Al ya tener la parametrización a coordenadas polares, integramos sobre esta nueva función con los límites de las nuevas variables.

```
f(x,y) := 3*x^2 + 3*y^2;
[x,y]: [r * cos(theta), r * sin(theta)];
integrate(integrate(f(x,y) * r, r, 0, 2*cos(theta)),
theta, -%pi/2,%pi/2);
```

```
(%i1) f(x,y) := 3*x^2 + 3*y^2;
2      2
(%o1)                         f(x, y) := 3 x  + 3 y
(%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, -%pi/2,%pi/2);
9 %pi
(%o3)                                    -----
2
(%i4)
```

4.3. Integrales triples

Esta sección no tuvo mayor complicación, pues solo repetimos el mismo procedimiento que para las integrales dobles, pero utilizando el comando de Maxima tres veces.

```
integrate(integrate(integrate(x^2*y*z,z,0,x+y),y,0,-x),x,0,1);

integrate(integrate(integrate(x^2*y*z,z,0,x+y),y,0,-x),x,0,1);
```

```
1
(%o1)                                  ---
168
(%i2)
```

4.4. Integrales con coordenadas cilíndricas y esféricas

El procedimiento para esta sección es muy similar al realizado para integrales con coordenadas polares, con la diferencia de que esta vez parametrizamos una función de tres variables con coordenadas cilíndricas. Cabe destacar que no se realizó un ejemplo para coordenadas esféricas porque el procedimiento es repetitivo, ya que la diferencia yace en la parametrización que usamos. Aquí se presentan los resultados:

```
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,0,3),
 r,0,2), theta,0,%pi);
```

```
(%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,0,3), r,0,2), theta,0,%pi);
(%o3)                                  24
(%i4)
```

4.5. Cambio de variable

Para esta sección se utilizó el teorema de cambio de variable que utilizamos actualmente en nuestro curso de Cálculo IV, el cual consiste en:

$$\iint_R f(x,y)dxdy = \iint_S f(g(u,v),h(u,v)) \left|\begin{matrix} d(x,y) \\ d(u,v) \end{matrix}\right| dudv$$

Utilizamos Maxima como una herramienta para calcular por separado cada uno de los componentes de este teorema. Teniendo todas las piezas pudimos integrar por medio de este método:

```
f(x,y) := x - y;
[x,y]: [u^3 - 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);
```

```
(%i1) f(x,y) := x - y;
(%o1)                          f(x, y) := x - y
(%i2) [x,y]: [u^3 - v^4, 5 * u * v];
3    4
(%o2)                          [u  - v , 5 u v]
(%i3) J: jacobian([x,y], [u,v]);
[    2       3 ]
(%o3)                          [ 3 u   - 4 v  ]
[              ]
[ 5 v    5 u   ]
(%i4) J: determinant(J);
4       3
(%o4)                            20 v  + 15 u
(%i5) integrate(integrate(f(x,y) * J, u,1,2), v,3,4);
156593765
(%o5)                             - ---------
252
(%i6)
```

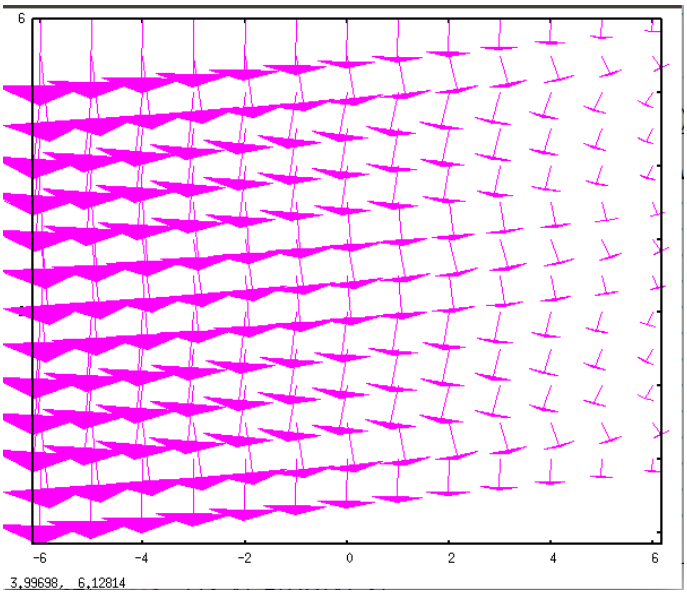
5. Cálculo Vectorial

5.1. Campos vectoriales

En esta sección se pidió graficar campos vectoriales a partir de proponer una función, la cual con ciertos comandos predeterminados en Maxima se

puede graficar como un campo vectorial. A continuación se presentan los resultados obtenidos:

```
load(draw);
F(x,y):=[cos(y^2),x-8];
coord:setify(makelist(k,k,-6,6));
points2d:listify(cartesian_product(coord,coord));
vf2d(x,y):=vector([x,y],[cos(y^2),x-8]/6);
vect2:makelist(vf2d(k[1],k[2]),k,points2d);
apply(draw2d,append([color=magenta],vect2));
```



5.2. Integrales de línea

Se propuso una función, la cual fue parametrizada y diferenciada de esta forma. Después de ello se calculó la integral de línea con las herramientas utilizadas en nuestro curso de Cálculo IV.

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

```
f(x,y) := x^2 + y^2;
2    2
(%o1)                          f(x, y) := x  + y
(%i2) [x,y]: [cos(t), sin(2*t)];
(%o2)                          [cos(t), sin(2 t)]
(%i3) rp: diff([x,y], t);
(%o3)                        [- sin(t), 2 cos(2 t)]
(%i4) romberg(f(x,y)*sqrt(rp . rp), t, 0, 1);
(%o4)                           1.635879048260743
(%i5)
```

5.3. Campos vectoriales conservativos

Esta parte del manual nos ayuda a comprobar que un campo vectorial es conservativo. Esto aún no se aborda en nuestro curso de Cálculo IV, pero fue interesante intentar entender cómo funcionan estos comandos. Aquí se presentan los resultados obtenidos.

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


F(x,y) := [x^3 - 3*y^2, 7*y^3 -x];
3      2     3
(%o1)                  F(x, y) := [x  - 3 y , 7 y  - x]
(%i2) load(vect);
(%o2)           /usr/share/maxima/5.21.1/share/vector/vect.mac
(%i3)  scalefactors([x,y]);
(%o3)                                  done
(%i4) curl(F(x,y));
3      2     3
(%o4)                     curl [x  - 3 y , 7 y  - x]
```

```
(%i5) express(curl(F(x,y)));
d      3        d    3      2
(%o5)                   -- (7 y  - x) - -- (x  - 3 y )
dx              dy
(%i6) ev(express(curl(F(x,y))), diff);
(%o6)                                6 y - 1
(%i7)
```

Referencias

[1] Maxima, un sistema de álgebra computacional, http://maxima.sourceforge.net/es/

[2] Multivariable Calculus with Maxima, http://gkerns.people.ysu.edu/maxima/maximaintro/