

Actividad #8 (Cómputo Simbólico con Maxima)

Instructor: Carlos Lizárraga Celaya

Student: Antonio Cota Rodríguez

Introducción

wxMaxima

El sistema de álgebra computacional Maxima es un motor de cálculo simbólico escrito en lenguaje Lisp publicado bajo licencia GNU GPL.

Cuenta con un amplio conjunto de funciones para hacer manipulación simbólica de polinomios, matrices, funciones racionales, integración, derivación, manejo de gráficos en 2D y 3D, manejo de números de coma flotante muy grandes, expansión en series de potencias y de Fourier, entre otras funcionalidades. Además tiene un depurador a nivel de fuente para el código de Maxima.

Maxima está basado en el sistema original de Macsyma desarrollado por MIT en los años 70. Es bastante fiable, tiene un buen recolector de basura, por lo que no desperdicia memoria. Viene con cientos de auto pruebas (test-suite).

Actividad

Durante toda la actividad se mostrarán imágenes para evidenciar el trabajo hecho, cada imagen cuenta con su respectiva descripción de lo que se realizó paso por paso. La dinámica es la siguiente; se mostrará la imagen y si el resultado dio una gráfica se presentará abajo del código.

Capítulo 2

2.1 Vectores y Álgebra lineal

```
(%i2) a: [3, 5, 7];
(a) [3, 5, 7]
Definimos un vector 'a' y un vector 'b'.

(%i3) b: [-8, 5, 0];
(b) [-8, 5, 0]

(%i4) (-2)*a;
(%o4) [-6, -10, -14]
Multiplicación por un escalar

(%i5) 2*a + b;
(%o5) [-2, 15, 14]
Adición de vectores
```

FIGURE 1 – Suma y producto por un escalar

```
(%i4) a.b;
(%o4) 1
Producto escalar

(%i5) load(vect);
(%o5) C:/Program Files/Maxima-sbcl-5.38.0/share/maxima/5.38.0/share/vector/vect.m

(%i6) a~b;
(%o6) [-8, 5, 0]~[3, 5, 7]
Producto cruz

(%i7) express(%);
(%o7) [-35, -56, 55]
Producto cruz resultado

(%i8) sqrt(a.a);
(%o8) sqrt(83)
Norma de un vector
```

FIGURE 2 – Producto cruz, punto y norma de un vector

```

(%i9) (a.b)/(a.a)*a;
(%o9)  $\left[ \frac{3}{83}, \frac{5}{83}, \frac{7}{83} \right]$  Proyección escalar

(%i10) c:[10,2,-3];
(%o10)  $[10, 2, -3]$  Definimos un nuevo vector 'c'

(%i11) a.(b~c);
(%o11)  $[3, 5, 7] \cdot [-8, 5, 0] \sim [10, 2, -3]$  Triple producto escalar

(%i12) express(%);
(%o12) -627 Resultado del TPE

```

FIGURE 3 – Proyección escalar y triple producto escalar

2.2 Lineas, planos y superficies cuádricas

```

(%i1) plane1: 4*x -2*y +9*z = 0;
(%o1)  $9z - 2y + 4x = 0$  Definimos un plano

(%i3) draw3d(enhanced3d = true, implicit(plane1, x,-5,5, y,-5,5, z,-7,7));
(%o3)  $[gr3d(implicit)]$ 

(%i4) Ellipse;
(%o4) Ellipse

(%i5) ellipsoid1: x^2/2 + 2*y^2 + z^2 = 5;
(%o5)  $z^2 + 2y^2 + \frac{x^2}{2} = 5$  Definimos un elipsoide

(%i6) draw3d(enhanced3d = true, implicit(ellipsoid1, x,-4,4, y,-3,3, z, -3,3));

```

FIGURE 4 – Ecuación de un plano y una elipsoide

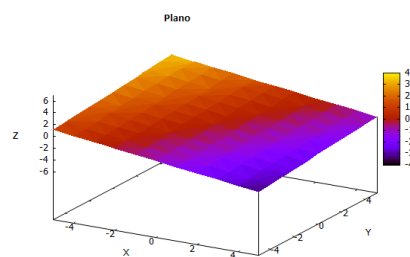


FIGURE 5 – Gráfica del plano

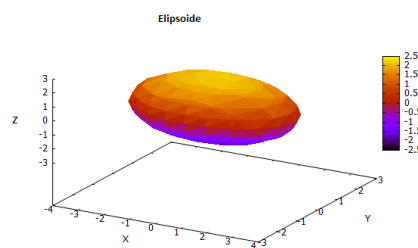


FIGURE 6 – Gráfica de la elipsoide

```

(%i8) a: [2,3,4];          Definimos un vector 'a'
(a)   [2,3,4]

(%i9) b: [-3,3,5];        Vector 'b'
(b)   [-3,3,5]

(%i10) c: [1,1,1];        Vector 'c'
(c)   [1,1,1]

(%i11) ab: b - a;          Distancia de entre 'a' y 'b'
(ab)  [-5,0,1]

(%i12) ac: c - a;          Distancia entre 'a' y 'c'
(ac)  [-1,-2,-3]

(%i13) n: express(ab ~ ac); Vector normal al plano
(n)   [2,-16,10]

(%i14) r: [x,y,z];        Vector 'r'
(r)   [x,y,z]

(%i15) r0: a;              Punto inicial
(r0)  [2,3,4]

--> plane: n.r = n.r0;      Ecuación del plano
(plane) 10 x-16 y+2 z=-4

(%i17) draw3d(enhanced3d = true,implicit(plane,x,-4,4,y,-4,4,z,-4,4));

```

FIGURE 7 – Ecuación de un plano a partir de tres puntos

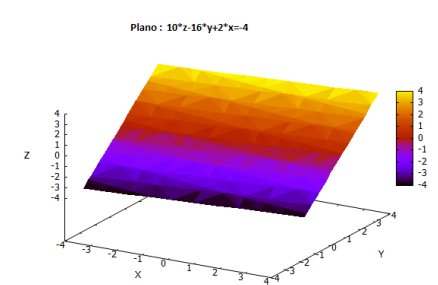


FIGURE 8 – Plano dado 3 puntos

```

(%i19) Cono: x^2 + y^2 = 3*z^2;          Definimos un cono y lo graficamos
(Cono)  y^2+x^2=3 z^2

(%i20) draw3d(enhanced3d = true,implicit(Cono, x,-1,1, y,-1,1, z,-0.5,0.5));

```

FIGURE 9 – Cono en Maxima

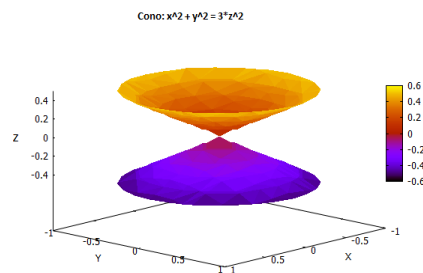


FIGURE 10 – Gráfica del cono

2.3 Funciones vectoriales

```

(%i8) r(t) := [2*t, cos(t), sin(t)];
(%o8) r(t) := [2 t, cos(t), sin(t)]
Definimos una función vectorial

(%i9) r(0);
(%o9) [0, 1, 0]
Evaluamos en 0 y 1

(%i10) r(1);
(%o10) [2, cos(1), sin(1)]

(%i11) float(%);
(%o11) [2.0, 0.5403023058681398, 0.8414709848078965]
Resultado numérico

Graficamos una función vectorial

(%i12) draw3d(parametric(sin(2*t), cos(4*t), 2*sin(t), t, -5, 5));

```

FIGURE 11 – Función vectorial evaluada

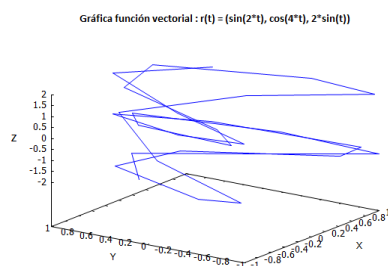


FIGURE 12 – Gráfica de la función vectorial

```

(%i1) r(t) := [3*t, cos(t), sin(t)];
(%o1) r(t) := [3 t, cos(t), sin(t)]
Definimos un vector

(%i2) diff(r(t), t);
(%o2) [3, -sin(t), cos(t)]
Derivamos

```

FIGURE 13 – Derivada de la función vectorial

```

(%i3) load(eigen);                                Cargamos paquete eigen
(%o3) C:/Program Files/Maxima-sbcl-5.38.0/share/maxima/5.38.0/share/matrix/eig

(%i4) uvect(r(t));                                Lo hacemos unitario
(%o4) 
$$\begin{bmatrix} \frac{3t}{\sqrt{\sin(t)^2 + \cos(t)^2 + 9t^2}}, \frac{\cos(t)}{\sqrt{\sin(t)^2 + \cos(t)^2 + 9t^2}}, \\ \frac{\sin(t)}{\sqrt{\sin(t)^2 + \cos(t)^2 + 9t^2}} \end{bmatrix}$$


(%i5) trigsimp(%);                                Simplificamos
(%o5) 
$$\begin{bmatrix} \frac{3t}{\sqrt{9t^2+1}}, \frac{\cos(t)}{\sqrt{9t^2+1}}, \frac{\sin(t)}{\sqrt{9t^2+1}} \end{bmatrix}$$


(%i6) define(T(t), %);
(%o6) 
$$T(t) := \begin{bmatrix} \frac{3t}{\sqrt{9t^2+1}}, \frac{\cos(t)}{\sqrt{9t^2+1}}, \frac{\sin(t)}{\sqrt{9t^2+1}} \end{bmatrix}$$


(%i7) define(Tp(t), diff(T(t), t));
(%o7) 
$$Tp(t) := \begin{bmatrix} \frac{3}{\sqrt{9t^2+1}} - \frac{27t^2}{(9t^2+1)^{3/2}}, -\frac{\sin(t)}{\sqrt{9t^2+1}} - \frac{9t\cos(t)}{(9t^2+1)^{3/2}}, \\ \frac{\cos(t)}{\sqrt{9t^2+1}} - \frac{9t\sin(t)}{(9t^2+1)^{3/2}} \end{bmatrix}$$


(%i8) uvect(Tp(t));                                Normalizamos
(%o8) 
$$\begin{bmatrix} \frac{\sqrt{729t^6+243t^4+27t^2+1} \left( \frac{3}{\sqrt{9t^2+1}} - \frac{27t^2}{(9t^2+1)^{3/2}} \right)}{\sqrt{(81t^4+99t^2+1)\sin(t)^2 + (81t^4+99t^2+1)\cos(t)^2 + 9}}, \\ \frac{\sqrt{729t^6+243t^4+27t^2+1} \left( -\frac{\sin(t)}{\sqrt{9t^2+1}} - \frac{9t\cos(t)}{(9t^2+1)^{3/2}} \right)}{\sqrt{(81t^4+99t^2+1)\sin(t)^2 + (81t^4+99t^2+1)\cos(t)^2 + 9}}, \\ \frac{\sqrt{729t^6+243t^4+27t^2+1} \left( \frac{\cos(t)}{\sqrt{9t^2+1}} - \frac{9t\sin(t)}{(9t^2+1)^{3/2}} \right)}{\sqrt{(81t^4+99t^2+1)\sin(t)^2 + (81t^4+99t^2+1)\cos(t)^2 + 9}} \end{bmatrix}$$


(%i9) trigsimp(%);                                Simplificamos
(%o9) 
$$\begin{bmatrix} \frac{3\sqrt{9t^2+1}\sqrt{729t^6+243t^4+27t^2+1}}{(81t^4+18t^2+1)\sqrt{81t^4+99t^2+10}}, - \\ \frac{\sqrt{9t^2+1}\sqrt{729t^6+243t^4+27t^2+1}((9t^2+1)\sin(t) + 9t\cos(t))}{(81t^4+18t^2+1)\sqrt{81t^4+99t^2+10}}, - \\ \frac{\sqrt{9t^2+1}\sqrt{729t^6+243t^4+27t^2+1}(9t\sin(t) + (-9t^2-1)\cos(t))}{(81t^4+18t^2+1)\sqrt{81t^4+99t^2+10}} \end{bmatrix}$$


```

FIGURE 14 – Vector tangente unitario

```

(%i10) define(N(t), %);
(%o10) N(t) := [ 3*sqrt(9*t^2+1)*sqrt(729*t^6+243*t^4+27*t^2+1) /
(81*t^4+18*t^2+1)*sqrt(81*t^4+99*t^2+10), -
sqrt(9*t^2+1)*sqrt(729*t^6+243*t^4+27*t^2+1)*(9*t^2+1)*sin(t)+9*t*cos(t) /
(81*t^4+18*t^2+1)*sqrt(81*t^4+99*t^2+10), -
sqrt(9*t^2+1)*sqrt(729*t^6+243*t^4+27*t^2+1)*(9*t*sin(t)+(-9*t^2-1)*cos(t)) /
(81*t^4+18*t^2+1)*sqrt(81*t^4+99*t^2+10) ]

Definimos el vector normal

(%i11) load(vect);
(%o11) C:/Program Files/Maxima-sbcl-5.38.0/share/maxima/5.38.0/share/vector/vect

Paquete para vectores

(%i12) express(T(t) ~ N(t));
(%o12) [ 729*t^6+243*t^4+27*t^2+1*sin(t)*(9*t^2+1)*sin(t)+9*t*cos(t) /
(81*t^4+18*t^2+1)*sqrt(81*t^4+99*t^2+10) -
sqrt(729*t^6+243*t^4+27*t^2+1)*cos(t)*(9*t*sin(t)+(-9*t^2-1)*cos(t)) /
(81*t^4+18*t^2+1)*sqrt(81*t^4+99*t^2+10),
3*t*sqrt(729*t^6+243*t^4+27*t^2+1)*(9*t*sin(t)+(-9*t^2-1)*cos(t)) /
(81*t^4+18*t^2+1)*sqrt(81*t^4+99*t^2+10) +
3*sqrt(729*t^6+243*t^4+27*t^2+1)*sin(t) /
(81*t^4+18*t^2+1)*sqrt(81*t^4+99*t^2+10),
3*t*sqrt(729*t^6+243*t^4+27*t^2+1)*(9*t^2+1)*sin(t)+9*t*cos(t) /
(81*t^4+18*t^2+1)*sqrt(81*t^4+99*t^2+10) -
3*sqrt(729*t^6+243*t^4+27*t^2+1)*cos(t) /
(81*t^4+18*t^2+1)*sqrt(81*t^4+99*t^2+10) ]

Producto cruz de T(t) y N(t)

(%i13) trigsimp(%);
(%o13) [ sqrt(81*t^4+99*t^2+10)*sqrt(729*t^6+243*t^4+27*t^2+1) /
729*t^6+972*t^4+189*t^2+10,
sqrt(81*t^4+99*t^2+10)*sqrt(729*t^6+243*t^4+27*t^2+1)*(3*sin(t)-3*t*cos(t)) /
729*t^6+972*t^4+189*t^2+10,
sqrt(81*t^4+99*t^2+10)*sqrt(729*t^6+243*t^4+27*t^2+1)*(3*t*sin(t)+3*cos(t)) /
729*t^6+972*t^4+189*t^2+10 ]

Simplificamos

Y lo definimos como nuestro vector binormal

(%i14) define(B(t), %);
(%o14) B(t) := [ sqrt(81*t^4+99*t^2+10)*sqrt(729*t^6+243*t^4+27*t^2+1) /
729*t^6+972*t^4+189*t^2+10,
sqrt(81*t^4+99*t^2+10)*sqrt(729*t^6+243*t^4+27*t^2+1)*(3*sin(t)-3*t*cos(t)) /
729*t^6+972*t^4+189*t^2+10,
sqrt(81*t^4+99*t^2+10)*sqrt(729*t^6+243*t^4+27*t^2+1)*(3*t*sin(t)+3*cos(t)) /
729*t^6+972*t^4+189*t^2+10 ]

Evaluamos en 1

(%i15) float(B(1));
(%o15) [0.99941573228705618, 0.9973785004022756, 0.9510016005605510 ]

```

FIGURE 15 – Vector binormal

Capítulo 3

3.1 Funciones de varias variables

```

(%i1) f(x,y) := (2*x^3 - y^2)^2;
(%o1) f(x,y) := (2*x^3 - y^2)^2
Definimos una función f(x,y)

(%i2) load(draw);
;; loading #P"C:/Users/Tony/maxima/binary/5_38_0/sbcl/1_3_4/share/draw/draw3d
;; loading #P"C:/Users/Tony/maxima/binary/5_38_0/sbcl/1_3_4/share/draw/draw3d
;; loading #P"C:/Users/Tony/maxima/binary/5_38_0/sbcl/1_3_4/share/draw/draw3d
;; loading #P"C:/Users/Tony/maxima/binary/5_38_0/sbcl/1_3_4/share/draw/draw3d
(%o2) C:/Program Files/Maxima-sbcl-5.38.0/share/maxima/5.38.0/share/draw/draw3d

(%i3) draw3d(explicit(f(x,y), x, -3, 3, y, -3, 3));
Graficamos

```

FIGURE 16 – Función de varias variables

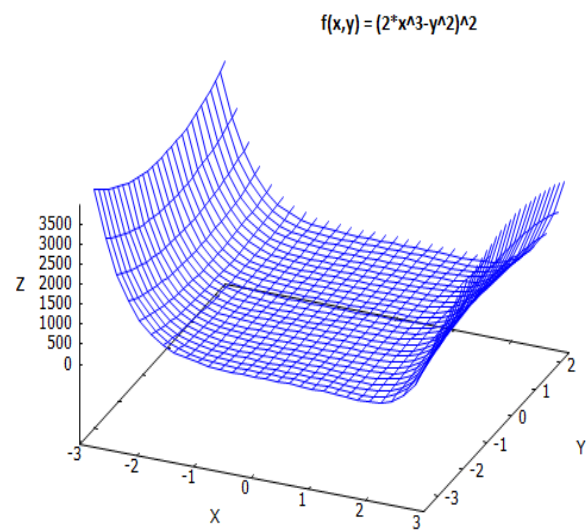


FIGURE 17 – Gráfica de la función

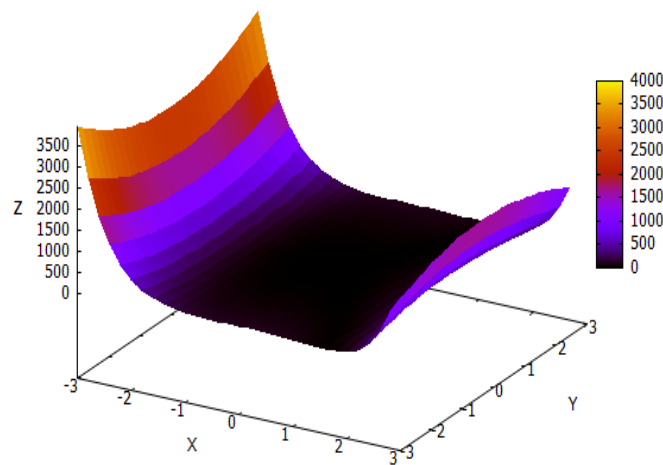


FIGURE 18 – Gráfica de la función 'enhanced'

<pre>(%i8) diff(f(x,y), x);</pre>	Derivada parcial con respecto a 'x'
<pre>(%o8) 12 x^2 (2 x^3 - y^2)</pre>	
<pre>(%i9) diff(diff(f(x,y), x), x);</pre>	Segunda derivada parcial con respecto a 'x'
<pre>(%o9) 24 x (2 x^3 - y^2) + 72 x^4</pre>	
<pre>(%i10) diff(diff(f(x,y), x), y);</pre>	Segunda derivada parcial primero con respecto a 'x' y después con respecto a 'y'
<pre>(%o10) -24 x^2 y</pre>	

FIGURE 19 – Derivadas parciales de una función de varias variables

3.2 Aproximaciones lineales y diferenciales

```
(%i11) f(x,y) := sin(x)*cos(y);          Definimos una función f(x,y)
(%o11) f(x,y) := sin(x) cos(y)

(%i12) taylor(f(x,y), [x,y], [1,2], 1);  Aplicamos Taylor
(%o12)/T/ sin(1) cos(2) + (cos(1) cos(2) (x-1) - sin(1) sin(2) (y-2) ) + ...

(%i13) diff(f(x,y));
(%o13) cos(x) cos(y) del (x) -sin(x) sin(y) del (y)  Obtenemos los diferenciales
```

FIGURE 20 – Aproximación lineal y diferenciales de una función de varias variables

3.3 Regla de la cadena y Derivación implícita

```
(%i27) F: x^2 + y^3 - z^3 + x*y*z - 1/z;  Definimos una función
(F)   -z^3 + x y z - 1/z + y^3 + x^2      F(x,y,z)

(%i28) Fx: diff(F, x);                    Derivamos implícitamente con
(Fx)   y z + 2 x                          respecto a 'x', 'y' y 'z'

(%i29) Fy: diff(F, y);
(Fy)   x z + 3 y^2

(%i30) Fz: diff(F, z);
(Fz)   -3 z^2 + 1/z^2 + x y

(%i31) [-Fx/Fy, -Fy/Fz];
(%o31) [ -y z - 2 x / (x z + 3 y^2), -x z - 3 y^2 / (-3 z^2 + 1/z^2 + x y) ]
```

FIGURE 21 – Aplicación de la regla de la cadena a una función

```
(%i27) F: x^2 + y^3 - z^3 + x*y*z - 1/z;  Definimos una función
(F)   -z^3 + x y z - 1/z + y^3 + x^2      F(x,y,z)

(%i28) Fx: diff(F, x);                    Derivamos implícitamente con
(Fx)   y z + 2 x                          respecto a 'x', 'y' y 'z'

(%i29) Fy: diff(F, y);
(Fy)   x z + 3 y^2

(%i30) Fz: diff(F, z);
(Fz)   -3 z^2 + 1/z^2 + x y

(%i31) [-Fx/Fy, -Fy/Fz];
(%o31) [ -y z - 2 x / (x z + 3 y^2), -x z - 3 y^2 / (-3 z^2 + 1/z^2 + x y) ]
```

FIGURE 22 – Derivadas implícitas de una función

3.4 Derivadas direccionales y Gradiente


```

(%i32) f(x,y) := tan(x+y)*tan(x*y);           Definimos una función f(x,y)
(%o32) f(x,y) := tan(x+y) tan(x y)

(%i33) load(vect);
(%o33)

C:/Program Files/Maxima-sbcl-5.38.0/share/maxima/5.38.0/share/vector/vect

(%i34) scalefactors([x,y]);                 Factores escalares
(%o34) done

(%i35) gdf: grad(f(x,y));
(gdf) grad(tan(x y) tan(y+x))              Calculamos el gradiente

(%i36) ev(express(gdf, diff));
(%o36) [y sec(x y)^2 tan(y+x) + tan(x y) sec(y+x)^2, x sec(x y)^2
tan(y+x) + tan(x y) sec(y+x)^2]

(%i37) define(gdf(x,y), %);
(%o37) gdf(x,y) := [y sec(x y)^2 tan(y+x) + tan(x y) sec(y+x)^2, x
sec(x y)^2 tan(y+x) + tan(x y) sec(y+x)^2]

(%i38) v: [3,4];                           Evaluamos
(%o38) [3,4]

(%i39) (gdf(1,2) . v)/sqrt(v . v);
(%o39)
3 (2 sec(2)^2 tan(3) + tan(2) sec(3)^2) + 4 (sec(2)^2 tan(3) + tan(2) sec(3)^2)
5

```

FIGURE 23 – Gradiente y Derivada direccional

3.5 Optimización y extremos locales

```

(%i51) f(x,y) := 3*x - 2*y^2 + 3*x^2 + 4*y;   Definimos una función
(%o51) f(x,y) := 3 x - 2 y^2 + 3 x^2 + 4 y    f(x,y)

(%i42) load(draw);                           Paquete draw
(%o42)

C:/Program Files/Maxima-sbcl-5.38.0/share/maxima/5.38.0/share/draw/draw.

(%i52) draw3d(enhanced3d = true, explicit(f(x,y), x, -20, 20, y, -20, 20),
(%o52) [gr3d(explicit) ])                    Graficamos

(%i56) draw3d(explicit(f(x,y), x, -20, 20, y, -20, 20),
contour = map);                             Grafica de contorno
(%o56) [gr3d(explicit) ]

(%i57) fx : diff(f(x,y), x);
(fx) 6 x + 3                               Parcial con respecto a 'x'

(%i58) fy : diff(f(x,y), y);
(fy) 4 - 4 y                               Parcial con respecto a 'y'

(%i59) solve([fx,fy], [x,y]);               Punto critico
(%o59) [[x = -1/2, y = 1]]

(%i60) H: hessian(f(x,y), [x,y]);           Hessiano
(H)
6  0
0 -4

(%i61) determinant(H);                     Determinante del Hessiano
(%o61) -24

```

FIGURE 24 – Optimización de una función

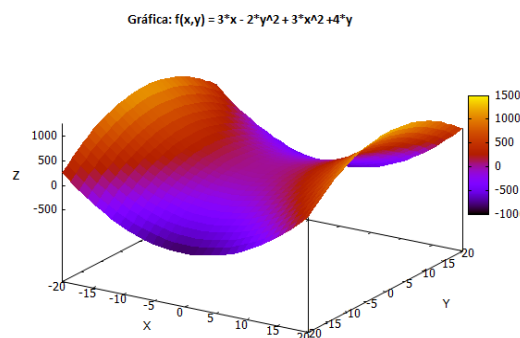


FIGURE 25 – Grafica de la función

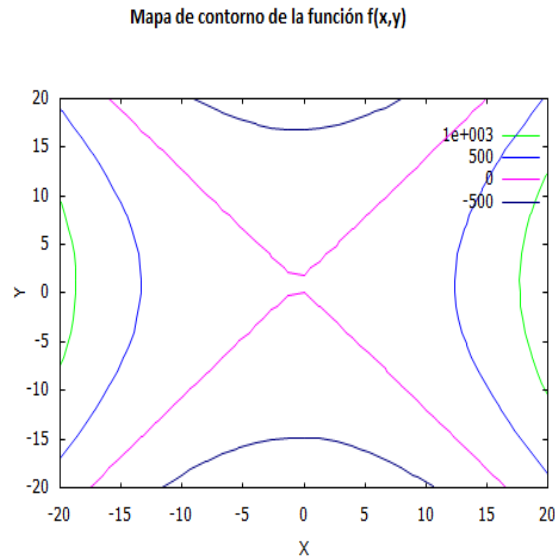


FIGURE 26 – Mapa contorno

Capítulo 4

4.1 Integrales Dobles

```
(%i62) f(x,y) := x^2 + x^3 + 2*y^2;
(%o62) f(x,y) := x^2 + x^3 + 2*y^2      Definimos la función

(%i63) integrate(integrate(f(x,y), y), x);
(%o63) x y^2 + x^4 y / 4 + x^3 y / 3      Integración indefinida

(%i64) integrate(integrate(f(x,y), y, x^(1/2), 2 - x), x, 0, 1);
(%o64) 329 / 120      Integración definida
```

FIGURE 27 – Integración Doble en Maxima

4.2 Integración en coordenadas polares

```
(%i65) f(x,y) := x^3 + y^3;
(%o65) f(x,y) := x^3 + y^3      Definimos la función

(%i66) [x,y] := [r*cos(theta), r*sin(theta)];
(%o66) [r*cos(theta), r*sin(theta)]      Cambio a coordenadas polares

(%i67) integrate(integrate(f(x,y) * r, r, 0, 2*cos(theta)), theta, -pi/2, pi/2);
(%o67) 7*pi / 4      Integración definida
```

FIGURE 28 – Integración en coordenadas polares

4.3 Integrales Triples

```
(%i4) integrate(integrate(integrate(x^2*y*z, z, 0, x+y), y, 0, -x), x, 0, 1);
(%o4) 1 / 168      Integración triple
```

FIGURE 29 – Integración triple

4.4 Integrales en coordenadas cilíndricas y esféricas

```

(%i5) f(x,y,z) := x*y*z;
(%o5) f(x,y,z) := x y z
Definimos una función f(x,y,z)

(%i6) [x,y,z] : [r*cos(theta), r*sin(theta), z];
(%o6) [r*cos(theta), r*sin(theta), z]
Realizamos un cambio de coordenadas (de
cartesianas a cilíndricas)

(%i7) integrate(integrate(integrate(f(x,y,z)*r, z,0,3), r,0,2), theta,0,%pi);
(%o7) 0
Integración definida

(%i8) kill(f,x,y,z);
(%o8) done
Eliminamos las variables

(%i9) f(x,y,z) := x*y*z;
(%o9) f(x,y,z) := x y z
Definimos nuevamente la función

(%i10) [x,y,z] : [rho*sin(phi)*cos(theta), rho*sin(phi)*sin(theta), rho*cos(theta)];
(%o10) [sin(phi)*rho*cos(theta), sin(phi)*rho*sin(theta), rho*cos(phi)]
Cambio de coordenadas a esféricas

(%i11) integrate(integrate(integrate(f(x,y,z)*rho^2*sin(phi), rho,0,1), theta,0,%pi),
phi,0,%pi/2);
(%o11) 2/27
Integración definida

```

FIGURE 30 – Integración en coordenadas cilíndricas y esféricas

4.5 Cambio de variables

```

(%i13) f(x,y) := x^2 + y^2;
(%o13) f(x,y) := x^2 + y^2
Definimos una función f(x,y)

(%i14) [x,y] : [u^2 - v^3, 5 * u*u * v];
(%o14) [u^2 - v^3, 5 u^2 v]
Realizamos un cambio de
variable cualquiera

(%i15) J: jacobian([x,y], [u,v]);
(%o15)
(J)
2 u   -3 v^2
15 u^2 v   5 u^3
Obtenemos el jacobiano de la
transformación

(%i16) J: determinant(J);
(%o16) 45 u^2 v^3 + 10 u^4
Calculamos el determinante del
jacobiano

(%i17) integrate(integrate(f(x,y) * J, u,1,2), v,3,4);
(%o17) 127020766415/2772
Integramos con el cambio de coordenadas

```

FIGURE 31 – Jacobiano de la transformación

Capítulo 5

5.1 Campos vectoriales

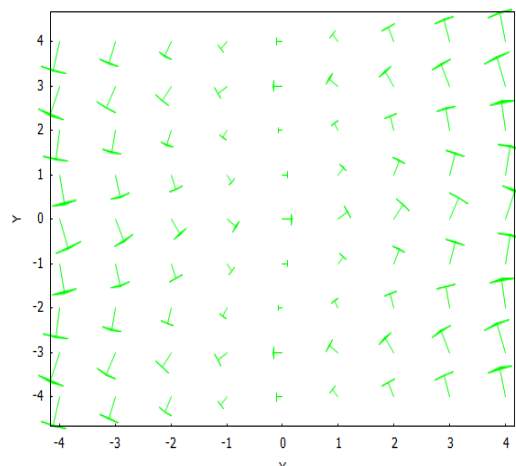


FIGURE 32 – Campo vectorial en 2 dimensiones

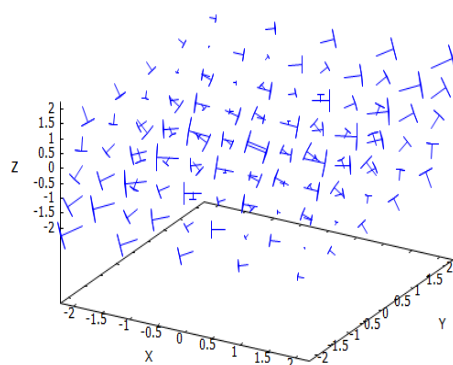


FIGURE 33 – Campo vectorial en 3 dimensiones

5.2 Integrales de Linea

```

(%i1) f(x,y) := x^3 - y^3;          Función
(%o1) f(x,y) := x^3 - y^3

(%i2) [x,y]: [cos(t), sin(2*t)];   Cambio de variable
(%o2) [cos(t), sin(2 t)]

(%i3) rp: diff([x,y], t);          Derivada
(%o3) [-sin(t), 2 cos(2 t)]

(%i4) romberg(f(x,y)*sqrt(rp . rp), t, 0, 1);
(%o4) 0.4361387938099539          Integral de línea de 0 a 1

```

FIGURE 34 – Integral de Linea

```

(%i5) F(x,y,z) := [x*y*z, x^2, y^2]; F(x,y,z)
(%o5) F(x,y,z) := [x y z, x^2, y^2]

(%i6) [x,y,z]: [t^2, t^3, t^4];   Cambio de variable
(%o6) [t^2, t^3, t^4]

(%i7) romberg(F(x,y,z) . diff([x,y,z], t), t, 0, 1);
(%o7) 1.01038961053072          Integral del campo vectorial

```

FIGURE 35 – Integral de Linea más general

5.3 Campos vectoriales conservativos

```

(%i7) F(x,y) := [x^3 + 5*y, 5*y^3 + 5*x];   Se define la función
(%o7) F(x,y) := [x^3+5 y, 5 y^3+5 x]

(%i8) ev(express(curl(F(x,y))), diff);      Igual a cero, por tanto el campo es
(%o8) 0                                       conservativo

```

FIGURE 36 – Campo vectorial conservativo

```

(%i1) F(x,y) := [4*x^3 - 5*y^2, 5*y^3 - 3*x];      Se define la función
(%o1) F(x,y) := [4 x^3-5 y^2, 5 y^3-3 x]

(%i2) load(vect);                                Se carga el paquete
(%o2) C:/Program Files/Maxima-sbcl-5.38.0/share/maxima/5.38.0/share/vector/vect.mac

(%i3) scalefactors([x,y]);                        Factores escalares
(%o3) done

(%i4) curl(F(x,y));                              Calculamos el rotacional de F
(%o4) curl ([4 x^3-5 y^2, 5 y^3-3 x])

(%i5) express(%);                                Resultado
(%o5)  $\frac{d}{dx} (5 y^3 - 3 x) - \frac{d}{dy} (4 x^3 - 5 y^2)$ 

(%i6) ev(% , diff);                              Diferente de cero, por tanto el campo es no conservativo
(%o6) 10 y-3

```

FIGURE 37 – Campo Vectorial No conservativo