1.1 Vectores geométricos

1 - 1.1.7 Sintáxis Básica

Haremos algunos cálculos sencillos para ir calentando.

```
(%i1) log(20);
(%o1) log(20)
```

Es probable que necesitemos el valor numérico de log(20).

```
(%i2) log(20),numer;
(%o2) 2.995732273553991
```

```
(%i3) 420/16000;
(%o3) 21/800
```

```
(%i4) 420/16000,numer;
(%o4) 0.02625
```

Podemos utilizar la función float para el mismo resultado

```
(%i6) float(log(20)); float(420/16000);
(%o5) 2.995732273553991
(%o6) 0.02625
```

El programa contiene una gran cantidad de funciones matemáticas básicas internas y entre ellas las trigonométricas:

```
(%i9) sin(%pi/3);cos(%pi/3);tan(%pi/3);
(%o7) sqrt(3)/2
(%o8) 1/2
(%o9) sqrt(3)
```

Ahora recurriremos a una de las facilidades que nos ofrece el programa para agrupar objetos matemáticos: las listas. Las listas se escriben entre corchetes y los objetos de la lista separados con comas.

```
(%i10) [sin(%pi/3),cos(%pi/3),tan(%pi/3)];
```

$$(\%o10)\quad \left[\frac{\sqrt{3}}{2},\frac{1}{2},\sqrt{3}\right]$$

```
(%i11) [sin(%pi/3),cos(%pi/3),tan(%pi/3)],numer;
(%o11) [0.8660254037844386,0.5000000000000001,1.732050807568877]
```

Cuando necesitemos generar una lista por medio de alguna regla específica o formula usamos la función makelist. La sintaxis es la siguiente:

```
(%i12) makelist(exp(t·x),t,1,10);
```

$$(\%o12)\quad \left[e^{x},e^{2\,x},e^{3\,x},e^{4\,x},e^{5\,x},e^{6\,x},e^{7\,x},e^{8\,x},e^{9\,x},e^{10\,x}\right]$$

Podemos también aplicar una función a cada elemento de la lista, en este caso a cada elemento le aplicaremos la función ln(x). Para tal fin utilizaremos el comando map.

```
(%i13) map(log,(makelist(exp(t·x),t,1,10)));
(%o13) [x,2 x,3 x,4 x,5 x,6 x,7 x,8 x,9 x,10 x]
```

Si queremos, por ejemplo, sumar todos los elementos de la lista anterior utilizamos la función apply con la operación que queremos realizar. Aquí aprovecharemos para utilizar un atajo muy práctico que consiste en el uso del símbolo %, que toma la última salida del programa para ser usado en la instrucción siguiente, de esta manera nos evitamos volver a escribir toda la instrucción anterior. La sintaxis para todo esto es:

```
(%i14) apply("+",%);
(%o14) 55 x
```

Podemos asignarle a una lista el nombre de una variable

```
(%i15) L: [sin(%pi/3),log(3),sqrt(2),abs(x),exp(x^2)];
```

$$(L)\quad \left[\frac{\sqrt{3}}{2},\log(3),\sqrt{2},|x|,e^{x^{2}}\right]$$

De manera que para aislar elementos de una lista escribimos:

(%i17) `L[1]; L[4];`

(%o16) $\frac{\sqrt{3}}{2}$

(%o17) $|x|$

Esto nos permite operar con sus elementos.

(%i18) `L[2]·(L[1] + L[4])/L[5];`

(%o18) $\log(3)\,e^{-x^2}\left(|x|+\frac{\sqrt{3}}{2}\right)$

La primera utilidad que le podemos dar a las listas es que nos permite definir vectores. Si queremos que el programa interprete los vectores a = (a1,a2,a3), b = (b1,b2,b3), c = (c1,c2,c3), v0 = (0,0,0), los podemos escribir como listas:

(%i22) `a:[a1,a2,a3]; b:[b1,b2,b3];c:[c1,c2,c3]; v0:[0,0,0];`

(a) *[a1,a2,a3]*

(b) *[b1,b2,b3]*

(c) *[c1,c2,c3]*

(v0) *[0,0,0]*

Como vimos, las operaciones básicas sobre los vectores cumplen un conjunto de propiedades:

(%i23) `a+b=b+a;`

(%o23) *[b1+a1,b2+a2,b3+a3]=[b1+a1,b2+a2,b3+a3]*

(%i24) `(a+b)+c=a+(b+c);`

(%o24) *[c1+b1+a1,c2+b2+a2,c3+b3+a3]=[c1+b1+a1,c2+b2+a2,c3+b3+a3]*

(%i25) `a+v0=a;`

(%o25) *[a1,a2,a3]=[a1,a2,a3]*

(%i26) `a-a=v0;`

(%o26) *[0,0,0]=[0,0,0]*

```
(%i27) α·(a+b)=α·a+α·b,factor;
```

$$(\%o27)\quad [(b1+a1)\,\alpha,(b2+a2)\,\alpha,(b3+a3)\,\alpha]=[(b1+a1)\,\alpha,(b2+a2)\,\alpha,(b3+a3)\,\alpha]$$

```
(%i28) (α+β)·a=α·a+β·a,factor;
```

$$(\%o28)\quad [a1\,(\beta+\alpha),a2\,(\beta+\alpha),a3\,(\beta+\alpha)]=[a1\,(\beta+\alpha),a2\,(\beta+\alpha),a3\,(\beta+\alpha)]$$

Con el siguiente comando limpiamos la memoria y borramos todos los cálculos anteriores

```
(%i29) kill(all)$
```

FIN

2 - 1.2.8 Vectores en componentes

Con el programa de manipulación simbólica Maxima haremos algunos cálculos sencillos con vectores. Nuevamente recomendamos ver el apéndice 6.1 como introducción al programa.
Dados los vectores, en coordenadas cartesianas: a = i + 2j + 3k y b = 7i + 8j + 9k

```
(%i1) a:[1,2,3];
```

$$(a)\quad [1,2,3]$$

```
(%i2) b:[7,8,9];
```

$$(b)\quad [7,8,9]$$

La multiplicación por escalares y suma es simple, si queremos calcular α a + β b, escribimos:

```
(%i3) α·a + β·b;
```

$$(\%o3)\quad [7\,\beta+\alpha,8\,\beta+2\,\alpha,9\,\beta+3\,\alpha]$$

Para el producto escalar procedemos utilizando el operador punto, como se muestra a continuación.

```
(%i4) a.b;
(%o4) 50
```

El cálculo de producto vectorial no es tan obvio, debemos cargar previamente la librería vect.

```
(%i5) load(vect)$
```

```
(%i6) express(a~b);
(%o6) [-6,12,-6]
```

La norma de un vector, como ya vimos, es: $\sqrt{a \cdot a}$

```
(%i7) sqrt(a.a);
(%o7) sqrt(14)
```

Si tenemos otro vector, digamos c = −4i + 5j − 6k, el producto triple: $a \cdot b \times c$ se calcula así:

```
(%i8) c:[-4,5,-6];
(c)   [-4,5,-6]
```

```
(%i9) a.express(b~c);
(%o9) 120
```

El ángulo entre los vectores a y b. En Maxima usamos la función acos(x) para el arcocoseno(x). Consultar el manual del programa para ver el resto de las funciones trigonométricas

```
(%i10) acos((a.b)/(sqrt(a.a)·sqrt(b.b)));
```

$$(\%o10)\quad \operatorname{acos}\left(\frac{50}{\sqrt{14}\,\sqrt{194}}\right)$$

Seguramente lo queremos es el valor numérico, esto se hace agregando la función float. Con la siguiente sintaxis logramos el objetivo:

```
(%i11) acos((a.b)/(sqrt(a.a)·sqrt(b.b))),float;
(%o11) 0.2858867976945072
```

Para finalizar, podemos considerar el problema de la independencia lineal de vectores. Tomemos el conjunto de vectores del ejemplo 1.2.7, es decir, los vectores: a = i+2j+3k, b = 4i+5j y c = 3i+2j+k

Disponemos de un comando que permite resolver sistemas de ecuaciones lineales, este comando se llama linsolve. Podemos escribir el sistema de ecuaciones como una lista y luego resolver:

```
(%i12) ecus:[α+4·β+3·γ=0,2·α+5·β+2·γ=0,3·α+γ=0];
(ecus) [3 γ+4 β+α=0,2 γ+5 β+2 α=0,γ+3 α=0]

(%i13) linsolve(ecus,[α,β,γ]);
(%o13) [α=0,β=0,γ=0]

(%i14) kill(all)$
```

FIN

3 - 1.3.4 Aplicaciones del álgebra vectorial

Consideremos el plano determinado por los puntos P = (1, 1, 1), Q = (2, 4, 6) y R = (2, 2, 2). Para construir la ecuación del plano primero ingresamos los vectores posición de cada punto. También incorporaremos las librerías vect y draw que necesitaremos.

```
(%i2) load(vect)$ load(draw)$

(%i3) P:[1,1,1];
(P)   [1,1,1]

(%i4) Q:[2,4,6];
(Q)   [2,4,6]

(%i5) R:[2,2,2];
(R)   [2,2,2]
```

Podemos verificar que los vectores estén en el mismo plano simplemente calculando el triple producto vectorial: P · (Q × R) entre ellos. Sabemos que si es nulo es porque los vectores son coplanares.

```
(%i6)  P.express(Q~R);
(%o6)  0
```

Necesitamos ahora calcular los vectores que van del punto P al punto Q y del punto P al punto R, es decir, los vectores: PQ = Q − P y PR = R − P

```
(%i7)  PQ:Q-P;
(PQ)   [1,3,5]

(%i8)  PR:R-P;
(PR)   [1,1,1]
```

Un vector normal al plano será sencillamente el vector N = PQ × PR:

```
(%i9)  N:express(PQ~PR);
(N)    [-2,4,-2]
```

Podemos escoger cualquiera de los vectores originales, en este caso al vector P, para escribir la ecuación del plano: N · r = N · P.

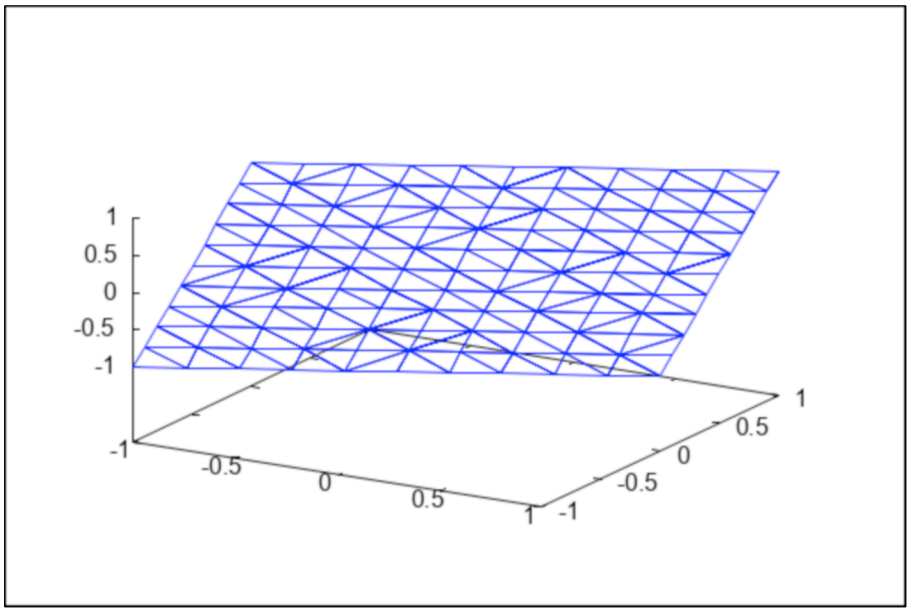
```
(%i10) r:[x,y,z];
(r)    [x,y,z]

(%i11) plano:N.r=N.P;
(plano) -2 z+4 y-2 x=0
```

Probemos a graficar el plano. Para saber mas sobre las diferentes opciones que incorpora Maxima para hacer gráficas, en dos o tres dimensiones, es recomendable consultar el manual de usuario. Utilizaremos el comando wxdraw3d para hacer que la figura aparezca embebida dentro de nuestra hoja de trabajo. Es recomendable consultar también las funciones draw y gr3d.

(%i12)
```
wxdraw3d(implicit(plano,x,-1,1,y,-1,1,z,-1,1));
```

(%t12)



(%o12)

4 - 1.4.6 Álgebra vectorial con índices

A fines prácticos, las transformaciones de coordenadas del tipo rotaciones o reflexiones es de utilidad representarlas por matrices. Por ejemplo, las rotaciones alrededor del eje z se puede representar como:

(%i13)
```
Rz:matrix([cos(θ),sin(θ),0],[-sin(θ),cos(θ),0],[0,0,1]);
```

(Rz)
$$\begin{pmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

De manera que un vector a transformara´ bajo esta rotacio´n en un nuevo vector a′

(%i14)
```
a:[a1,a2,a3];
```

(a)
$$[a1, a2, a3]$$

(%i15)
```
Rz.a;
```

(%o15)
$$\begin{pmatrix} a2\sin(\theta)+a1\cos(\theta) \\ a2\cos(\theta)-a1\sin(\theta) \\ a3 \end{pmatrix}$$

Si la rotación se hace al rededor del eje x o y las matrices correspondientes son las siguientes:

(%i16)
```
Rx:matrix([1,0,0],[0,cos(θ),-sin(θ)],[0,sin(θ),cos(θ)]);
```

(Rx)
$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{pmatrix}$$

(%i17)
```
Ry:matrix([cos(θ),0,sin(θ)],[0,1,0],[-sin(θ),0,cos(θ)]);
```

(Ry)
$$\begin{pmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{pmatrix}$$

Notemos que el determinante de estas matrices es:

(%i18)
```
determinant(Rx),trigsimp;
```

(%o18)
$$\sin(\theta)^2+\cos(\theta)^2$$

(%i19)
```
trigrat(%);
```

(%o19) 1

Si la rotación es en un ángulo de θ = 60° entonces:

(%i20)
```
sublis([θ=%pi/3], Rx);
```

(%o20)
$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{2} & -\frac{\sqrt{3}}{2} \\ 0 & \frac{\sqrt{3}}{2} & \frac{1}{2} \end{pmatrix}$$

Si el eje de rotación se define a través de un vector unitario, digamos, û = ux i + uy j + uz k, la matriz de rotación para un ángulo θ es:

(%i21)
```
t:1-cos(θ)$
```

(%i22)
```
R:matrix([cos(θ)+ux^2·t,ux·uy·t−uz·sin(θ),ux·uz·t+uy·sin(θ)],[uy·ux·t+uz·sin(θ),cos(θ)+uy^2
```

(R)

$$\begin{pmatrix} \cos(\theta)+ux^2\,(1-\cos(\theta)) & ux\,uy\,(1-\cos(\theta))-uz\,\sin(\theta) & uy\,\sin(\theta)+ux\,uz\,(1-\cos(\theta)) \\ uz\,\sin(\theta)+ux\,uy\,(1-\cos(\theta)) & \cos(\theta)+uy^2\,(1-\cos(\theta)) & uy\,uz\,(1-\cos(\theta))-ux\,\sin(\theta) \\ ux\,uz\,(1-\cos(\theta))-uy\,\sin(\theta) & ux\,\sin(\theta)+uy\,uz\,(1-\cos(\theta)) & \cos(\theta)+uz^2\,(1-\cos(\theta)) \end{pmatrix}$$

Por ejemplo, si el eje de rotación coincide con el vector û =1/√3 + j/√3 + k/√3 y la rotación es en un ángulo de θ = 60°, entonces:

(%i24)
```
[ux,uy,uz]:1/sqrt(3)·[1,1,1]; θ:%pi/3$
```

(%o23) $\left[\frac{1}{\sqrt{3}},\frac{1}{\sqrt{3}},\frac{1}{\sqrt{3}}\right]$

Al evaluar con el comando ev podemos ver como queda la matriz R con estos valores definidos

(%i25)
```
Ru:ev(R);
```

(Ru)
$$\begin{pmatrix} \frac{2}{3} & -\frac{1}{3} & \frac{2}{3} \\ \frac{2}{3} & \frac{2}{3} & -\frac{1}{3} \\ -\frac{1}{3} & \frac{2}{3} & \frac{2}{3} \end{pmatrix}$$

De manera que un vector, por ejemplo, a = i + 2j + 3k transformará bajo esta rotación de la manera siguiente:

(%i26)
```
a:[1,2,3];
```

(a) $[1,2,3]$

(%i27)
```
Ru.a;
```

(%o27)
$$\begin{pmatrix} 2 \\ 1 \\ 3 \end{pmatrix}$$

Por otro lado, una reflexión en el plano xy es más sencilla de representar porque es la matriz:

(%i28)
```
Re:matrix([-1,0,0],[0,1,0],[0,0,1]);
```

(Re)
$$\begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

De manera que bajo ésta reflexión el vector transforma como:

(%i29)
```
Re.a;
```

(%o29)
$$\begin{pmatrix} -1 \\ 2 \\ 3 \end{pmatrix}$$

(%i30)
```
kill(all)$
```

FIN

5 - 1.5.8 Un comienzo a la derivación e integración de vectores

Maxima nos permite hacer cálculos con el operador ∇ en coordenadas cartesianas, más adelante veremos que también es posible en otros sistemas de coordenadas. Los operadores que se pueden expresar son: grad(gradiente), div(divergencia), curl(rotacional), laplacian(laplaciano). Debemos hacer uso de la librería vect para poder utilizar los operadores diferenciales. Por otro lado, se debe usar también la función express(expr) para transformar los nombres de los operadores diferenciales en expresiones que contienen derivadas parciales y con la función ev evaluamos la expresión.
Veamos como se hace si queremos calcular el gradiente de la siguiente función:

(%i1)
```
load(vect)$
```

(%i2)
```
f:(x^2+y^2)/(x^2+y^2+z^2)^(1/2);
```

(f)
$$\frac{y^2+x^2}{\sqrt{z^2+y^2+x^2}}$$

(%i3) `grad(f);`

(%o3) $$\text{grad}\left(\frac{y^2+x^2}{\sqrt{z^2+y^2+x^2}}\right)$$

Estaremos haciendo uso del símbolo % que representa la última expresión de salida, de esta manera nos ahorramos estar escribiendo nuevamente el último resultado en la nueva linea de entrada.

(%i4) `express(%);`

(%o4) $$\left[\frac{d}{dx}\frac{y^2+x^2}{\sqrt{z^2+y^2+x^2}},\frac{d}{dy}\frac{y^2+x^2}{\sqrt{z^2+y^2+x^2}},\frac{d}{dz}\frac{y^2+x^2}{\sqrt{z^2+y^2+x^2}}\right]$$

Notemos que el resultado anterior es una lista del tipo [x,y,z], que es la manera como el programa maneja los vectores. En este caso, el primer elemento de la lista es la componente x del vector gradiente. De la misma manera para las componentes y y z. Evaluamos las derivadas de la última salida:

(%i5) `ev(%, diff);`

(%o5) $$\left[\frac{2x}{\sqrt{z^2+y^2+x^2}}-\frac{x(y^2+x^2)}{(z^2+y^2+x^2)^{3/2}},\frac{2y}{\sqrt{z^2+y^2+x^2}}-\frac{y(y^2+x^2)}{(z^2+y^2+x^2)^{3/2}},-\frac{(y^2+x^2)z}{(z^2+y^2+x^2)^{3/2}}\right]$$

En este punto podemos intentar simplificar las expresiones anteriores, esto lo hacemos con el comando ratsimp y así obtener finalmente el vector gradiente:

(%i6)
```
ratsimp(%);
```

(%o6)
$$\left[\frac{\sqrt{z^2+y^2+x^2}\,(2xz^2+xy^2+x^3)}{z^4+(2y^2+2x^2)z^2+y^4+2x^2y^2+x^4},\frac{\sqrt{z^2+y^2+x^2}\,(2yz^2+y^3+2x^2y)}{z^4+(2y^2+2x^2)z^2+y^4+2x^2y^2+x^4},-\frac{(y^2+x^2)z}{(z^2+y^2+x^2)^{3/2}}\right]$$

Hagamos uso de los otros operadores. Por ejemplo, dado un vector calculemos la divergencia, $\nabla\cdot a$. Esta vez combinaremos algunos comandos en una misma linea.

(%i7)
```
a:[x^2/(x^2+y^2),y^2/(x^2+y^2),z^2/(x^2+y^2)];
```

(a)
$$\left[\frac{x^2}{y^2+x^2},\frac{y^2}{y^2+x^2},\frac{z^2}{y^2+x^2}\right]$$

(%i8)
```
express(div (a));
```

(%o8)
$$\frac{d}{dz}\frac{z^2}{y^2+x^2}+\frac{d}{dy}\frac{y^2}{y^2+x^2}+\frac{d}{dx}\frac{x^2}{y^2+x^2}$$

(%i9)
```
ev(%, diff),ratsimp;
```

(%o9)
$$\frac{(2y^2+2x^2)z+2xy^2+2x^2y}{y^4+2x^2y^2+x^4}$$

Calculemos ahora el rotor de a, $\nabla\times a$, pero combinado más comandos en una sola instrucción:

(%i10)
```
ev(express(curl(a)),diff);
```

(%o10)
$$\left[-\frac{2yz^2}{(y^2+x^2)^2},\frac{2xz^2}{(y^2+x^2)^2},\frac{2x^2y}{(y^2+x^2)^2},-\frac{2xy^2}{(y^2+x^2)^2}\right]$$

Intentemos simplificar nuevamente para ver lo que ocurre:

(%i11) `ratsimp(%);`

(%o11) $$\left[-\frac{2yz^2}{y^4+2x^2y^2+x^4},\frac{2xz^2}{y^4+2x^2y^2+x^4},-\frac{2xy^2-2x^2y}{y^4+2x^2y^2+x^4}\right]$$

Finalmente, calculemos el laplaciano de f, todo en una sola instrucción

(%i12) `ratsimp(ev(express(laplacian(f)),diff));`

(%o12) $$\frac{4z^2}{(z^2+y^2+x^2)^{3/2}}$$

Podemos verificar rapidamente que el laplaciano f = ∇ · ∇f

(%i13) `div(grad(f));`

(%o13) $$\mathrm{div}\left(\mathrm{grad}\left(\frac{y^2+x^2}{\sqrt{z^2+y^2+x^2}}\right)\right)$$

(%i14) `ratsimp(ev(express((%)),diff));`

(%o14) $$\frac{4z^2}{(z^2+y^2+x^2)^{3/2}}$$

(%i15) `kill(all)$`

Maxima integra simbólicamente funciones por medio del comando integrate(expr,x,a,b). Esto es, calcula la integral definida de la expr respecto de x con los límites de integración a y b. Si se omiten los límites de integraci´on se calcula la integral indefinida. Por ejemplo

(%i1) `integrate(a·sin(x+b)^3,x);`

(%o1) $$a\left(\frac{\cos^3(x+b)}{3}-\cos(x+b)\right)$$

Calculemos ahora el área de la región comprendida entre las funciones: f(x)=x^3−3x^2+1 y g(x)=−x+1. Aprovecharemos este ejercicio para aprender, entre otras cosas, a graficar y definir funciones. Primero que todo, debemos introducir al programa las dos funciones y a diferencia de los cálculos anteriores, usaremos := para definirlas.

(%i3)
```
f(x):=x^3-3·x^2+1; g(x):=-x+1;
```

(%o2) $f(x):=x^3-3x^2+1$

(%o3) $g(x):=-x+1$

La diferencia de estas dos funciones la podemos llamar la función d(x).

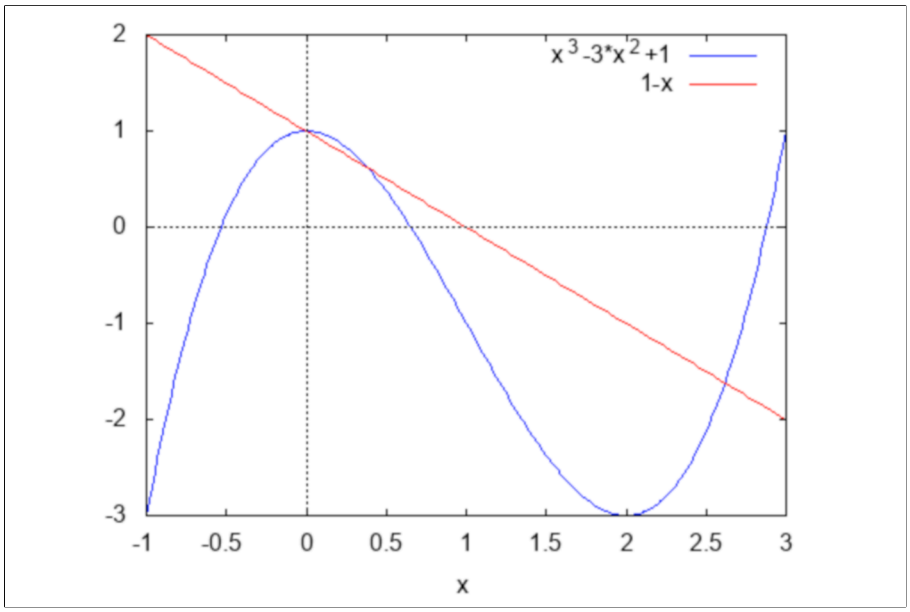
(%i4)
```
d(x):=f(x)-g(x);
```

(%o4) $d(x):=f(x)-g(x)$

Haremos la gráfica de las dos funciones. Maxima permite hacer gráficos en 2D con el comando plot2d. Pero aquí utilizaremos una variante y haremos el gráfico con el comando wxplot2d para que la figura aparezca embebida dentro de nuestra hoja de cálculo. Es necesario introducir las funciones como una lista, es decir, dentro de corchetes. Por otra parte, le pediremos al programa que grafique para los valores comprendidos de: $-1 \leq x \leq 3$. Posteriormente, necesitaremos encontrar los puntos donde las funciones se interceptan. Para este fin, utilizaremos el comando solve, que resuelve la ecuación algebraica y devuelve una lista de igualdades con las variables despejadas. Si la expresión a resolver no es una igualdad, se supone que se quiere resolver la ecuación ya igualada a cero. Entonces, para graficar las funciones f(x) y g(x) entre [−1,3] escribimos el siguiente comando:

(%i5)
```
wxplot2d([f(x),g(x)],[x,-1,3]);
```

(%t5)



(%o5)

Calculemos los puntos donde se interceptan las curvas:

(%i6)
```
solve(f(x)-g(x),x);
```

(%o6) $$[x=-\frac{\sqrt{5}-3}{2},x=\frac{\sqrt{5}+3}{2},x=0]$$

Los valores numéricos son:

(%i7)
```
float(%);
```

(%o7) $[x=0.3819660112501051,x=2.618033988749895,x=0.0]$

Procedemos ahora sí a integrar la función diferencia para encontrar el área contenida dentro de las dos funciones.

(%i8)
```
integrate(d(x),x,0,-(sqrt(5)-3)/2)+integrate(-d(x),x,-(sqrt(5)-3)/2,(sqrt(5)+3)/2);
```

(%o8) $$\frac{5^{3/2}+11}{8}+\frac{5^{3/2}-11}{4}$$

(%i9)
```
float(%);
```

(%o9) 2.817627457812107

Calculemos ahora las dos integrales que aparecen en el ejemplo 8 de 1.5.7. Escribiremos primero los dos integrandos:

(%i11) `int1:(3·x^2+2·x·y^3)·dx; int2:6·x·y·dy;`

(int1) $dx\,(2\,x\,y^3+3\,x^2)$

(int2) $6\,dy\,x\,y$

Realizaremos el cambio de variable a través de comando subst que nos permite sustituir expresiones dentro de otras:

(%i12) `e1:subst([x=2·t^2,dx=4·t,y=t^3+t,dy=3·t^2+1],int1);`

(e1) $4\,t\left(4\,t^2\,(t^3+t)^3+12\,t^4\right)$

(%i13) `e2:subst([x=2·t^2,dx=4·t,y=t^3+t,dy=3·t^2+1],int2);`

(e2) $12\,t^2\,(3\,t^2+1)\,(t^3+t)^3$

Podemos introducir las integrales sin evaluarlas en el momento, esto se consigue mediante el operador de comilla simple. En este caso la función no es evaluada y devuelve una expresi´on simbólica o imagen pictórica.

(%i14) `int1:'integrate(e1,t,0,sqrt(2)/2);`

(int1) $\displaystyle\int_0^{\frac{1}{\sqrt{2}}} 4\,t\left(4\,t^2\,(t^3+t)^3+12\,t^4\right)dt$

Ahora si evaluamos la integral:

(%i15) `ev(int1,integrate),expand;`

(%o15) $\dfrac{9305}{3003\,2^{5/2}}+1$

Y para la segunda integral:

(%i16) `int2:'integrate(e2,t,0,sqrt(2)/2);`

(int2) $\displaystyle\int_0^{\frac{1}{\sqrt{2}}} 12\,t^2\,(3\,t^2+1)\,(t^3+t)^3\,dt$

(%i17)
```
ev(int2,integrate),expand;
```
(%o17) $\frac{65}{32}$

(%i18)
```
kill(all)$
```

Para finalizar, aclaremos la diferencia entre asignar una expresión a una variable y la definición de funciones. Consideremos el movimiento de una partícula que sigue la trayectoria:

(%i1)
```
x:x0+v0·t+a·t^2/2;
```
(x) $x0+t\,v0+\frac{a\,t^2}{2}$

Si queremos evaluarla para algún valor de la variable t, digamos t = 1, entonces podemos escribir:

(%i2)
```
subst(t=1,x);
```
(%o2) $x0+v0+\frac{a}{2}$

Si queremos evaluar la expresión con el resto de parámetros, entonces:

(%i3)
```
subst([t=1,a=-9.8,x0=0,v0=10],x);
```
(%o3) 5.1

Las sustituciones se realizan pero no se asignan a las variables. Podemos ver que x sigue siendo lo que le asignamos originalmente.

(%i4)
```
x;
```
(%o4) $x0+t\,v0+\frac{a\,t^2}{2}$

Ahora definamos la expresión anterior pero como la función x(t).

(%i5)
```
x(t):= x0+v0·t+a·t^2/2;
```
(%o5) $x(t):=x0+v0\,t+\frac{a\,t^2}{2}$

Evaluarla es ahora más sencillo:

(%i6) `x(1);`

(%o6) $x0+v0+\frac{a}{2}$

También podemos escribir:

(%i7) `subst([a=-9.8,x0=0,v0=10],x(1));`

(%o7) 5.1

Si queremos asignarle los valores a los parámetros de manera global, entonces escribimos:

(%i10) `a:-9.8$ x0:0$ v0:10$`

De manera que:

(%i11) `x(1);`

(%o11) 5.1

Para limpiar los parámetros que acabamos de asignar hacemos lo siguiente:

(%i12) `kill(a,v0,x0)$`

Por lo tanto:

(%i13) `x(1);`

(%o13) $x0+v0+\frac{a}{2}$

La velocidad y la aceleración son funciones fáciles de calcular:

(%i14) `'diff(x(t),t)=diff(x(t),t);`

(%o14) $\frac{d}{d\,t}\left(x0+t\,v0+\frac{a\,t^2}{2}\right)=v0+a\,t$

(%i15) `'diff(x(t),t,2)=diff(x(t),t,2);`

(%o15) $\frac{d^2}{d\,t^2}\left(x0+t\,v0+\frac{a\,t^2}{2}\right)=a$

En el caso de que estemos interesados en definir una nueva función a partir de algún cálculo anterior podemos utilizar la función "define". Entonces, lo que vamos a hacer en las siguientes lineas de comandos, es definir la función velocidad a partir de la derivada de la posición.

```
(%i16) diff(x(t),t);
(%o16) v0+a t

(%i17) define(v(t),%);
(%o17) v(t):=v0+a t

(%i18) v(1);
(%o18) v0+a

(%i19) subst([a=-9.8,x0=0,v0=10],v(1));
(%o19) 0.1999999999999993

(%i20) kill(all)$
```

Cuando necesitemos manejar campos vectoriales podemos utilizar las listas. Veamos el primer ejemplo de 1.5.7 donde: r=3t^2i+(4t^3 -t)j+tk. Escribamos el vector posición:

```
(%i1) [x,y,z]: [3·t^2,4·t^3-t,t];
                    2    3
(%o1) [3 t ,4 t -t,t]
```

La velocidad:

```
(%i2) v: diff([x,y,z],t);
                      2
(v)   [6 t,12 t -1,1]
```

La aceleración:

```
(%i3) a: diff([x,y,z],t,2);
(a)   [6,24 t,0]
```

Podemos facilmente calcular:

```
(%i4)  v.[bx,by,bz];
                          2
(%o4)  by (12 t  - 1) + 6 bx t + bz

(%i5)  solve(%,bz);
                                2
(%o5)  [bz = - 12 by t  - 6 bx t + by]

(%i6)  kill(all)$

       FIN
```