Funciones elementales y gráficas

1 Comandos básicos: definición de funciones.

En esta asignatura trabajaremos principalmente con funciones. Para definir un objeto en maxima debemos usar : . Para definir una función existen dos comandos. Puede utilizarse := o el comando define. Todas las definiciones deben terminar con ; y para que el programa las ejecute es necesario presionar simultáneamente shift+enter. Veremos en los siguientes ejemplos las diferencias entre los comandos.

```
(%i1) f:x;
(f)   x
```

En este ejemplo hemos asignado a la letra f el valor x. No es una función, cada vez que usemos f la reemplazará por x, pero no podemos evaluar en valores particulares:

```
(%i2) f+f;
(%o2) 2 x

(%i3) f(1);
(%o3) x(1)
```

Este comando es útil para dar un nombre a alguna expresión más compleja, pero no sirve para definir funciones. Para ello, usamos la notación usual f(x):=expresion, o bien define(f(x),expresion).

```
(%i4) f(x):=x+1;
(%o4) f(x):=x+1

(%i7) f(1); f(2); f(3);
(%o5) 2
(%o6) 3
(%o7) 4

(%i8) define(g(t),t+5);
(%o8) g(t):=t+5
```

```
(%i11) g(1); g(2); g(3);
(%o9) 6
(%o10) 7
(%o11) 8
```

De esta manera, maxima reconoce ya desde la definición que lo que aparece entre paréntesis es una variable, y no importa su nombre. Podemos por ejemplo componer funciones de la manera usual, sumarlas, etc.

```
(%i12) f(g(x));
(%o12) x+6
```

```
(%i13) f(x)+g(x);
(%o13) 2 x+6
```

Una de las características más potentes de máxima es que permite trabajar con lenguaje simbólico (es decir, introducir parámetros sin especificar sus valores numéricos). Consideremos por ejemplo la siguiente función:

```
(%i14) h(x):=f(x)+c;
(%o14) h(x):=f(x)+c
```

```
(%i17) h(1); h(2); h(3);
(%o15) c+2
(%o16) c+3
(%o17) c+4
```

2 Funciones ya programadas en máxima

Máxima tiene ya incluidas la suma, la resta, el producto y el cociente usuales. Los comandos son los usuales, teniendo que tener cuidado con el producto que se realiza con * (el asterisco, que automáticamente máxima convierte en un punto) :
+ : suma
- : resta
• : producto
/ : cociente

(%i21) 3+5; 5−3; 3/2; 3·2;

(%o18) 8

(%o19) 2

(%o20) $\frac{3}{2}$

(%o21) 6

La potencia se define con ^ y la raiz cuadrada con sqrt. Para calcular raices de otros indices, recordamos que la raiz n-esima de un número no es más que elevar ese número a la potencia 1/n. Máxima reconoce exponentes fraccionarios y deja expresadas como raices aquellas que dan números irracionales:

(%i25) 2^3; sqrt(4); sqrt(2); 8^(1/3);

(%o22) 8

(%o23) 2

(%o24) $\sqrt{2}$

(%o25) 2

Para obtener una aproximación numérica de un número como la raiz de 2, usamos el comando float. Podemos usar % para indicar a máxima que evalue la nueva función en una entrada inmediatamente anterior;

(%i26) sqrt(3);

(%o26) $\sqrt{3}$

(%i27) float(%);

(%o27) 1.732050807568877

Estas operaciones pueden aplicarse también a funciones:

(%i28) f(x)^2+3·g(x)−h(x);

(%o28) $(x+1)^2+3\,(x+5)-x-c-1$

El valor absoluto está dado por abs:

(%i29) a(x):=abs(x);

(%o29) $a(x):=|x|$

```
(%i33) a(−3); a(−5); a(x)^2; a(x)^3;
```
(%o30) 3
(%o31) 5
(%o32) x^2
(%o33) $x^2\,|x|$

Funciones que pueden resultar de utilidad son la función parte entera y la función signo. Los comandos son:
entier(x) : parte entera de x
signum(x) : devuelve 1 si x es positivo, -1 si x es negativo, 0 si x es cero.

```
(%i37) e(x):=entier(x); e(1.5); e(5); e(sqrt(2));
```
(%o34) $e(x) := \text{entier}(x)$
(%o35) 1
(%o36) 5
(%o37) 1

```
(%i41) s(x):=signum(x); s(5); s(−5); s(0);
```
(%o38) $s(x) := \text{signum}(x)$
(%o39) 1
(%o40) −1
(%o41) 0

También podemos usar las funciones trigonometricas:
cos : coseno
sin : seno
tan : tangente
csc: cosecante
sec : secante
Anteponiendo la letra "a" a las funciones anteriores obtenemos sus inversas (acos, asin, atan, acsc, asec).
El número "pi" se introduce con %pi. Es importante recordar que las funciones trigonométricas están dadas para el sistema radian.

```
(%i47) cos(0); sin(%pi/2); cos(%pi/6); sin(%pi/6); tan(%pi/3); tan(%pi/4);
```

(%o42) 1

(%o43) 1

(%o44) $\frac{\sqrt{3}}{2}$

(%o45) $\frac{1}{2}$

(%o46) $\sqrt{3}$

(%o47) 1

```
(%i50) acos(1); asin(sqrt(3)/2); atan(1);
```

(%o48) 0

(%o49) $\frac{\pi}{3}$

(%o50) $\frac{\pi}{4}$

3 Funciones dadas por partes.

Maxima permite definir "funciones a trozos" utilizando el comando lógico if - then - else. Lo explicaremos mediante un ejemplo:

```
(%i51) k(x):= if x>0 then cos(x) else sin(x);
```

(%o51) $k(x) := \text{if } x > 0 \text{ then } \cos(x) \text{ else } \sin(x)$

```
(%i54) k(0); k(%pi); k(-%pi);
```

(%o52) 0

(%o53) −1

(%o54) 0

Es decir, la función k vale coseno de x si x es positivo y seno de x si x es negativo o cero.

Será importante contar con los siguientes comandos:
> mayor
>= mayor o igual
< menor
<= menor o igual

Veamos otro ejemplo:

```
(%i55) m(x):= if x>=3 then x^2 else −2·x+1;
```

(%o55) $m(x):=\text{if } x\geq 3 \text{ then } x^2 \text{ else } (-2)\,x+1$

```
(%i59) m(0); m(3); m(4); m(2);
```

(%o56) 1
(%o57) 9
(%o58) 16
(%o59) −3

Podemos usar esta estructura para definir funciones en más de dos partes, prestando mucha atención a la sintaxis.

Consideremos la función n(x) dada por:
1 si x es menor o igual a cero
2 si cero es menor que x menor o igual a 2
3 si x es mayor que 2.

```
(%i60) n(x):=if x<= 0 then 1 else if x<=2 then 2 else 3;
```

(%o60) $n(x):=\text{if } x\leq 0 \text{ then } 1 \text{ else if } x\leq 2 \text{ then } 2 \text{ else } 3$

```
(%i67) n(−1); n(0); n(1); n(1.5); n(2);n(3); n(1000);
```

(%o61) 1
(%o62) 1
(%o63) 2
(%o64) 2
(%o65) 2
(%o66) 3
(%o67) 3

4 Simplificación de expresiones

Muchas veces máxima no reconoce identidades ocultas y arroja resultados complejos. Es importante en estos casos trabajar con los comandos que permiten simplificar expresiones. Veamos un ejemplo:

```
(%i68) (c+1)^2−c^2−2·c−1;
```

(%o68) $(c+1)^2-c^2-2\,c-1$

Como podemos observar, maxima "no se da cuenta" que esta expresion debe ser 0.El comando más usual para simplificar expresiones algebraicas es ratsimp

```
(%i69) ratsimp(%);
(%o69) 0
```

Otros comandos útiles son:
ratsimp: simplifica expresiones algebraicas
radcan : simplifica radicales
factor : factoriza una expresión
trigsimp : simplifica expresiones trigonométricas

Otros comandos pueden encontrarse en la pestaña "simplificar"

```
(%i70) x^2+2·x+1;
(%o70) x^2+2 x+1
```

```
(%i71) factor(%);
(%o71) (x+1)^2
```

```
(%i72) cos(x)^2+sin(x)^2;
(%o72) sin(x)^2+cos(x)^2
```

```
(%i73) trigsimp(%);
(%o73) 1
```

5 Gráficas de funciones

Existen dos comandos distintos para graficar funciones. El primero, y más clásico de maxima, es el comando plot2d. Un comando reciente que ofrece más opciones, pero es un poco más complicado a la hora de escribir las instrucciones, es el comando draw2d.

5.1 El comando plot2d

Este comando permite introducir varias particularidades de a la grafica que pueden consultarse en la ayuda. Veremos las características principales.
En su forma base, el comando plot2d tiene los siguientes argumentos:

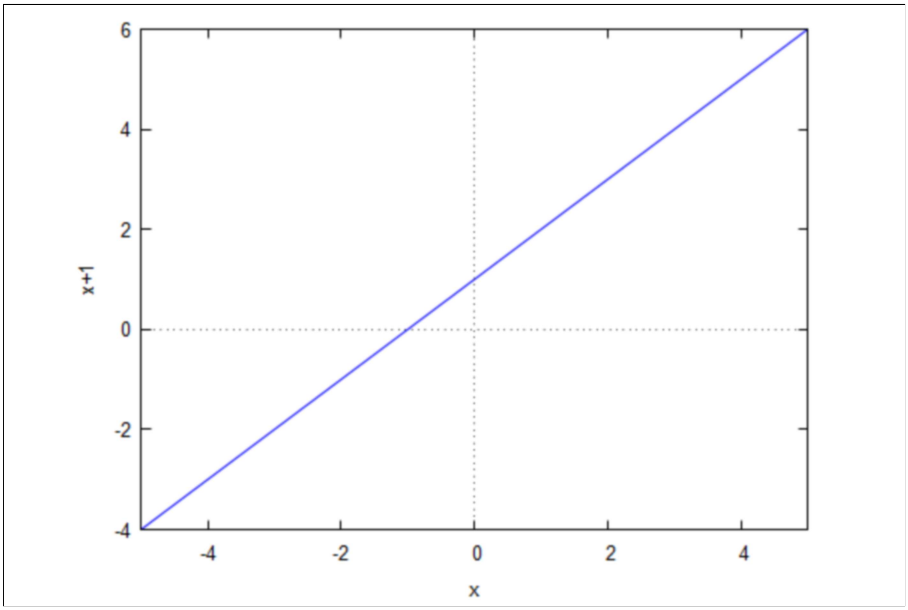
plot2d(funcion, [variable, limite inferior del intervalo, limite superior del intervalo]);

La grafica aparecerá en una ventana emergente, que nos permitirá manipularla.

La opción wxplot2d inserta la gráfica en la hoja de trabajo pero no nos permite modificarla.

(%i74)
```
plot2d(f(x),[x,-5,5]);
```
(%o74) *[C:/Users/Francisco/AppData/Local/Temp/maxout13572.gnuplot]*

(%i75)
```
wxplot2d(f(x),[x,-5,5]);
```
(%t75)



(%o75)

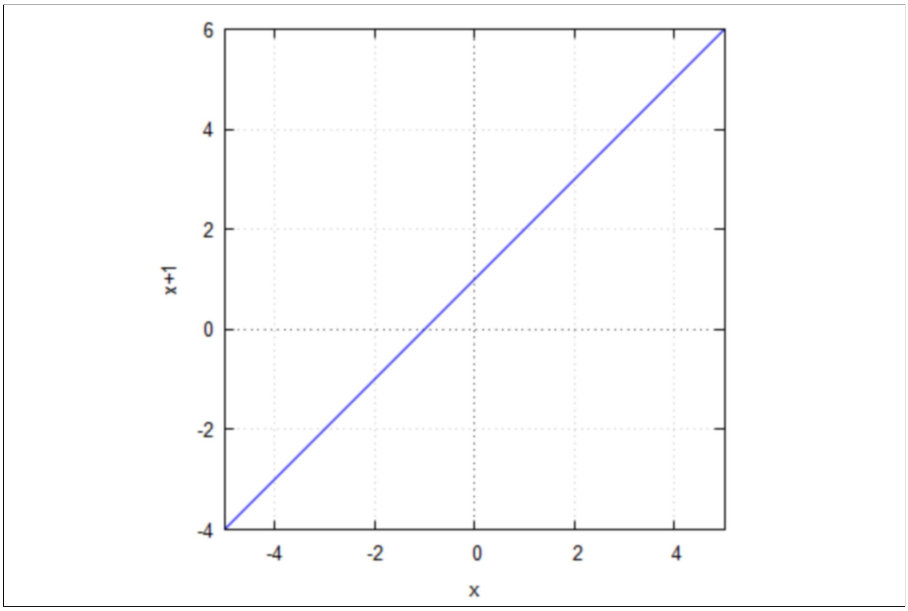
Una primera observación que podemos hacer, es que maxima escala comunmente los ejes x e y de manera diferente. Para evitar esto podemos incluir el comando

same_xy

dentro de las instrucciones. Otra opcion que puede incluirse es grid2d, que incluye el mallado dentro de la grafica:

(%i76) wxplot2d(f(x),[x,−5,5],same_xy,grid2d);

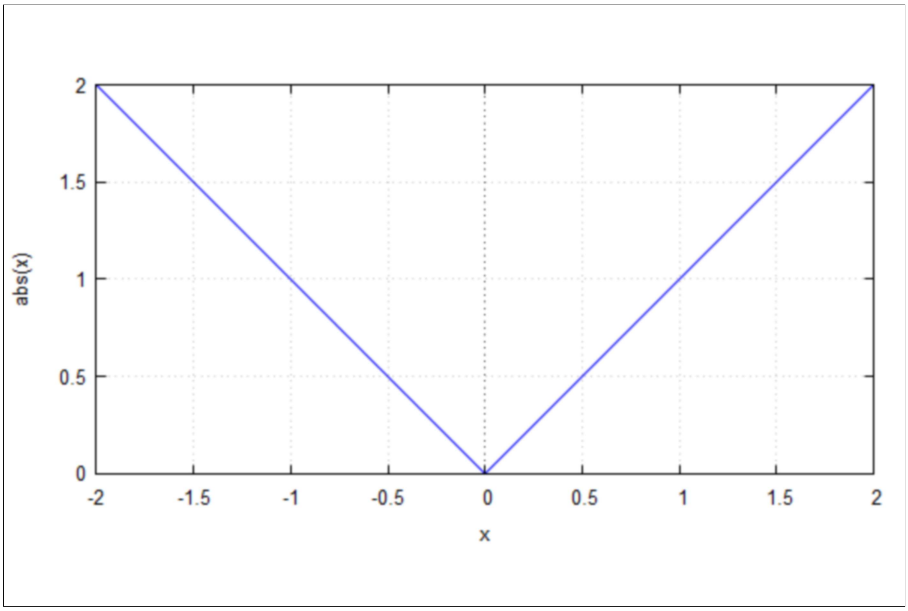
(%t76)



(%o76)

(%i77) wxplot2d(a(x),[x,−2,2],same_xy,grid2d);

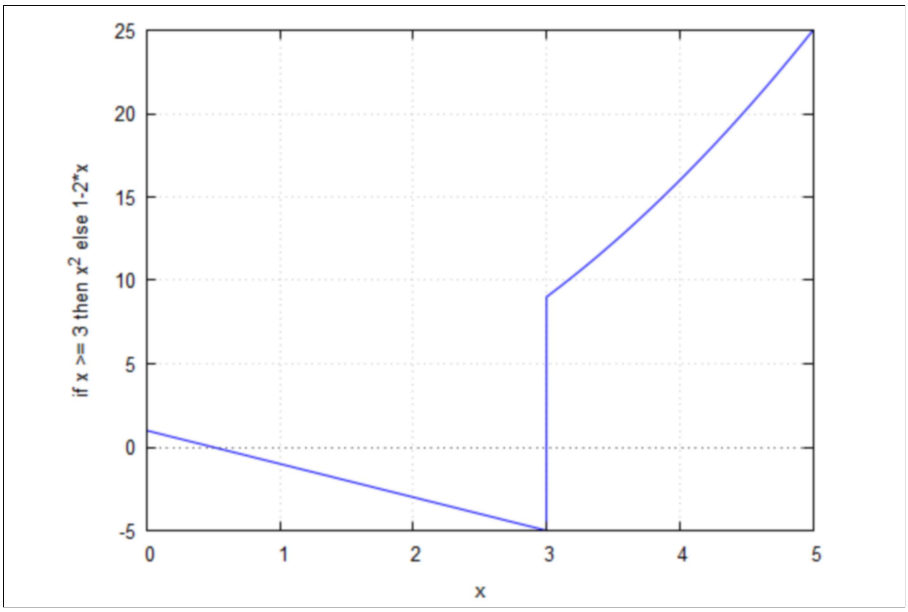
(%t77)



(%o77)

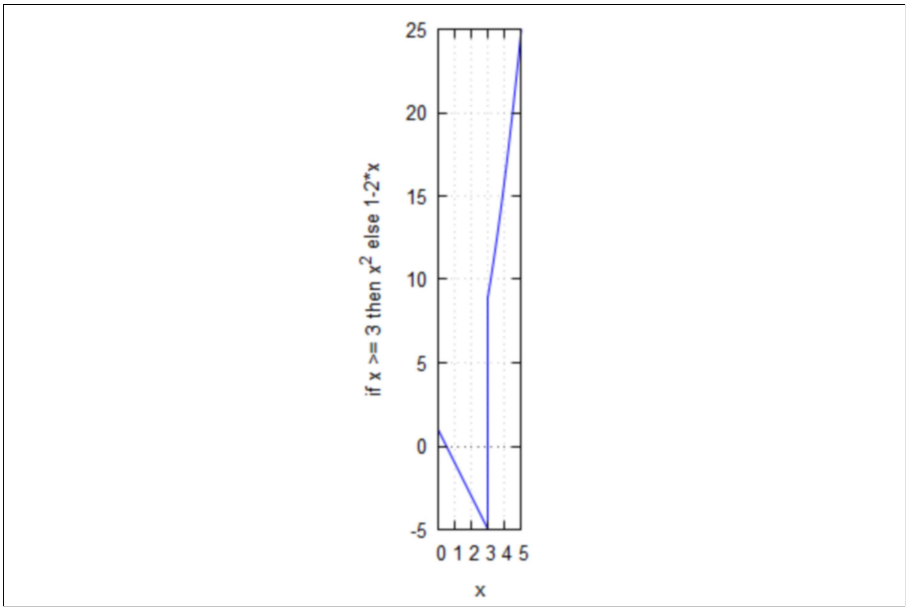
(%i79)
```
wxplot2d(m(x),[x,0,5],grid2d);
wxplot2d(m(x),[x,0,5],same_xy,grid2d);
```

(%t78)



(%o78)

(%t79)

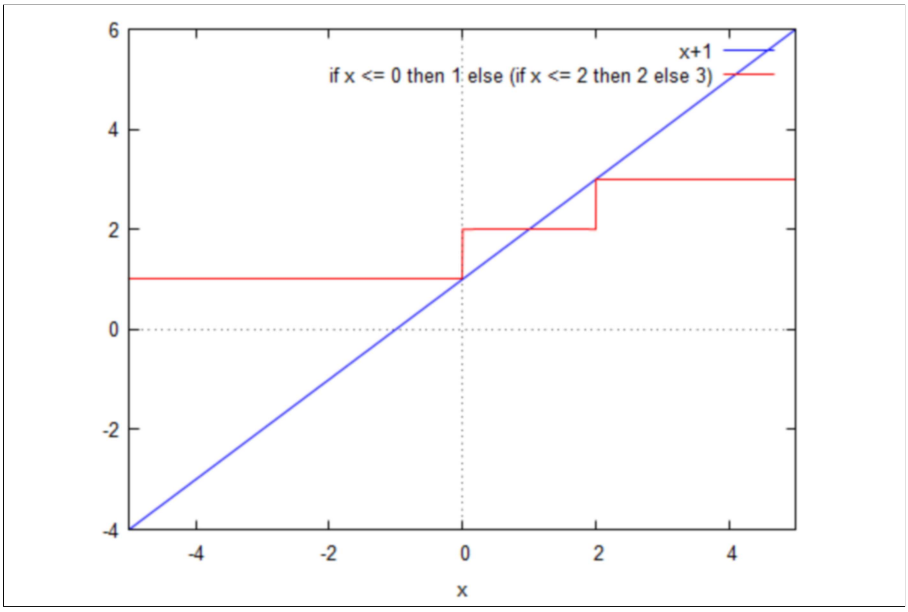


(%o79)

Podemos graficar ademas varias funciones a la vez con la estructura plot2d([funcion1,funcion2, etc.],[variable, limite inferior del intervalo, limite superior del intervalo])

(%i80) wxplot2d([f(x),n(x)],[x, −5, 5]);

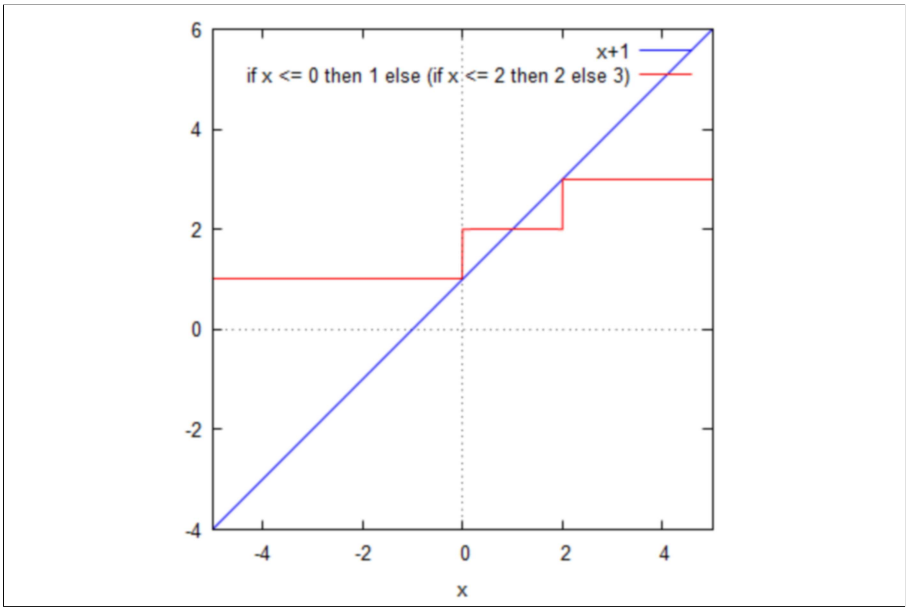
(%t80)



(%o80)

(%i81) wxplot2d([f(x),n(x)],[x, −5, 5],same_xy);

(%t81)



(%o81)

Como podemos observar, existe un problema al intentar graficar funciones definidas a trozos, pues maxima completa con una linea vertical en los puntos de salto. Deberemos tener cuidado al interpretar las graficas en estos casos.

5.2 El comando draw2d

Primero que nada para poder usar el comando draw debemos cargar un paquete:

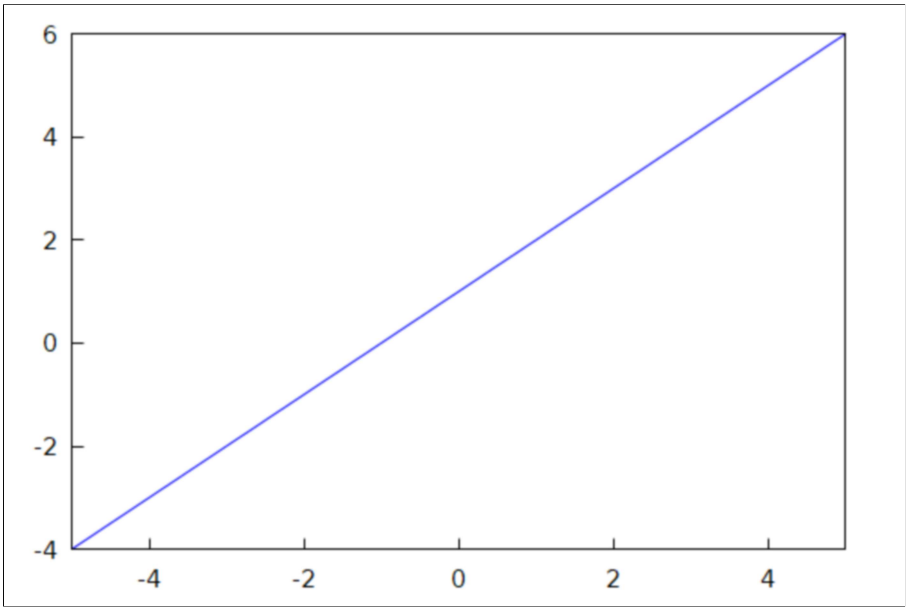
(%i82)
```
load(draw);
```
(%o82) *C:\maxima-5.42.2\share\maxima\5.42.2\share\draw\draw.lisp*

Draw presenta algunas complicaciones, como por ejemplo que tenemos que indicar en cada caso de qué tipo de gráfica se trata. Para el caso de graficas de funciones, la instrucción base es:

draw2d(explicit(funcion, variable, minimo intervalo, maximo intervalo));

Igual que con plot2d, podemos anteponer wx para que el grafico se incluya en la hoja de trabajo:

(%i96)
```
wxdraw2d(explicit(f(x),x,-5,5));
```
(%t96)



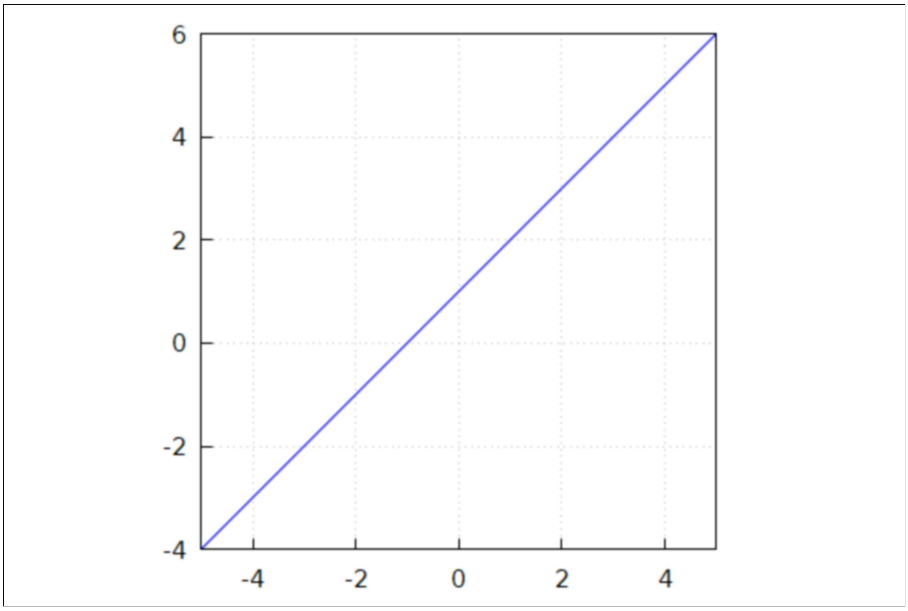
(%o96)

Para agregar las grillas en la gráfica y escalar los ejes, usamos los comandos:

grid=true
proportional_axes=xy

(%i100)
```
wxdraw2d(proportional_axes=xy,
    explicit(f(x),x,−5,5),
grid=true);
```

(%t100)



(%o100)

Pareciera que draw2d presenta una interfaz más compleja para obtener los mismos resultados con plot2d, y es cierto. Para graficar funciones con plot2d es suficiente. Veremos más adelante que draw presenta algunas caracterísiticas interesantes para realizar otro tipo de gráficas.

6 *Otrás gráficas*

6.1 Ecuaciones paramétricas e implícita en 2d.

Maxima permite además graficar graficas de curvas o superficies dadas en sus ecuaciones paramétricas o en forma explicita. Abordaremos estos temas más adelante, incluimos aquí algunos ejemplos.

Una curva dada en forma paramétrica es aquella en que las coordenadas (x,y) o (x,y,z) de los puntos de la curva están dados en función de un parámetro. El caso más simple es el de una recta, dada por x=x_0+t*v_1, y=y_0+t*v_2. E el caso general, tendremos x=f(t), y=g(t) (y eventualmente z=z(t) para curvas en el espacio). En el caso de la recta, tendremos f(t)=x_0+t*v_1, g(t)=y_0+tv_2.

Los comandos en este caso son:

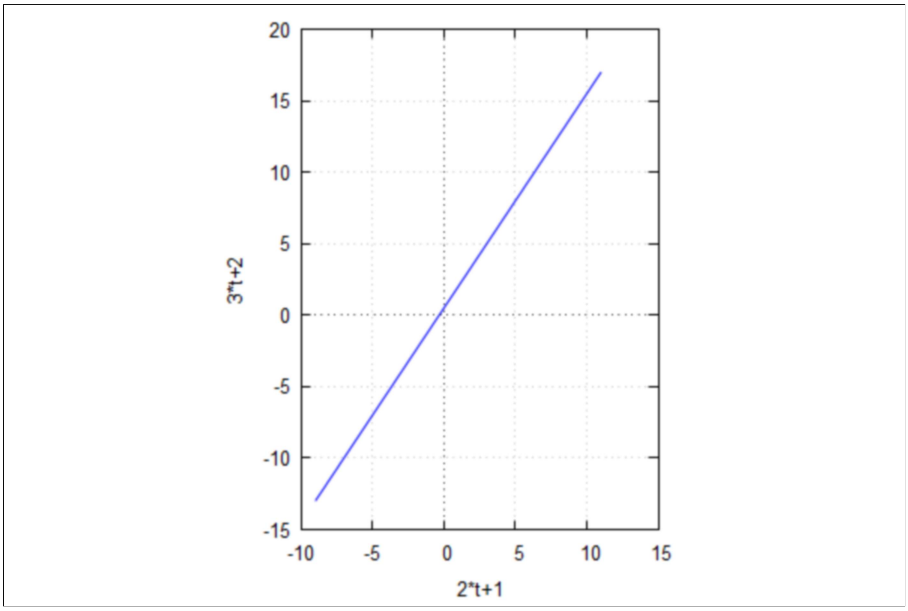
plot2d([parametric, f(t),g(t),[t,minimo del int, maximo del int]]).
draw2d(parametric(f(t),g(t),t, minimo int, maximo int));

De la misma manera que antes, podemos reemplazarlo por wxplot2d o wxdraw2d para que incluya el grafico en la hoja de trabajo. Podemos agregar también los comandos respectivos para que los ejes estén igualmente escalados o para que aparezca la grilla.

Por ejemplo, para graficar la recta x(t)=1+2*t, y(t)=2+3*t para t en [-5,5] hacemos:

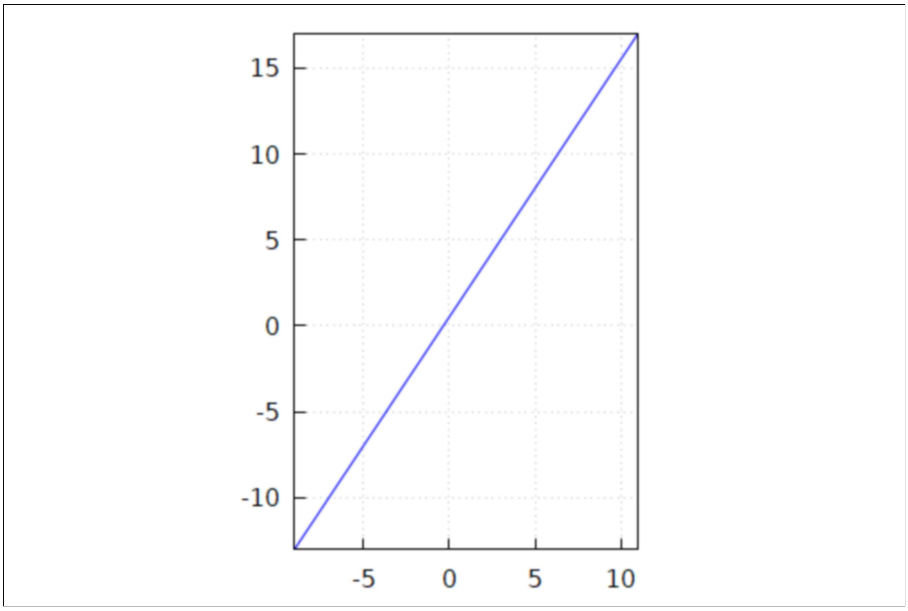
(%i110)
```
wxplot2d([parametric,1+2·t,2+3·t,[t,−5,5]],
    same_xy,
    grid2d);
wxdraw2d(parametric(1+2·t,2+3·t, t, −5,5),
proportional_axes=xy,
    grid=true);
```

(%t109)



(%o109)

(%t110)

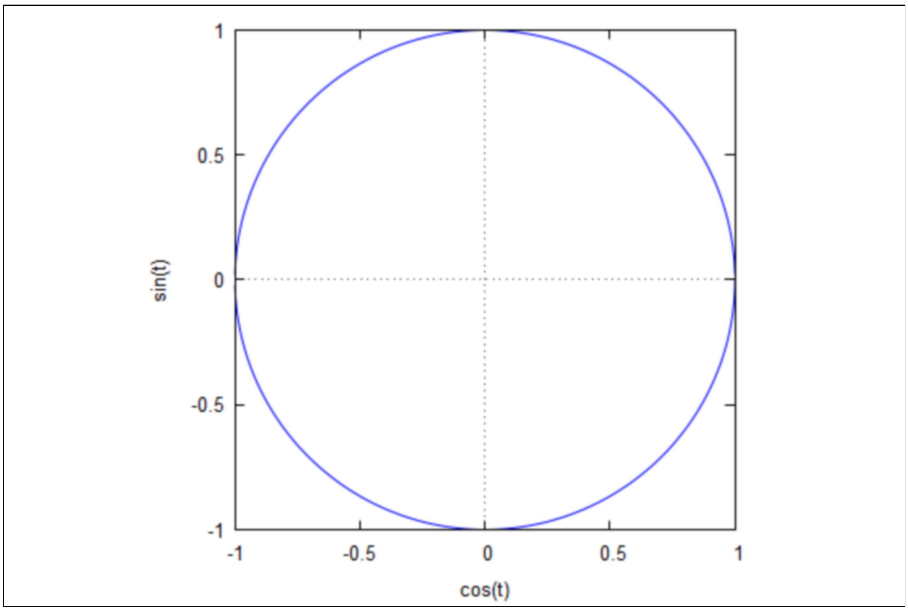


(%o110)

Grafiquemos ahora la curva dada por x(t)=cos(t), y(t)=sen(t), t en [0,2pi]

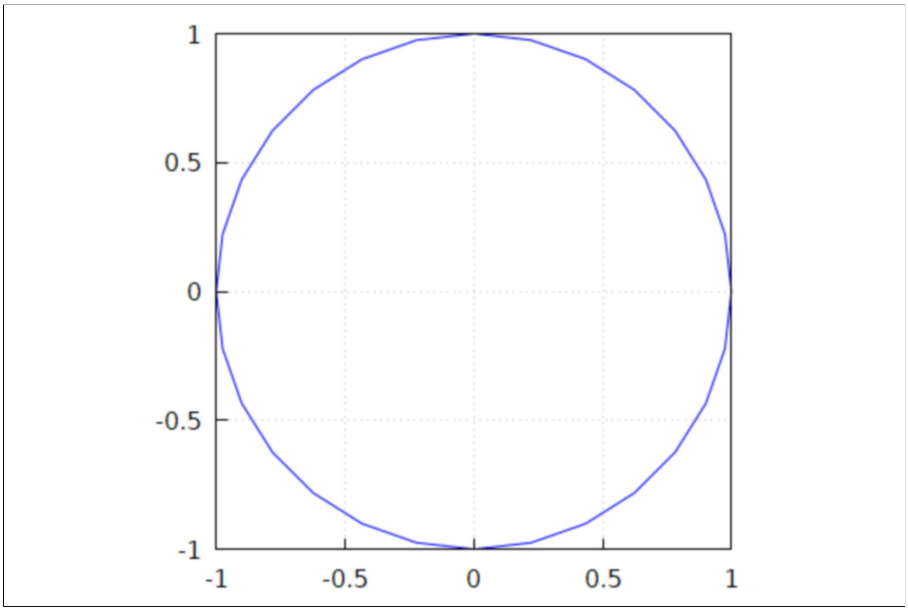
(%i113)
```
wxplot2d([parametric, cos(t),sin(t),[t,0,2·%pi]],same_xy);
wxdraw2d(parametric(cos(t),sin(t),t,0,2·%pi),
proportional_axes=xy,
grid=true);
```

(%t112)



(%o112)

(%t113)



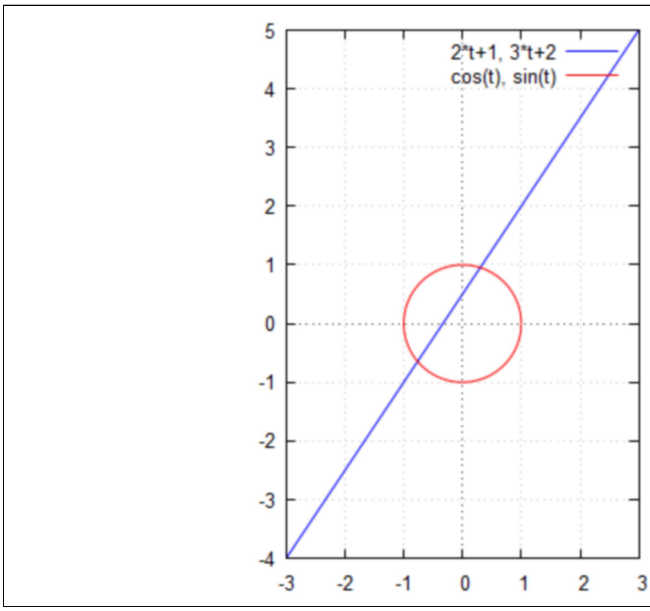
(%o113)

Podemos finalmente graficar dos curvas dadas en forma paramétrica, colocando una después de otra entre corchetes o entre paréntesis, según el comando:

plot2d([[parametrica1],[parametrica2]]);
draw2d(parametric(curva1),parametric(curva2));

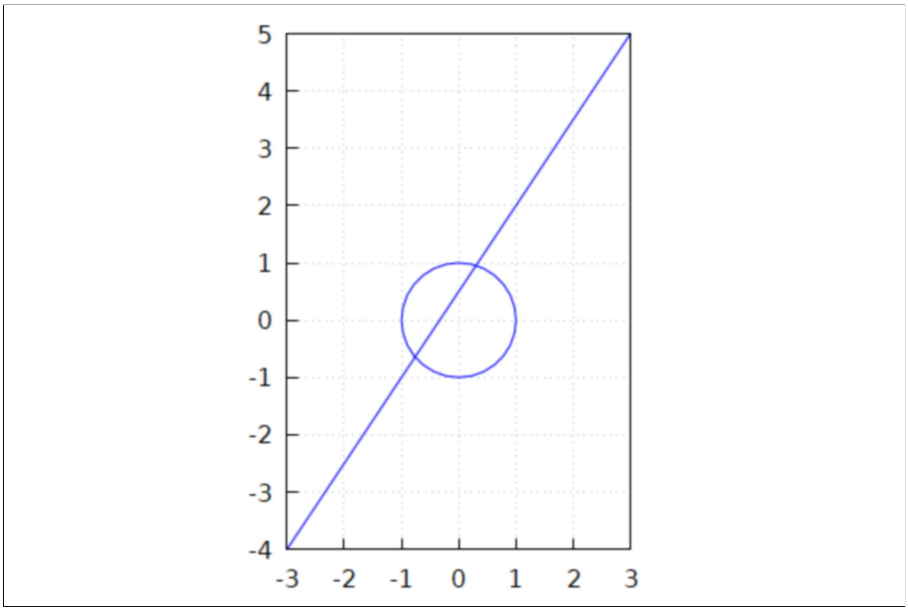
(%i117)
```
wxplot2d([[parametric,1+2·t,2+3·t,[t,−2,1]],[parametric, cos(t),sin(t),[t,0,2·%pi]]], same_xy,grid2d);
wxdraw2d(parametric(1+2·t,2+3·t,t,−2,1),parametric(cos(t),sin(t),t,0,2·%pi),
proportional_axes=xy,
   grid=true);
```

(%t116)



(%o116)

(%t117)



(%o117)

También es posible insertar una curva en paramétricas y la gráfica de una función, segun el formato

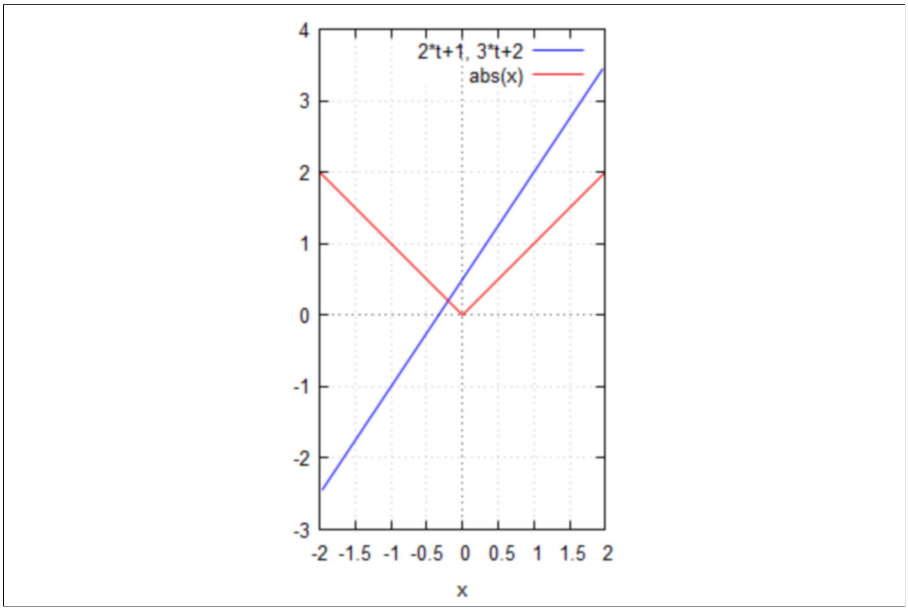
plot2d([[parametrica 1 segun el formato anterio],ley de la funcion],[x, minimo x, maximo x])
draw2d(parametric(curva1),explicit(funcion2))

Vemos en el caso de graficas simultaneas que el comando draw2d comienza a ser más cómodo.

(%i119)
```
wxplot2d([[parametric,1+2·t,2+3·t,[t,−2,1]],a(x)],[x,−2,2], same_xy,grid2d);
wxdraw2d(parametric(1+2·t,2+3·t,t,−2,1),explicit(a(x),x,−2,2),
   proportional_axes=xy,
   grid=true);
```

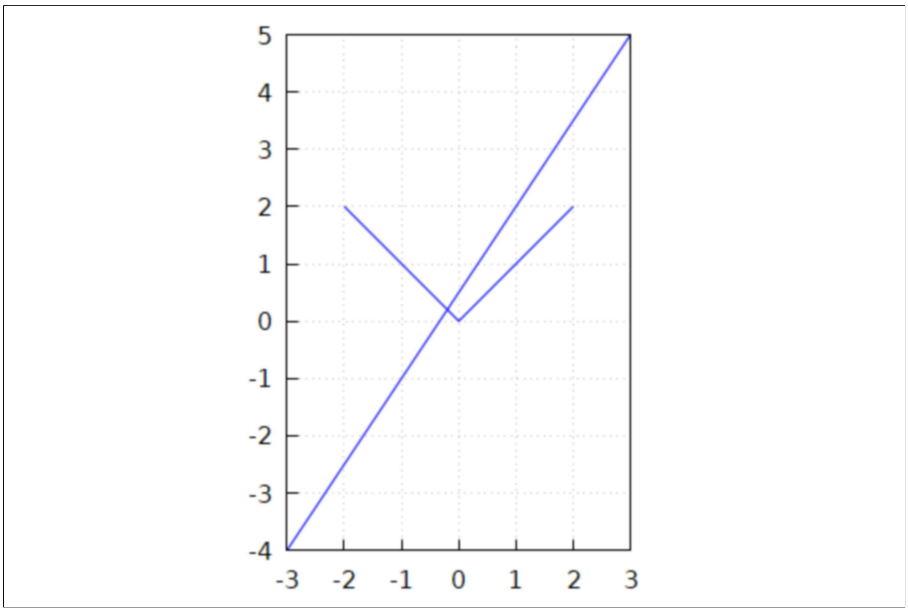
plot2d: some values were clipped.

(%t118)



(%o118)

(%t119)

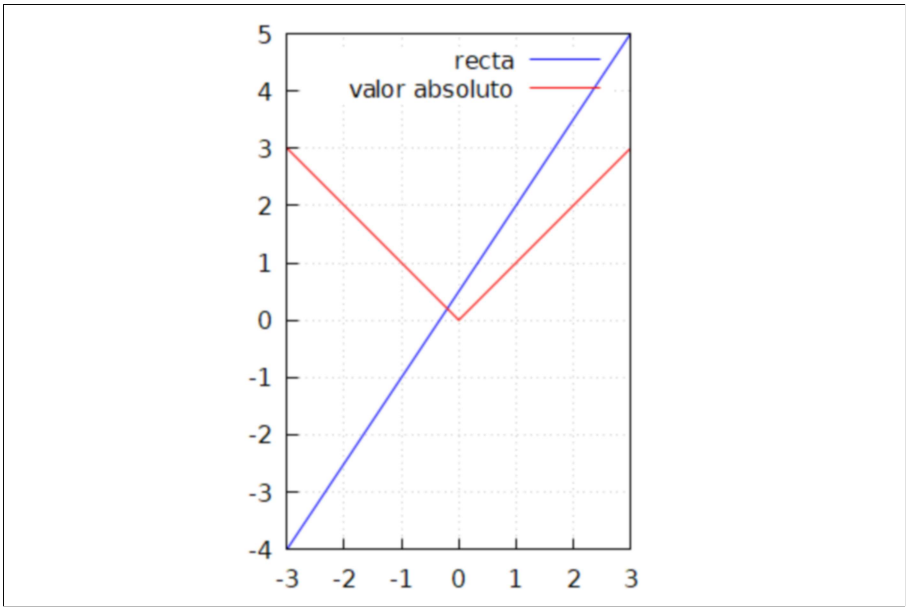


(%o119)

El comando draw permite agregar colores y nombres a las gráficas con los comandos color y key="nombre":

(%i121)
```
wxdraw2d(color=blue, key="recta",parametric(1+2·t,2+3·t,t,−2,1),color=red, key="valor absoluto",e
    proportional_axes=xy,
    grid=true);
```

(%t121)



(%o121)

Finalmente es posible graficar curvas dada a traves de ecuaciones implicitas. En el caso de la recta, la ecuacion implicita es de la forma ax+by+c=0, en el caso de la circunferencia, x^2+y^2=1. En el caso general, es una ecuación de la forma E(x,y)=0.

Para utilizar plot es necesario cargar el paquete implicit_plot.

En draw2d esta forma está contemplada con el comando draw2d(implicit(ecuación,x,xmin,xmax,y,ymin,ymax)) y repetirlo en caso de que se deseen graficar más de una ecuación.

(%i89)
```
load(implicit_plot);
```

(%o89) *C:\maxima−5.42.2\share\maxima\5.42.2\share\contrib\implicit_plot.lisp*

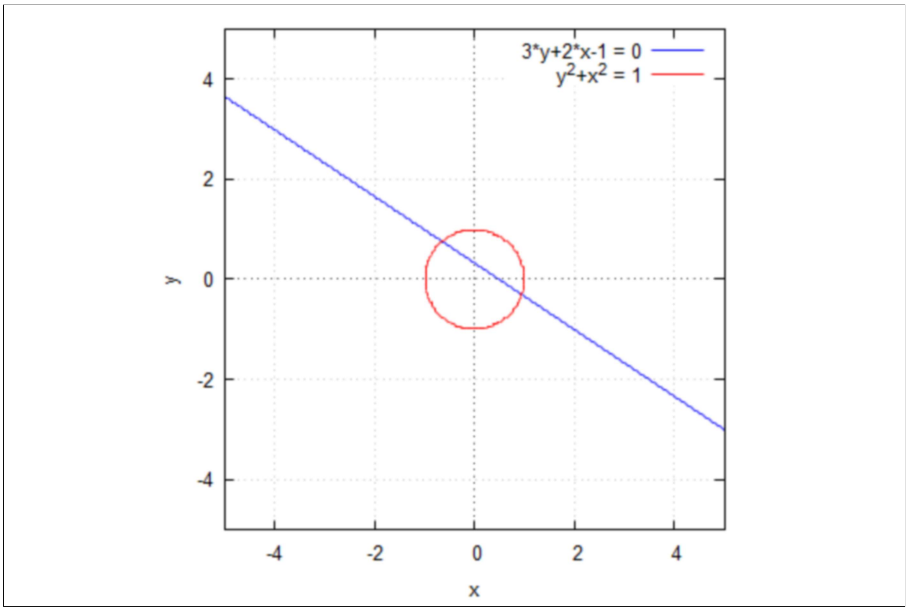
Este formato permite tambien hacer varias graficas a la vez, y el comando es de la forma

implicit_plot[[ecuacion1, ecuacion2, etc], [x, minimo, maximo], [y, minimo maximo]];

Nuevamente es posible agregar wx adelante para que lo inserte en la hoja de trabajo, y los comandos same_xy o grid2d:

(%i90)
```
wximplicit_plot([2·x+3·y−1=0,x^2+y^2=1],[x,−5,5],[y,−5,5],same_xy,grid2d);
```

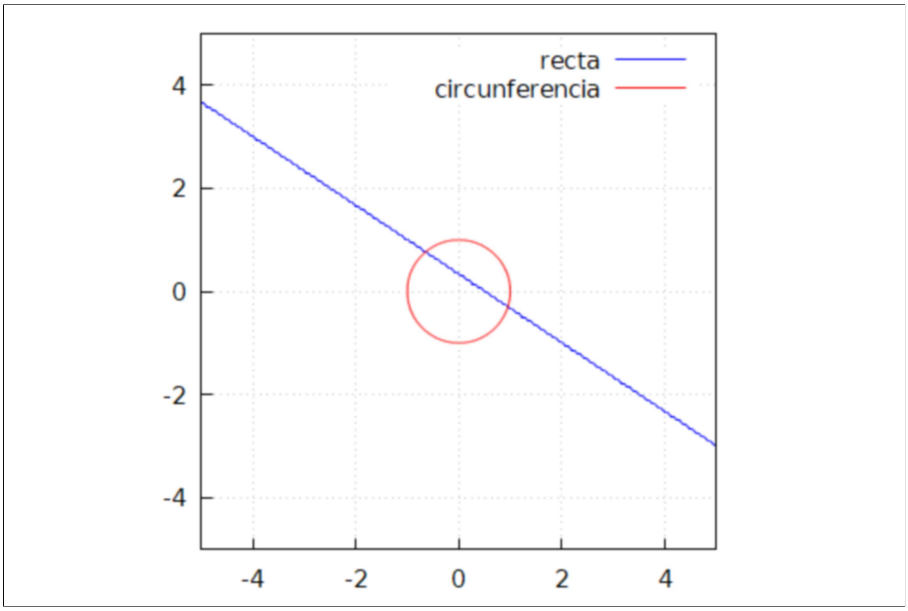
(%t90)



(%o90)

(%i127)
```
wxdraw2d(
    color=blue, key="recta",implicit(2·x+3·y−1=0,x,−5,5,y,−5,5),
    color=red, key="circunferencia",implicit(x^2+y^2=1,x,−1,1,y,−1,1),
proportional_axes=xy,
grid=true);
```

(%t127)



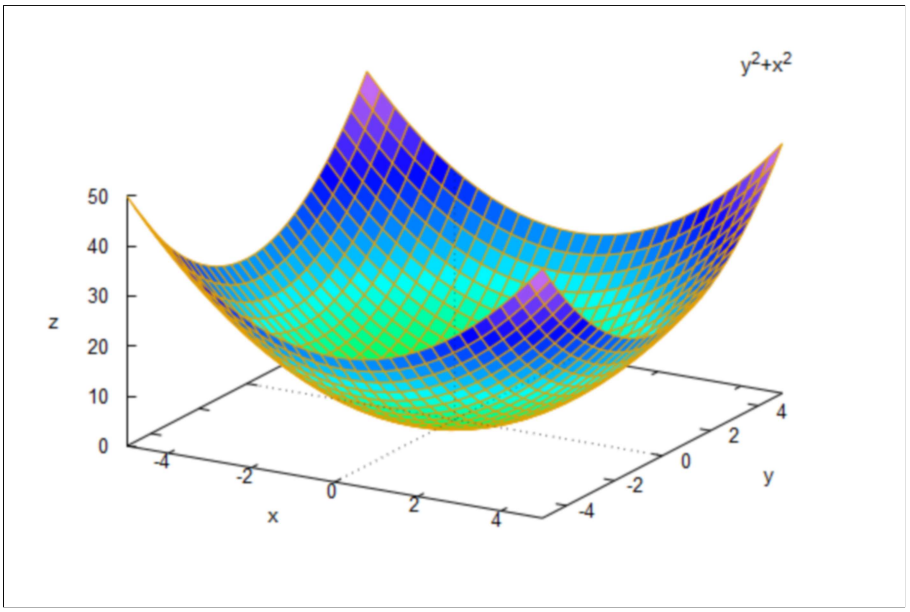
(%o127)

6.2 Gráficas 3d

Es cuando pretendemos trabajar en tres dimensiones (lo haremos más adelante en la materia) que el comando plot resulta insuficiente y en cambio draw es una herramienta muy potente.
Retomaremos estos temas más adelante, veamos por ahora algunos ejemplos. En primer lugar podemos graficar con plot3d una función de dos variables a valores reales como en el ejemplo siguiente:

(%i91)
```
wxplot3d(x^2+y^2,[x,−5,5],[y,−5,5]);
```

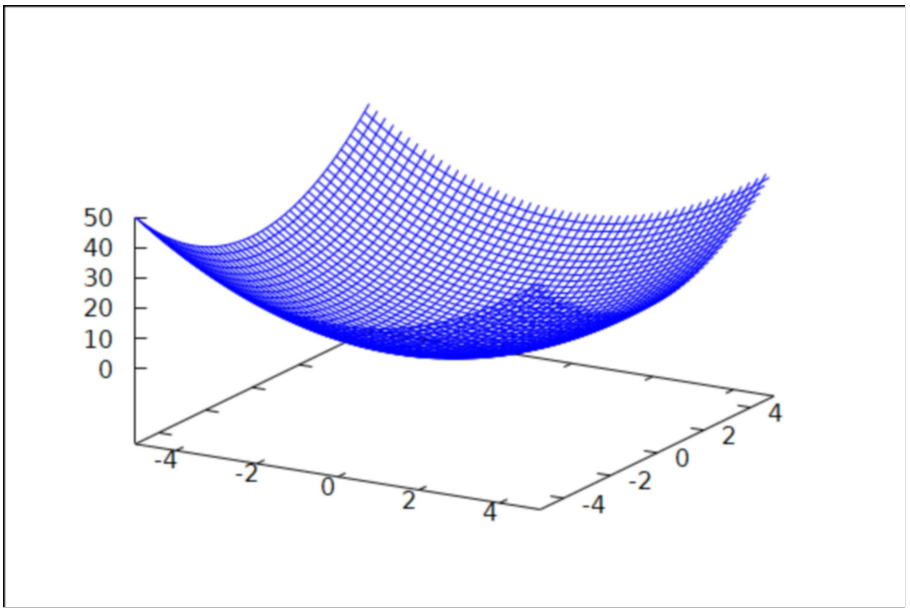
(%t91)



(%o91)

(%i128)
```
wxdraw3d(explicit(x^2+y^2,x,−5,5,y,−5,5));
```

(%t128)



(%o128)

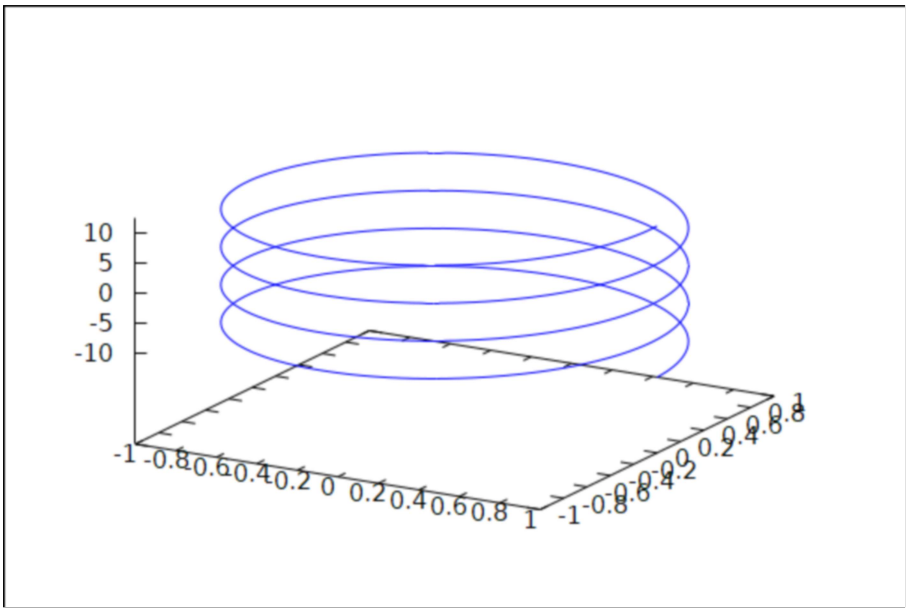
Veremos más adelante qué otras características podemos incluir a la gráfica.

Por el momento sólo observaremos que para otro tipo de representaciones (paramétricas, implícita) draw3d presenta funciones que plot3d no tiene.

Para graficar curvas parametricas en 3d el comando es análogo a 2d.

(%i137)
```
wxdraw3d(nticks=1000,parametric(cos(t),sin(t),t,t,-4·%pi,4·%pi));
```

(%t137)

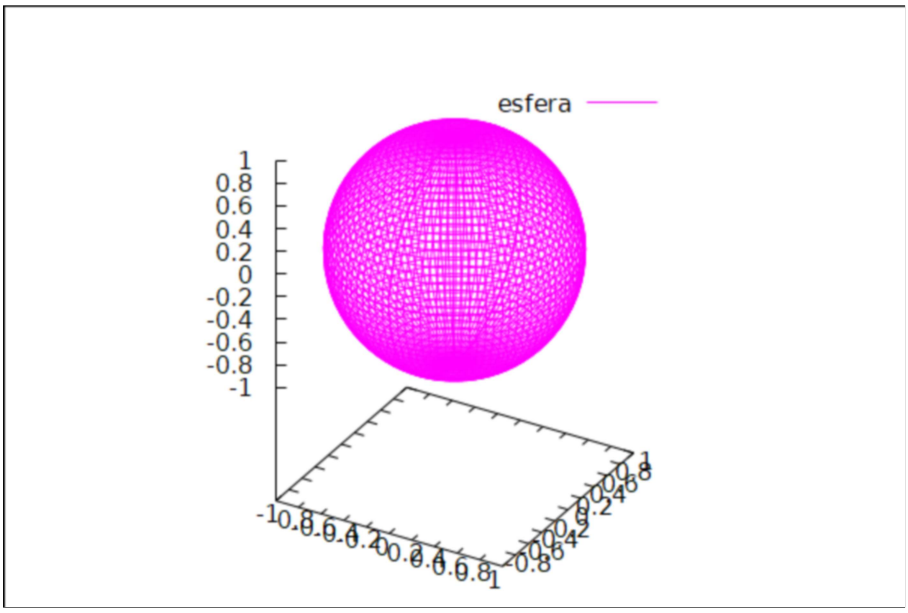


(%o137)

Mostramos solo un ejemplo de como graficar una superficie dada en ecuaciones parametricas o implicitas:

(%i141)
```
wxdraw3d(color=magenta, key="esfera",parametric_surface(cos(t)·sin(s),cos(t)·cos(s),sin(t),t,0,2·%
proportional_axes=xyz);
```

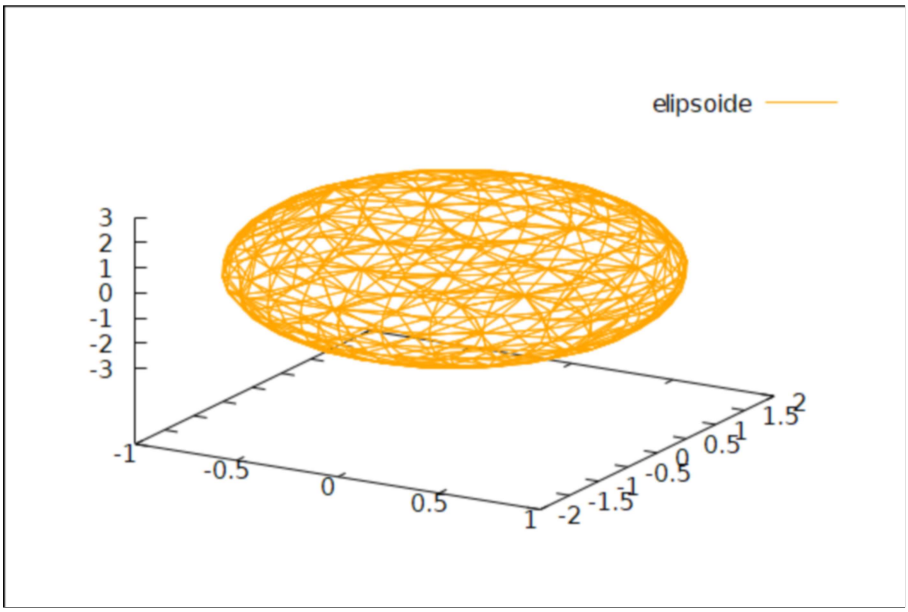
(%t141)



(%o141)

(%i150)
```
wxdraw3d(color=orange,key="elipsoide",implicit(x^2+y^2/4+z^2/9=1,x,−1,1,y,−2,2,z,−3,3));
```

(%t150)



(%o150)