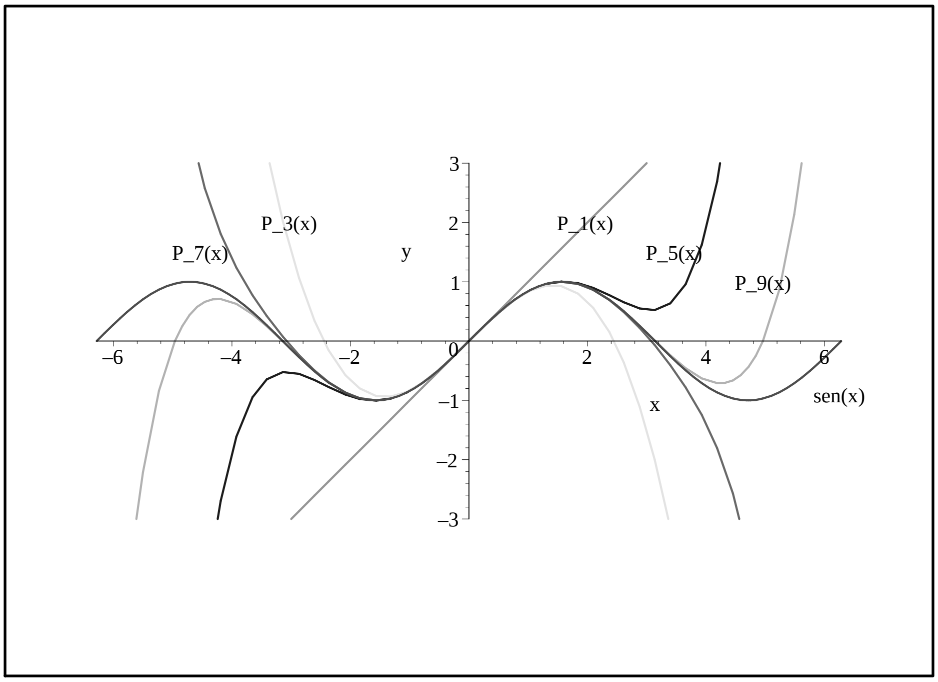
Prácticas de cálculo diferencial en la Ingeniería de Montes

por

Ana Luzón y Pilar Cristobal

.



Índice General

Capítulo 1

Introducción al Maple

1.1 Introducción a las hojas de trabajo

Observa la barra de herramientas: En **File** encontraremos las ordenes necesarias para *abrir* y *guardar* documentos (también representados con iconos más abajo).

Visita el **help** en la barra de herramientas, la *Introducción* y *Matemáticas*.

Observa los iconos de la derecha de hacer y deshacer: el *sumatorio* te pedira una fórmula, la *T* te permitira escribir un texto como este y *[>* te permite abrir una linea para trabajar en Maple en el lugar de la hoja que desees. Dentro de una hoja de trabajo te puedes mover de arriba y abajo pero los comandos siempre se meten en la memoria del ordenador en el orden que los introduzcas independientemente del lugar donde los hayas escrito. Con esto hay que tener cuidado con la asignación de valores a las variables.

Normalmente empezaremos la hoja de trabajo con un **restart** para inicializar las variables.

```
> restart:
```

El texto, en forma automática, aparece en negro, los comandos de Maple que introducimos en rojo y lo que contesta Maple en azul. Por ejemplo:

```
> 2+2;
```

$$4$$

La **sintaxis** de Maple obliga a terminar cualquier linea de comando con ; o : observa la diferencia:

```
> 2+2:
```

1.2 Operaciones básicas

Las operaciones aritméticas habituales se realizan por medio de: suma '+', resta '-', producto '*', división '/' y elevar a un exponente '^'. La jerarquia de las operaciones es la habitual de matem áticas asi que si queremos variarla tendremos que utilizar paréntesis necesariamente.

```
> 2+3;
```

$$5$$

```
> 2-3;
```

$$-1$$

```
> 2*3;
```

$$6$$

```
> 2/3;
```

$$\frac{2}{3}$$

Observar que lo que contesta Maple es el número exacto simbólico, si quisieramos su expresión decimal podríamos hacerlo de varias maneras pero una de ellas consiste en que basta meter uno de los números como decimal y esto basta añadiendole el punto decimal

```
> 2./3;
```

$$.6666666667$$

```
> 2^3;
```

$$8$$

Ejercicio 1

Con la fecha de tu nacimiento: día, mes y año realiza las siguientes operaciones. Suma los tres números, al año restale el día y el mes, divide el año por el producto del día y el mes. Eleva el año al día y todo ello al mes.

Otra cuestión fundamental es la asignación de valores a las variables; si queremos asignarlas para una única línea de comando utilizaremos el símbolo '**=**', si queremos que la asignación a la variable sea para toda la hoja de trabajo hasta que volvamos a introducir el comando restart pondremos el símbolo '**:=**', otra opción es la asignación como una función, de manera que podamos tratarla como tratamos habitualmente a las funciones. Observa las diferencias:

```
> a=3;
```

$$a = 3$$

```
> a;
```

$$a$$

```
> a:=3;
```

$$a := 3$$

```
> a;
```

$$3$$

```
> a:=x->x^2;
```

$$a := x \to x^2$$

```
> a(3);
```

$$9$$

También a la hora de evaluar y utilizar distintos comandos sobre las variables depende de como las hayamos definido. Por ejemplo:

```
> f:=x^2+2;
```

$$f := x^2 + 2$$

```
> subs(x=2,f);
```

$$6$$

```
> g:=x->x^2+2;
```

$$g := x \to x^2 + 2$$

```
> subs(x=2,g);
```

$$g$$

Para funciones el comando subs no funciona, en cambio:

```
> g(2);
```

$$6$$

También Maple tiene ya funciones definidas que podemos utilizar:

```
> sin(Pi/2);
```

$$1$$

```
> cos(Pi);
```

$$-1$$

```
> tan(Pi/4);
```

$$1$$

```
> sqrt(4);
```

$$2$$

```
> log(1);
```

$$0$$

```
> exp(1);
```

$$e$$

Y números especiales

```
> exp(1);
```

$$e$$

```
> I^2;
```

$$-1$$

```
> Pi;
```

$$\pi$$

Si queremos calcular los primeros 30 decimales de π simplemente calculamos

```
> evalf(Pi,30);
```

$$3.14159265358979323846264338328$$

Ejercicio 2

Volvemos a utilizar la fecha de tu nacimiento. Construye una función cuadrática que su coeficiente de segundo grado sea el día, el de primer grado el mes y el término independiente el año. Evaluarla en los números que forman la fecha de hoy: dia, mes y año. Hacerlo de dos maneras, mediante una función que llamaremos funcion1 y mediante una variable que llamaremos variable1 para toda la hoja de trabajo.

1.3 Primeros comandos

Vamos a dar unos cuantos comandos para empezar a trabajar, pero iremos añadiendo según vayamos necesitando. Por ejemplo para definir una función a trozos

```
> f1:=piecewise(x<=-1,3*x+3,-1<x and
> x<1,(a*x^3-b*x^2+c*x-d)/(1+x^2),x>=1,-2*x+3);
```

Para derivar e integrar funciones

```
> diff(sin(x),x);
```

$$\cos(x)$$

```
> diff(3*y^4-5*y^2+3,y);
```

$$12\,y^3 - 10\,y$$

```
> Int(sin(x),x)=int(sin(x),x);
```

$$\int \sin(x)\,dx = -\cos(x)$$

Observar que el comando **Int** es un comando inerte, es decir, representa la integral simbólicamente pero no la calcula, en cambio el comando **int** si que la calcula. Observar que la constante de integración que pone por defecrto Maple es C=0, esto hay que tenerlo en cuenta. También se pueden calcular integrales definidas:

```
> int(x^2,x=0..1);
```

$$\frac{1}{3}$$

Ejercicio 3

(a) Haz una tabla con las derivadas de las funciones elementales principales

(b) Haz una tabla con las integrales inmediatas principales.

También podemos resolver ecuaciones

```
> ecu1:=a*x^2+b*x+c=0;
```

$$ecu1 := a\,x^2 + b\,x + c = 0$$

```
> solve(ecu1,{x});
```

$$\{x = \frac{1}{2}\,\frac{-b+\sqrt{b^2-4\,a\,c}}{a}\},\ \{x = \frac{1}{2}\,\frac{-b-\sqrt{b^2-4\,a\,c}}{a}\}$$

O un sistema

```
> eq1:=-1/2*a-1/2*b-1/2*c-1/2*d=0;
```

$$eq1 := -\frac{1}{2}\,a - \frac{1}{2}\,b - \frac{1}{2}\,c - \frac{1}{2}\,d = 0$$

```
> eq2:=1/2*a-1/2*b+1/2*c-1/2*d=1;
```

$$eq2 := \frac{1}{2}\,a - \frac{1}{2}\,b + \frac{1}{2}\,c - \frac{1}{2}\,d = 1$$

```
> solve({eq1,eq2},{a,b,c,d});
```

$$\{b = -1 - d,\ c = c,\ d = d,\ a = 1 - c\}$$

Dentro del núcleo principal de Maple estan los comandos principales pero si necsitamos algunos específicos necesitaremos cargar algunas librerias donde estan estos comandos. En general nosotros vamos a utilizar paquetes como **student**, **plots**, **linalg**, etc. En particular este ultimos sirve para comandos utilizados en **algebra lineal**

```
> with(linalg):
```

```
Warning, the protected names norm and trace have been redefined and
unprotected
```

Definimos una matriz

```
> A:=matrix(4,4,[1,2,3,3,4,-2,3,-1,5,6,5,8,6,7,8,9]);
```

$$A := \begin{bmatrix} 1 & 2 & 3 & 3 \\ 4 & -2 & 3 & -1 \\ 5 & 6 & 5 & 8 \\ 6 & 7 & 8 & 9 \end{bmatrix}$$

Podemos calcular el determinante de la matriz A;

```
> det(A);
```

$$-61$$

Y su rango

```
> rank(A);
```

$$4$$

También podemos definir vectores

```
> b:=vector([1,2,3,4]);
```

$$b := [1, 2, 3, 4]$$

Y resolver el sistema AX=b

```
> linsolve(A,b);
```

$$\left[\frac{23}{61}, \frac{4}{61}, \frac{12}{61}, \frac{-2}{61}\right]$$

Pero también puede ocurrir que la solución **dependa de un par ámetro**

```
> A1:=matrix(3,3,[1,2,3,4,5,6,7,8,9]);
```

$$A1 := \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

```
> b1:=vector([1,2,3]);
```

$$b1 := [1, 2, 3]$$

```
> linsolve(A1,b1);
```

$$\left[-\frac{1}{3} + \_t_1, \frac{2}{3} - 2\,\_t_1, \_t_1\right]$$

También podemos resolver ecuaciones matriciales como AX=B

```
> B:=matrix(4,4,[1,2,-2,3,4,3,-6,-1,5,3,4,5,6,7,8,9]);
```

$$B := \begin{bmatrix} 1 & 2 & -2 & 3 \\ 4 & 3 & -6 & -1 \\ 5 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 \end{bmatrix}$$

```
> linsolve(A,B);
```

$$\begin{bmatrix} 1 & \frac{11}{61} & \frac{84}{61} & \frac{-24}{61} \\ 0 & \frac{108}{61} & \frac{614}{61} & \frac{147}{61} \\ 0 & \frac{80}{61} & \frac{12}{61} & \frac{75}{61} \\ 0 & \frac{-115}{61} & \frac{-490}{61} & \frac{-104}{61} \end{bmatrix}$$

También podemos factorizar, desarrollar o simplificar expresiones:

```
> expand(sin(2*x));
```

$$2\sin(x)\cos(x)$$

```
> expand((x-1)*(x-2)*(x-3));
```

$$x^3 - 6\,x^2 + 11\,x - 6$$

```
> factor(x^3-6*x^2+11*x-6);
```

$$(x-1)\,(x-2)\,(x-3)$$

```
> f3 := sinh(x);
```

$$f3 := \sinh(x)$$

```
> convert(f3, exp);
```

$$\frac{1}{2}\,e^{x}-\frac{1}{2}\,\frac{1}{e^{x}}$$

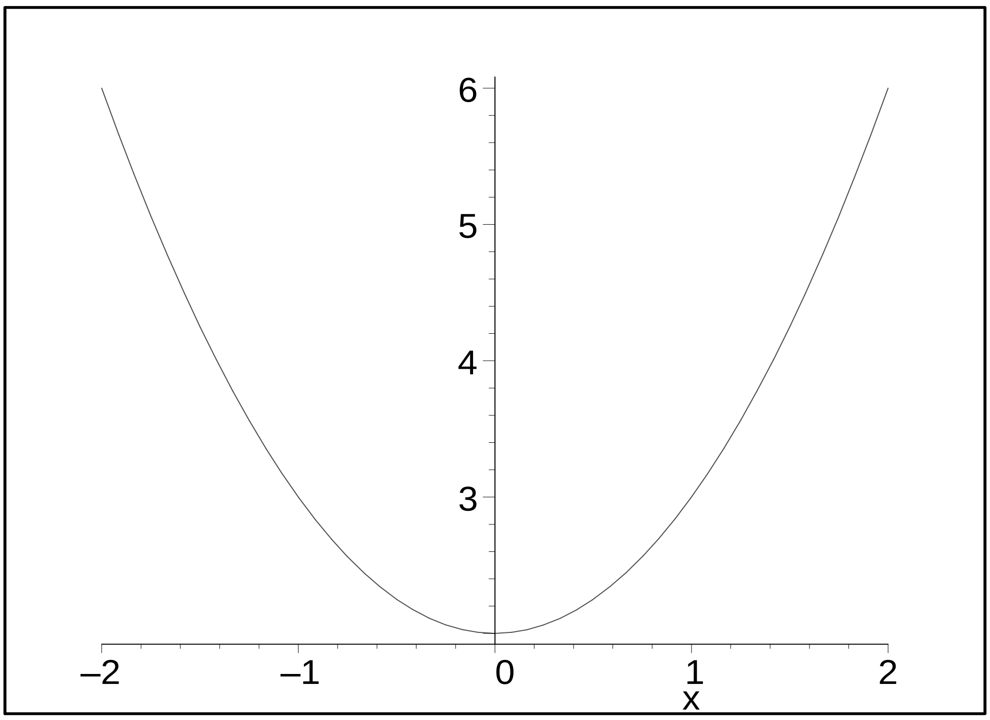
Ejercicio 4

Haz una tabla con las expresiones trigonométricas más frecuentes.

1.4 Primeras gráficas

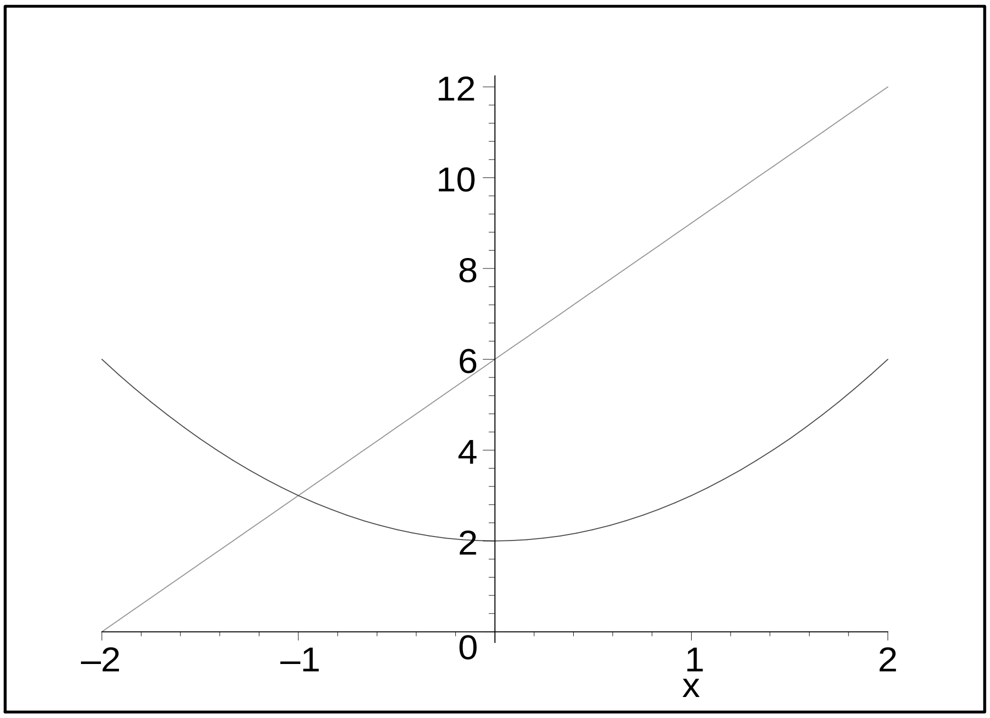
Quizá uno de las aplicaciones principales de cualquier paquete de c álculo matemático es la representación gráfica. Para las gráficas más sencillas basta con el núcleo principal pero muchas veces necesitaremos cargar el paquete **plots** que tiene comandos más especificos para las gráficas.

```
> plot(f,x=-2..2);
```



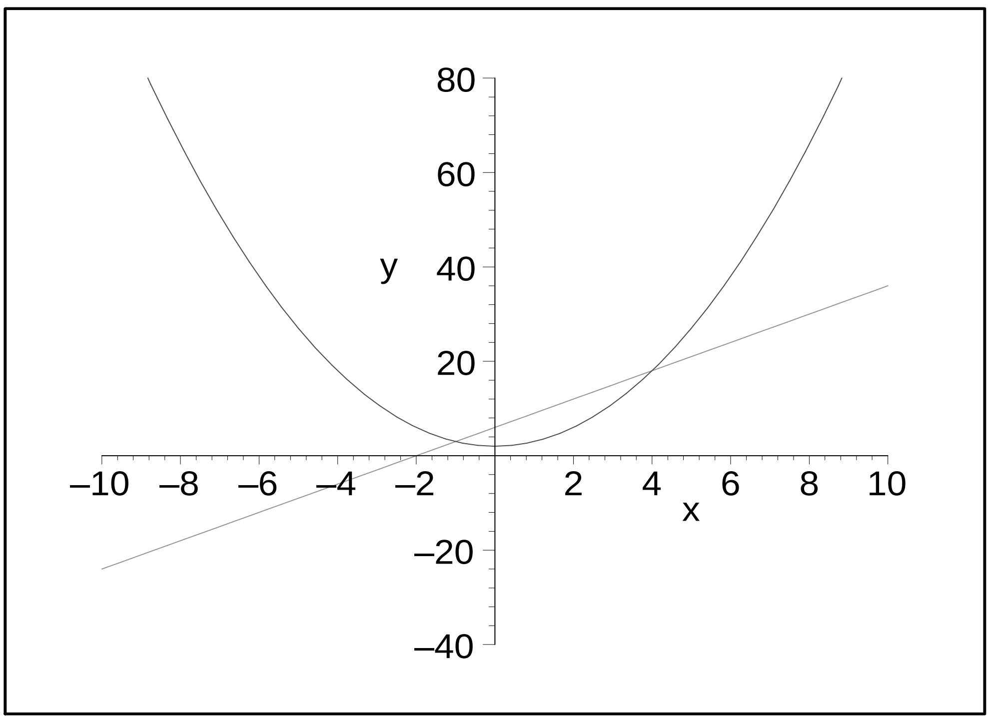
Tambien podemos pintar dos funciones a la vez

```
> g:=3*x+6:
> plot([f,g],x=-2..2);
```



Observar que en particular si queremos ver graficamente que la ecuaci ón f=g tiene dos soluciones debemos cambiar, en este caso ampliar, la ventana que aparece el gráfico, es decir los intervalos de la x y la y

```
>  plot([f,g],x=-10..10, y=-40..80);
```



Ejercicio 5

Representa gráficamente las funciones del ejercicio 3

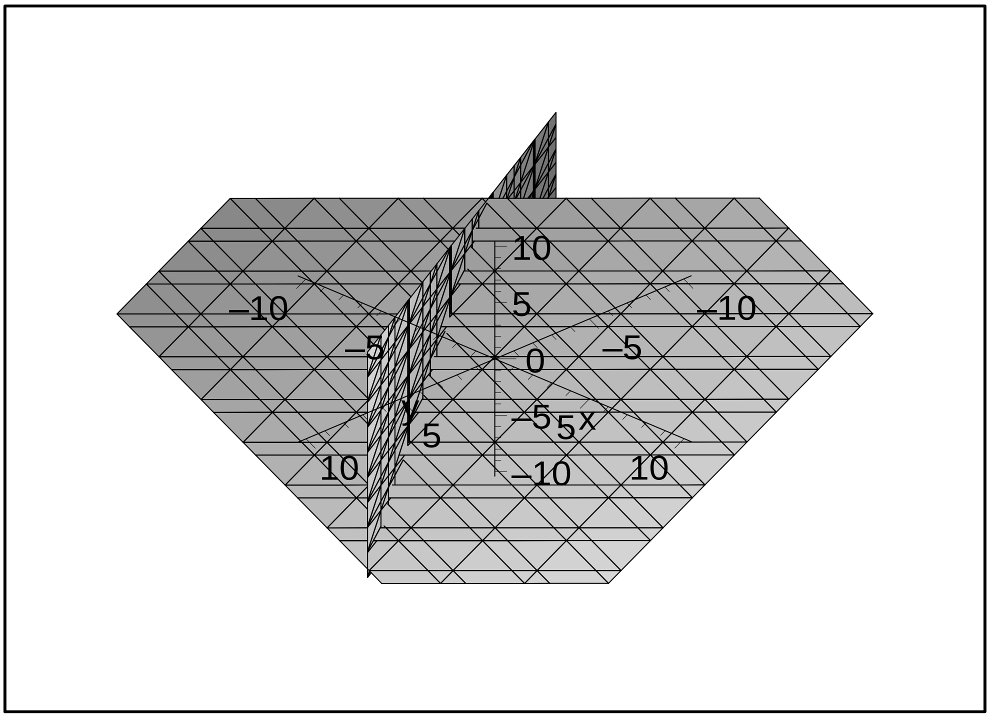
Cargamos el paquete **plots**

```
> with(plots):
```

```
Warning, the name changecoords has been redefined
```

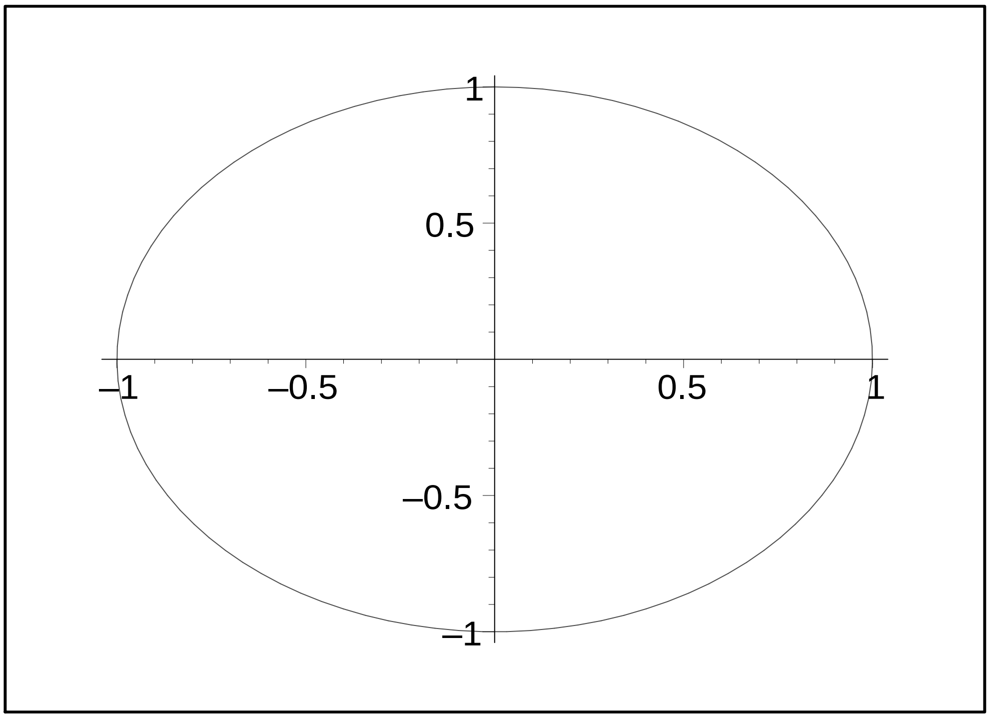
Ahora pintaremos un par de planos

```
> implicitplot3d({x+y+z= 4,
> 2*x-4*y=7},x=-10..10,y=-10..10,z=-10..10,axes=normal);
```



Hacemos una animación

```
> animate([(t/14)*sin(u),(t/14)*cos(u),u=-Pi..Pi],
> t=-14..14,frames=100,numpoints=100);
```


1
0.5
–1
–0.5
0.5
1
–0.5
–1

Capítulo 2

Representación gráfica

2.1 Curvas en el plano

Representación de curvas explícitas.

```
> restart:
```

Definimos varias expresiones explicitas

```
> f:=x+1:
```

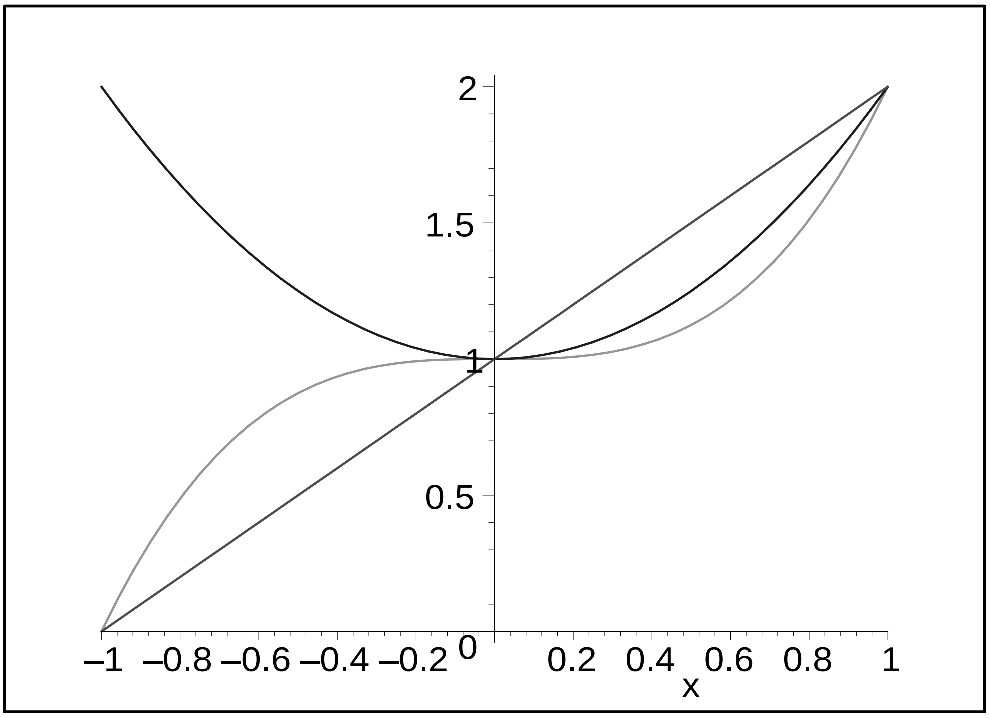
```
> g:=x^2+1:
```

```
> h:=x^3+1:
```

Las representamos . Observamos que si no le ponemos a x valores entre los que varia por defecto Maple nos lo representa entre -10 y 10. También podemos añadir entre que valores queremos que vaya la y de manera similar a la x. Añadimos las opciones de **color** y **thickness** al comando plot y observamos su funcionamento.

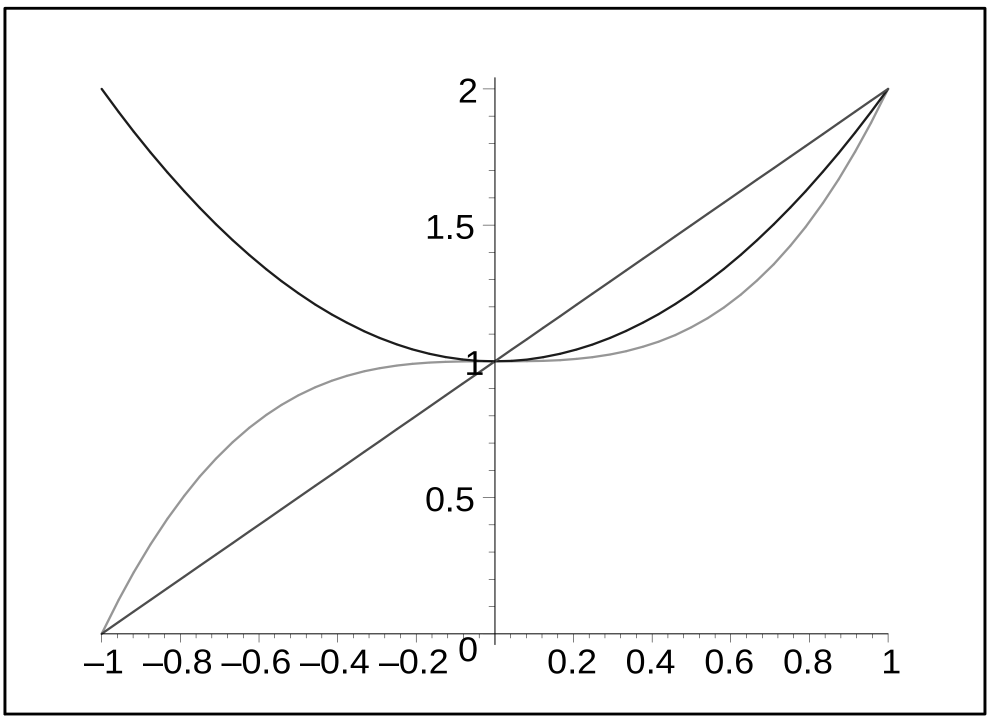
Ejercicio 1: Varia en la gráfica , el intervalo de x, sin poner nada, añade valores para la y, varia los colores y el valor de thickness y observa lo que hace.

```
> plot([f,g,h],x=-1..1, color=[red, blue, green],thickness=3);
```



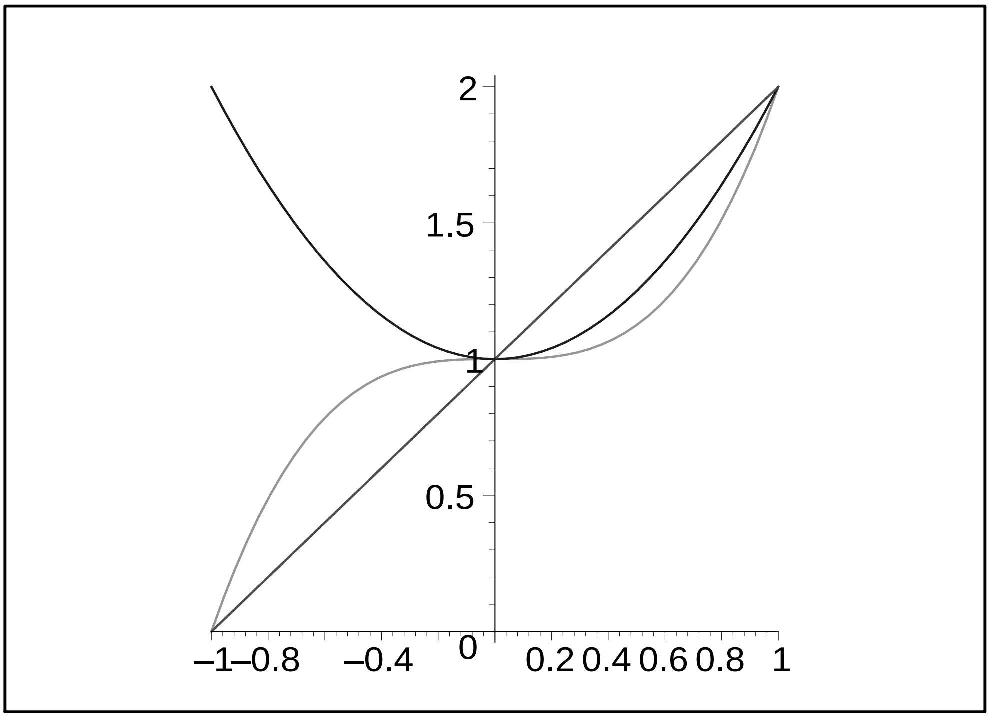
Si definimos funciones el comando plot es algo distinto, observa la diferencia.

```
> funcf:=x->x+1:
> funcg:=x->x^2+1:
> funch:=x->x^3+1:
> plot([funcf,funcg,funch],-1..1,color=[red,blue,green],thickness=3);
```



Por defecto el eje y se ajusta a la gráfica que estemos representando y, en general, con una escala distinta al eje y, si queremos que se utilice la misma escala tendremos que incluir la orden **scaling=constrained.** Observar la diferencia.

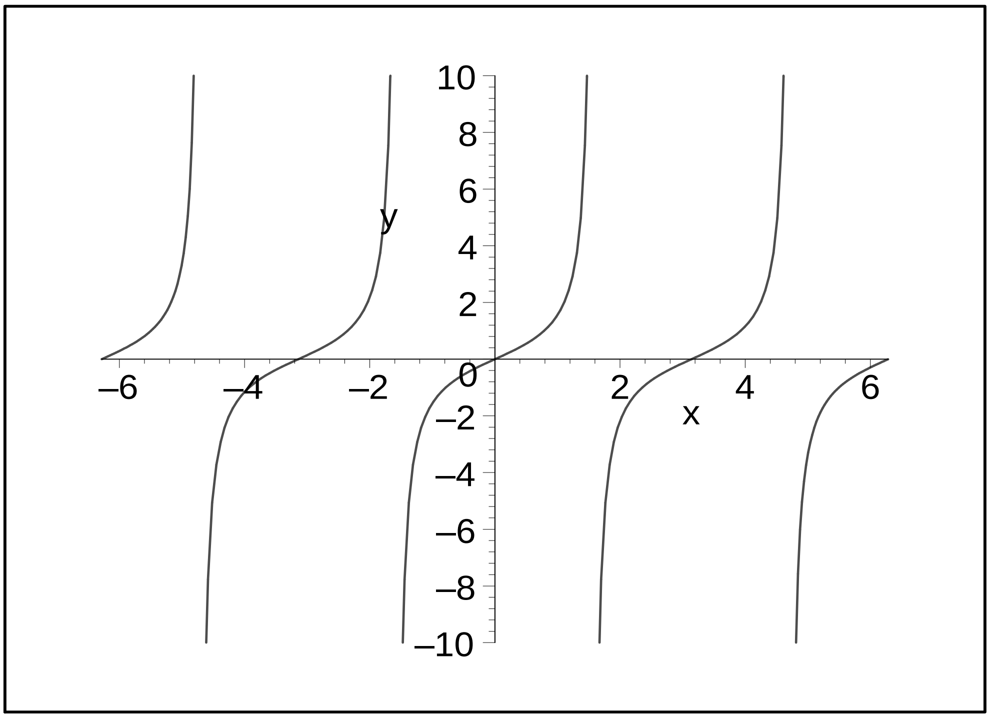
```
> plot([funcf,funcg,funch],-1..1,color=[red,blue,green],thickness=3,sca
> ling=constrained);
```



A veces la gráfica de la función tiene alguna discontinuidad, en este caso es conveniente utilizar el comando **discont=true.**

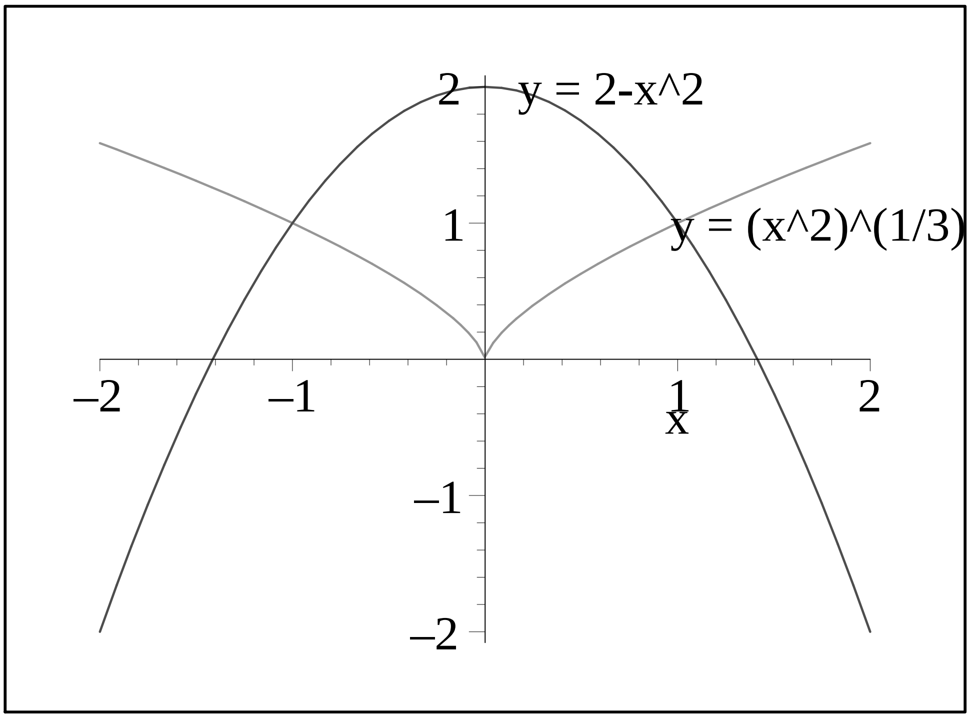
Ejercicio 2: Pinta esta siguiente gráfica con y sin esta opción.

```
> plot(tan(x),x=-2*Pi..2*Pi,y=-10..10,discont=true, thickness=3);
```



También a veces querremos dentro de la gráfica escribir textos, cambiar fuentes, etc y para ello utilizaremos el comando display que nos permite definir diferences objetos y luego representarlos todos juntos.

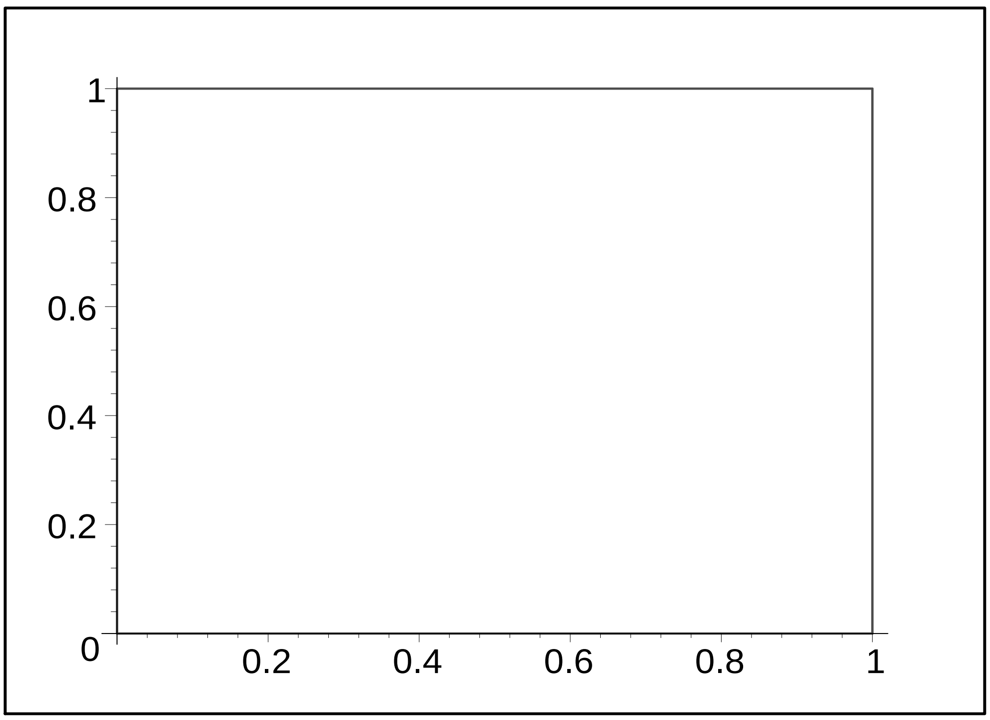
```
> restart:

> with(student):

> with(plots):
Warning, the name changecoords has been redefined

> G:=plot([2-x^2,(x^2)^(1/3)],x=-2..2,thickness=3):

> t1:=textplot([0.7,2,'y=2-x^2']):

> t2:=textplot([1.8,1.,'y=(x^2)^(1/3)']):

> display([G,t1,t2],font=[TIMES,ROMAN, 14]);
```



Ejercicio 3: Varia los valores y observa como funciona la gráfica.

También podemos representar polígonos simplemente definiendo sus vértices:

```
> plot([[0,0],[0,1],[1,1],[1,0],[0,0]],thickness=3);
```



Ejercicio 4: Pintar un triángulo.

Representación de curvas implícitas.

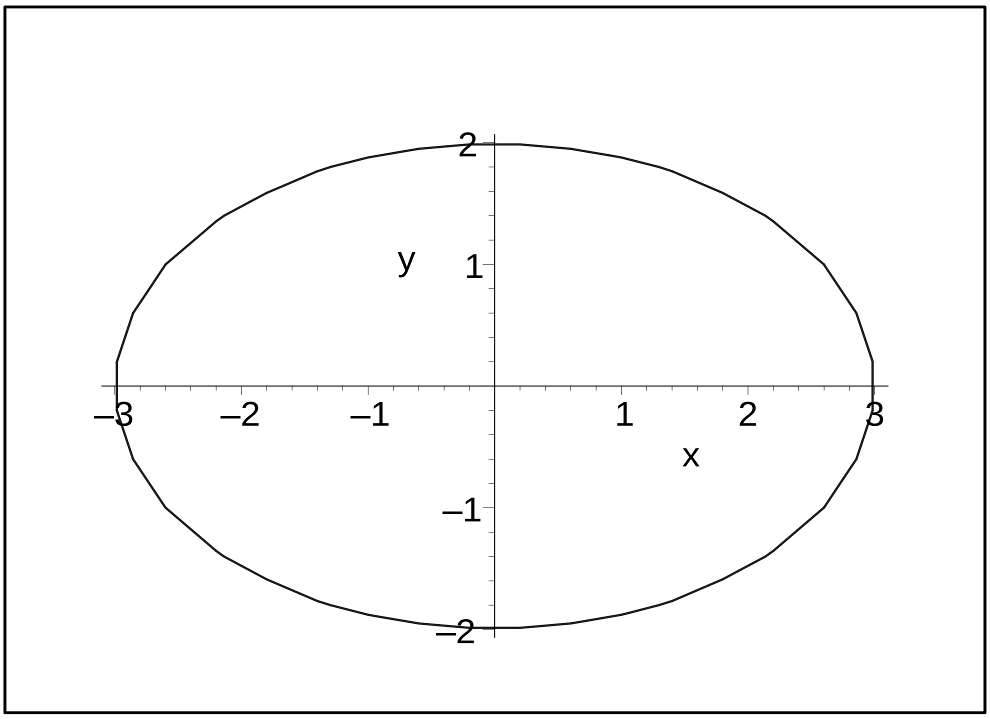
Decimos que una ecuación esta en forma implícita cuando tenemos todos los valores de las variables al mismo lado de la igualdad. Por ejemplo la ecuación de la recta siguiente es en forma implícita. ax+by+c=0. Cuando queremos pintar una curva en forma implicita en Maple necesitamos utilizar el comando **implicitplot** de la libreria **plots.** Vamos a pintar curvas de segundo grado. Y le ponemos titulo.

```
>  restart:
```

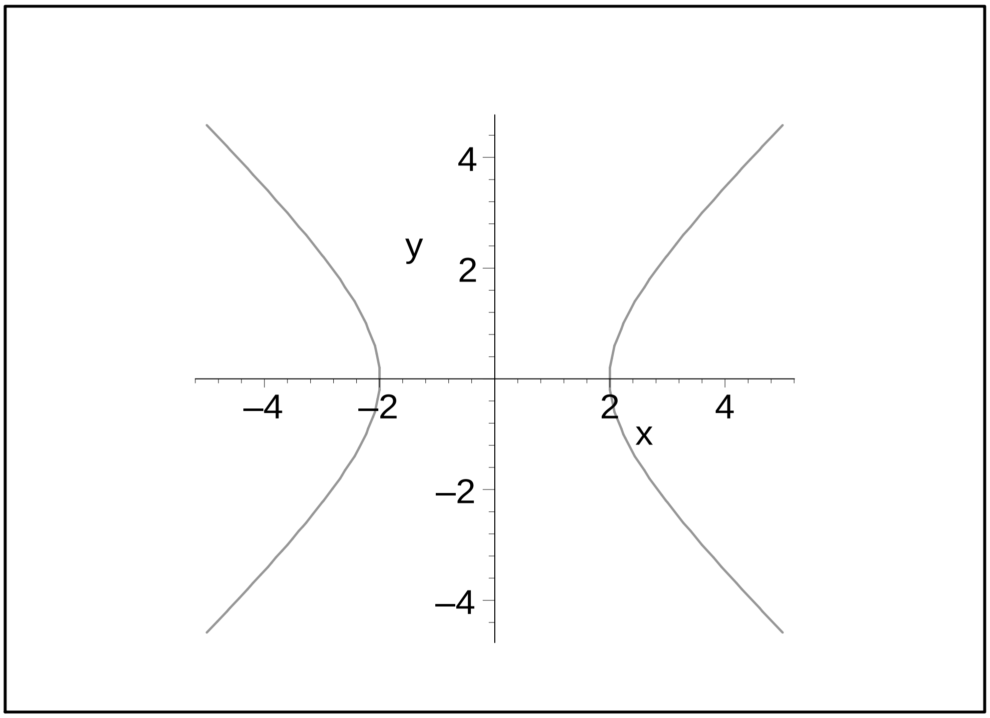
```
>  with(plots):
```

```
Warning, the name changecoords has been redefined
```

```
>  implicitplot(x^2/9+y^2/4=1,x=-5..5,y=-5..5, scaling=constrained,
>  thickness=3, color=blue, title='Elipse');
```




```
>  implicitplot(x^2/4-y^2/4=1,x=-5..5,y=-5..5, scaling=constrained,
>  thickness=3, color=green, title='Hiperbola');
```



Ejercicio 5: Pinta, con expresiones en forma implicita, una recta que tenga de pendiente 2 y pase por el punto (1,2), una parabola que tenga el vértice en el punto (1,1) y una circunferencia de centro (2,3) y de radio 1.5.

Representación de curvas en paramétricas.

Igual que las rectas se pueden dar con diferentes tipos de fórmulas: explícitas, impilicitas y dependientes de un parámetro, también las curvas en el plano y en el espacio se pueden representar paramétricamente. Cargamos el paquete **plots**, para tener a nuestra disposición todos los comandos necesarios para pintar.

```
> restart:
```

```
> with(plots):
```

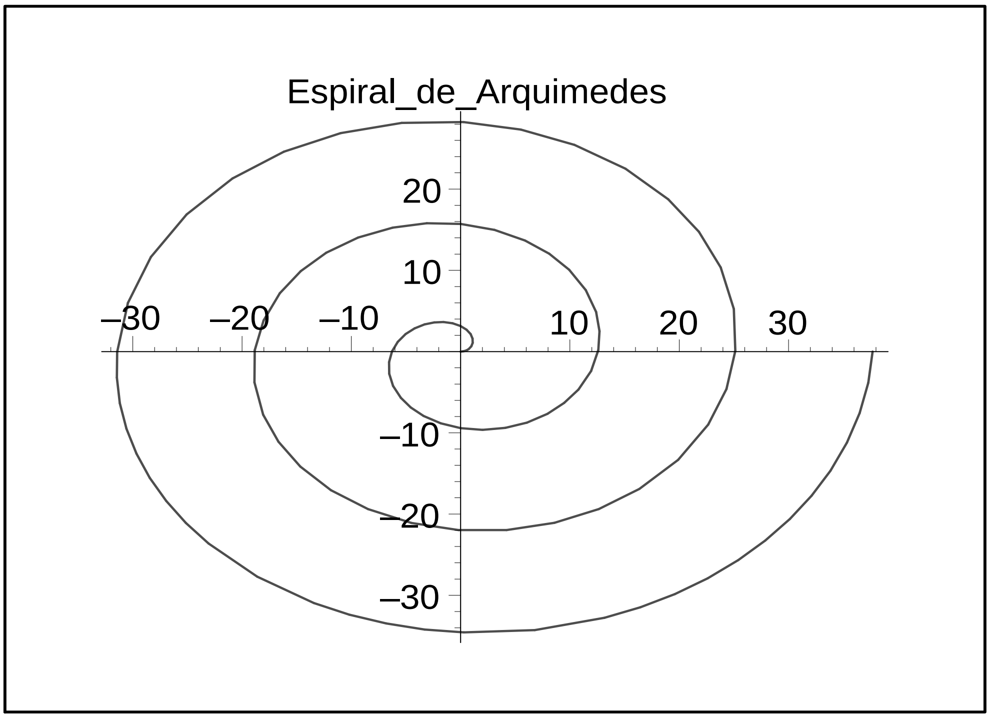
```
Warning, the name changecoords has been redefined
```

Un ejemplo en el plano puede ser la **Espiral de Arqu ímedes:**

x(t)=2*t*cos(t)

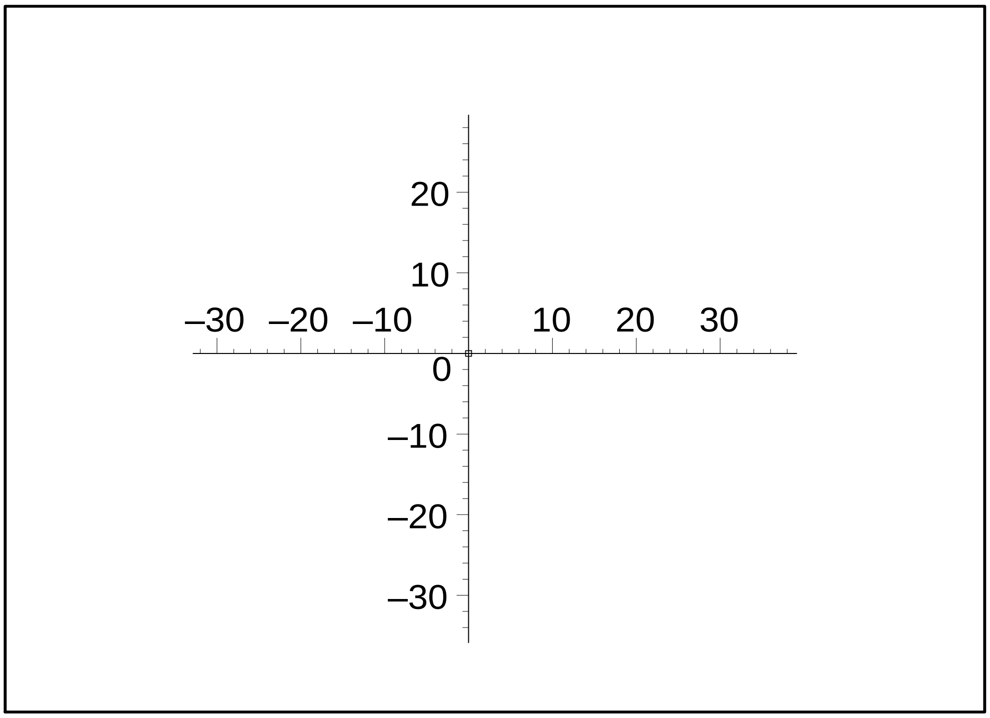
y(t)=2*t*sin(t)

```
> plot([2*t*cos(t),2*t*sin(t),t=0..6*Pi],title='Espiral_de_Arquimedes',
> thickness=3);
```



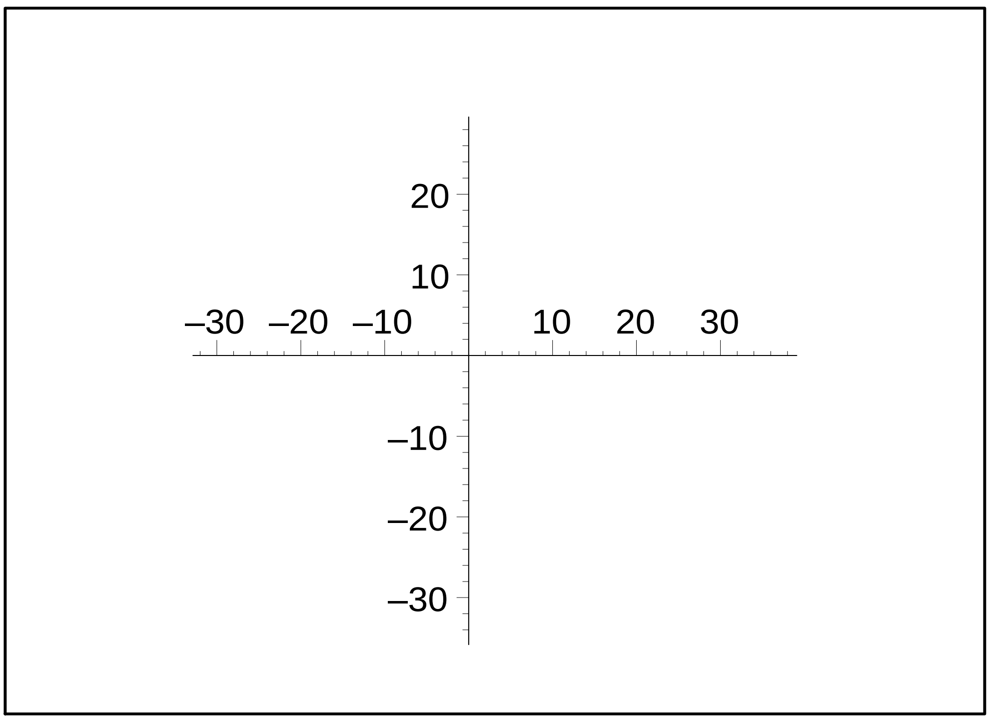
Cuando una curva esta dada en forma paramétrica puede pensarse que el parámetro representa el tiempo. Por ejemplo podemos pensar que es el movimiento de una cierta partícula en el plano (o en el espacio). Supongamos que esto es asi en el caso de la curva anterior, y lo que vamos a hacer es ver este movimiento en acción mediante una animación:

```
> animate([2*t*cos(t),2*t*sin(t)],t=0..6*Pi,style=point,symbol=box,titl
> e='Espiral_de_Arquimedes',thickness=10,scaling=constrained,
> frames=100, color=blue);
```



Si ademas queremos que se quede la curva dibujada haremos:

```
>  animatecurve([2*t*cos(t),2*t*sin(t),t=0..6*Pi],title='Espiral_de_Arqu
>  imedes',thickness=3,scaling=constrained);
```



Ejercicio 6. (a) Pinta una circunferencia en paramétricas.

(b) Pinta una animación que describa una circunferencia.

(c) Pinta una animación que describa una cicloide.

2.2 Curvas en el espacio

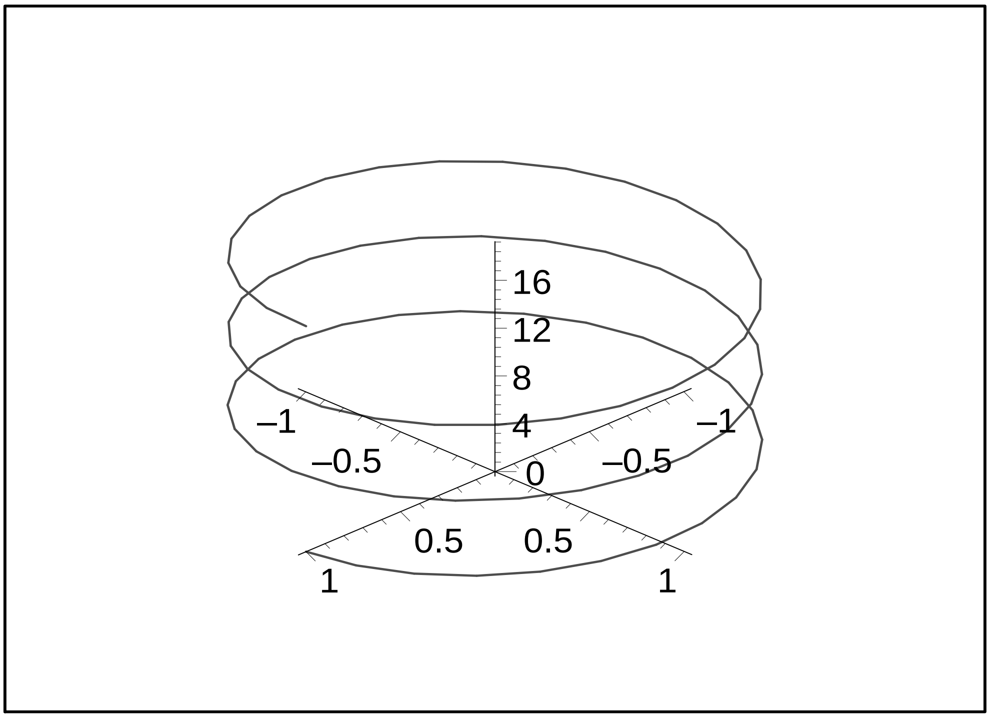
```
> restart:
```

```
> with(plots):
```

```
Warning, the name changecoords has been redefined
```

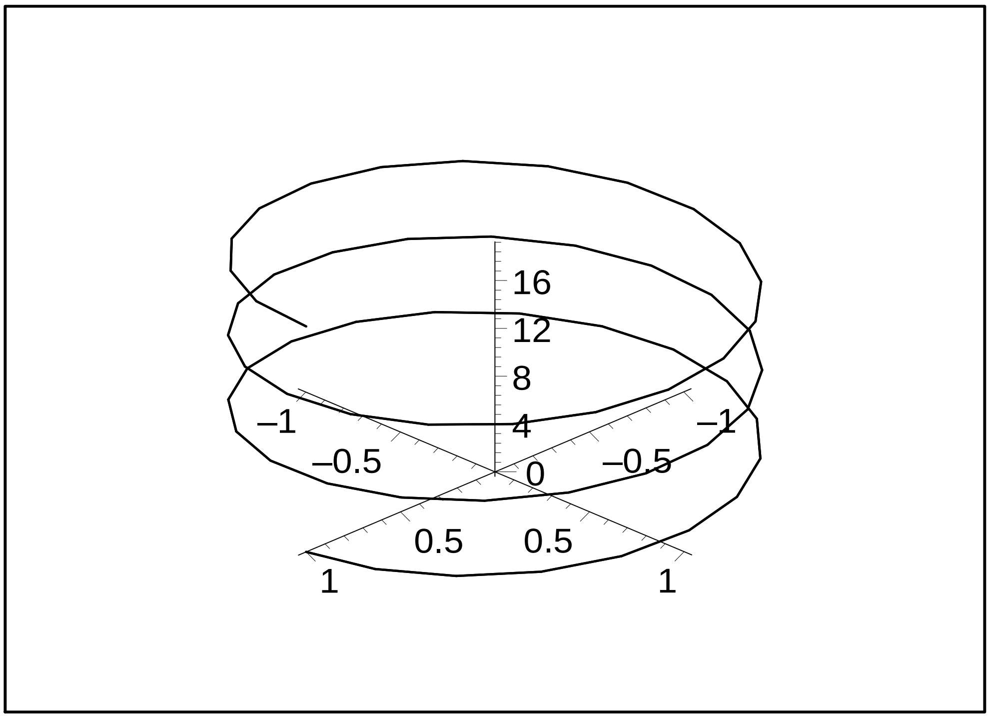
Si lo que queremos es pintar una**curva en paramétricas** pero en el espacio utilizamos el comando **spacecurve.** Aquí numpoints=80 nos sirve para alisar la curva y que no sea demasiado poligonal.

```
> spacecurve([cos(t),sin(t),t],t=0..6*Pi,color=red,thickness=3,axes=NOR
> MAL,numpoints=80);
```

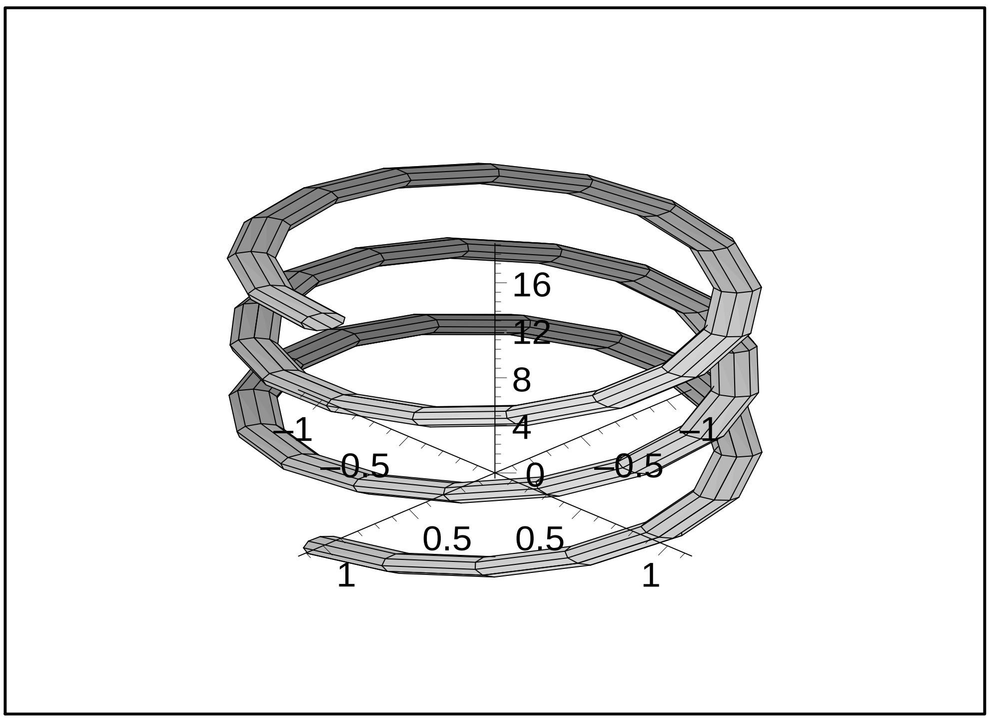


O También con el comando**plot3d.** Aquí utilizamos grid=[60,2] para alisar la curva, 60 para la t y 2 para la s. El mínimo que se puede poner. También se puede utilizar el comando**tubeplot,** con sus opciones correspondientes que dibuja un tubo entorno a la curva.

```
> plot3d([cos(t),sin(t),t],t=0..6*Pi,s=0..1,color=red,thickness=3,axes=
> NORMAL,grid=[60,2]);
```




```
> tubeplot([cos(t),sin(t),t],t=0..6*Pi,radius=0.1,axes=NORMAL);
```



Ejercicio 7: Varia los valores de las opciones y observa como varia la figura.

Pinta una espiral en el espacio que empiece en el (0,0,0) y vaya aumentando su radio.

2.3 Superficies

Superficies en forma explícita

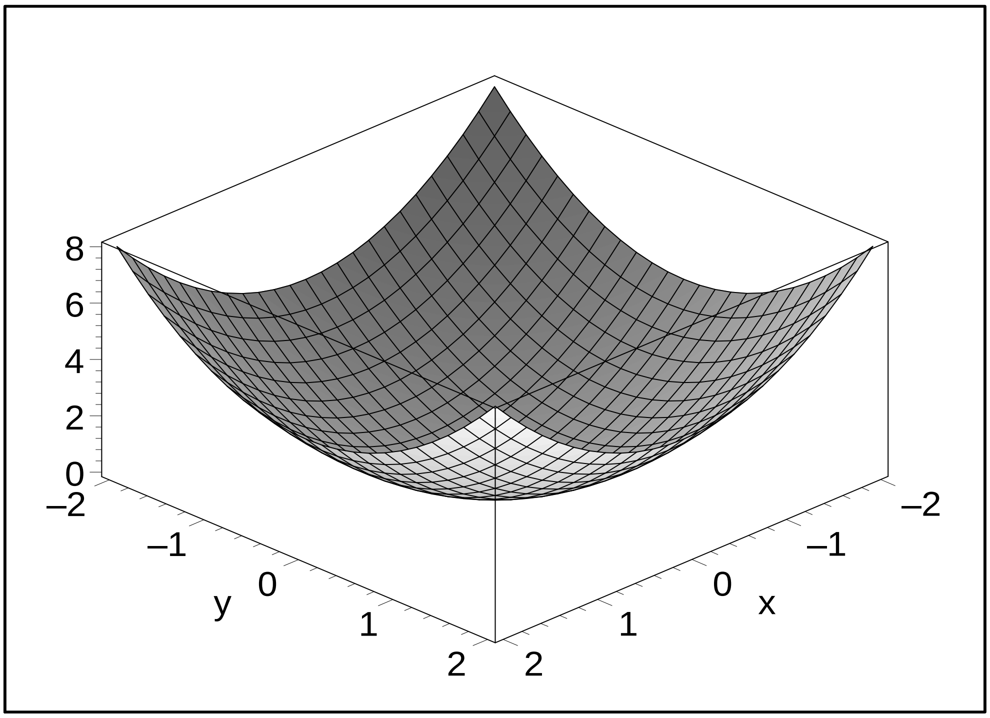
De manera similar al comando plot, esta el comando que pinta superficies el el espacio, dicho comando es el **plot3d, ahora será necesario poner el límite para la x y la y. Recordar que las figuras en 3 dimensiones podemos pincharlas y movelar. Estudia la barra de herramientas cuando pinchas la figura.**

```
> restart:
```

```
> with(plots):
```

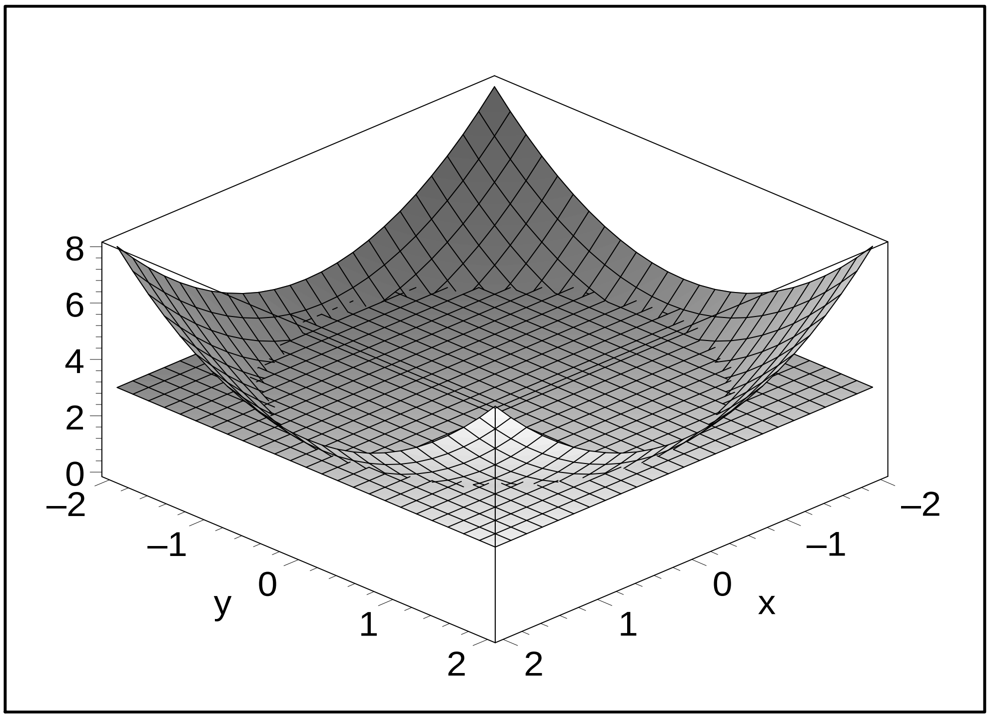
```
Warning, the name changecoords has been redefined
```

```
> plot3d(x^2+y^2,x=-2..2,y=-2..2,axes=boxed);
```



Ahora pintamos la misma superficie cortada por un plano

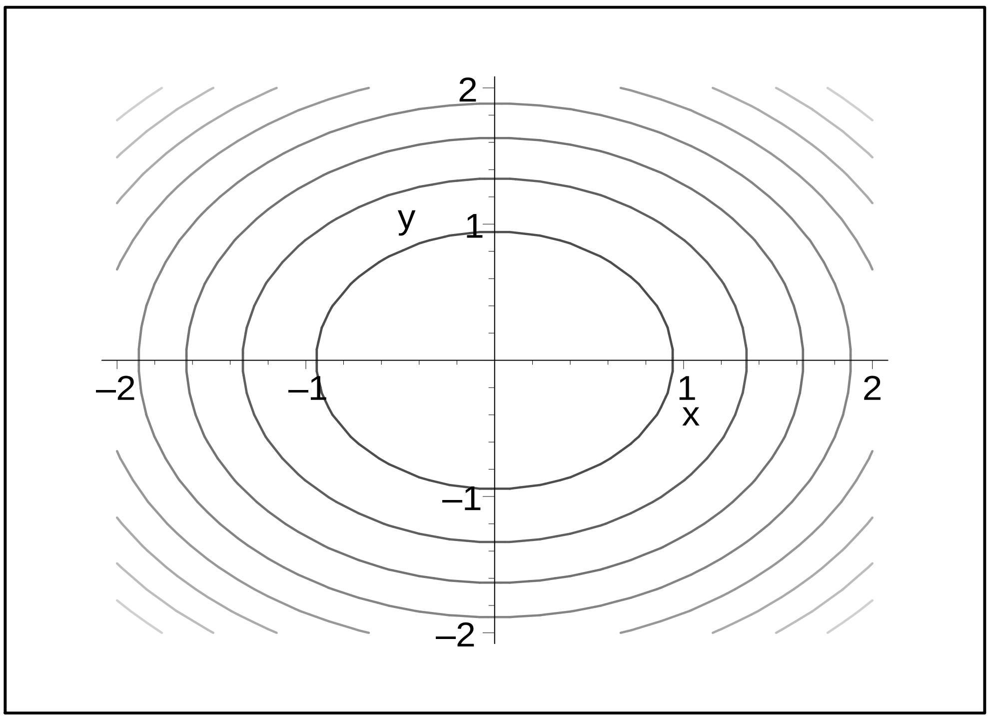
```
> plot3d({x^2+y^2,3},x=-2..2,y=-2..2,axes=boxed);
```



Curvas de nivel, densidad y campos gradientes.

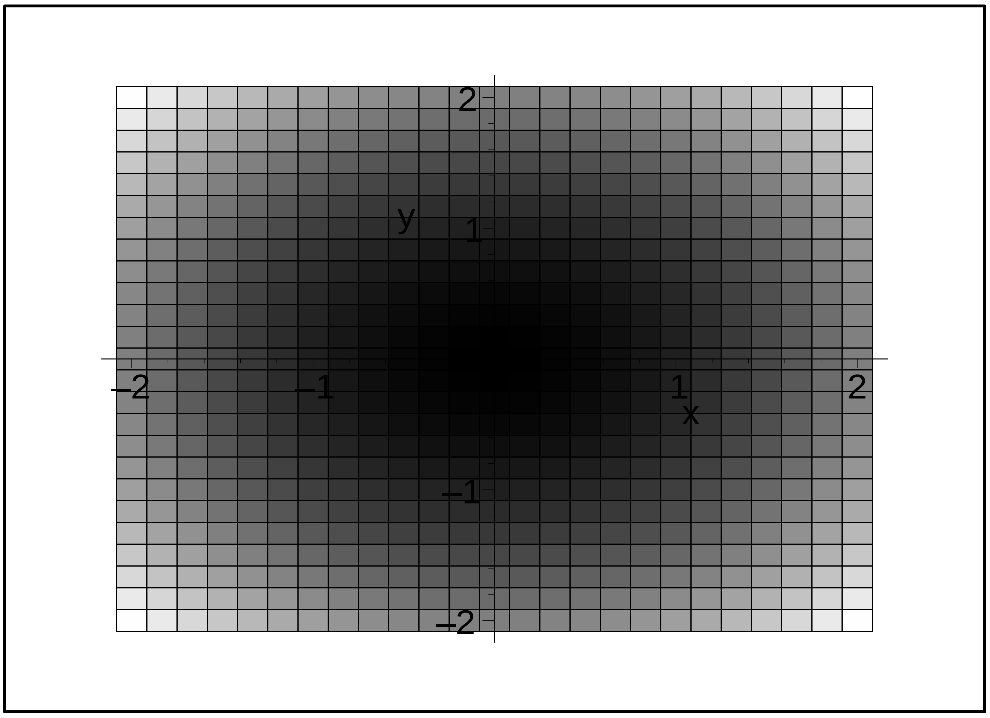
Para pintar las curvas de nivel, conjunto de puntos donde la función es constante, f(x,y)=c, utilizamos el comando **contourplot**

```
>  contourplot(x^2+y^2,x=-2..2,y=-2..2,thickness=3);
```



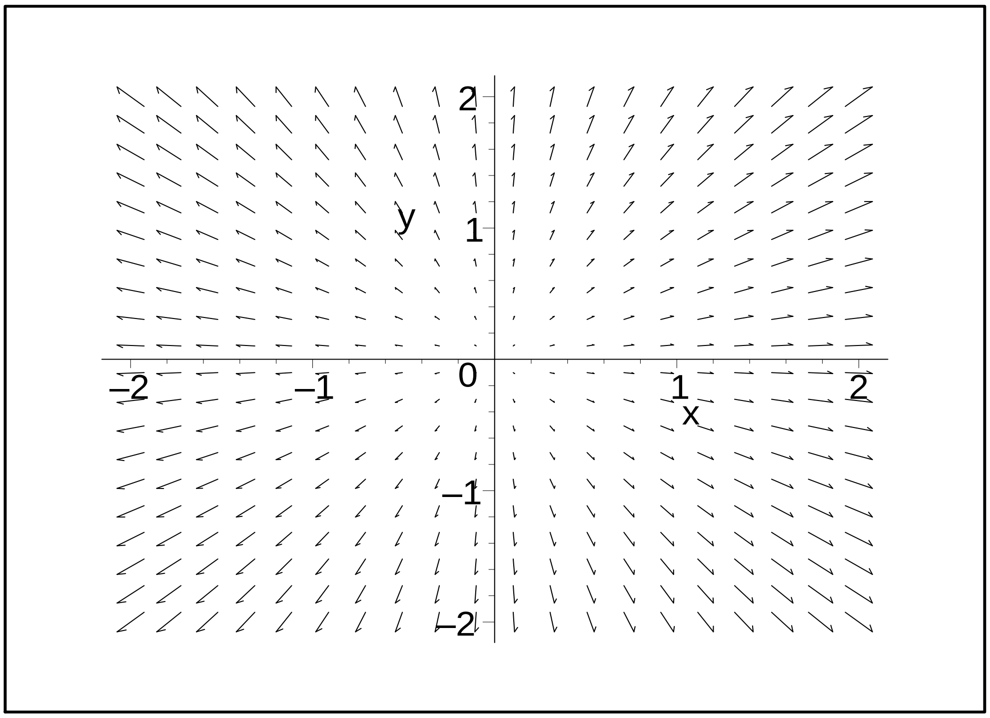
O similarmente utilizando un gráfico de densidad:

```
> densityplot(x^2+y^2,x=-2..2,y=-2..2);
```



Para pintar un campo gradiente utilizamos el comando **gradplot**

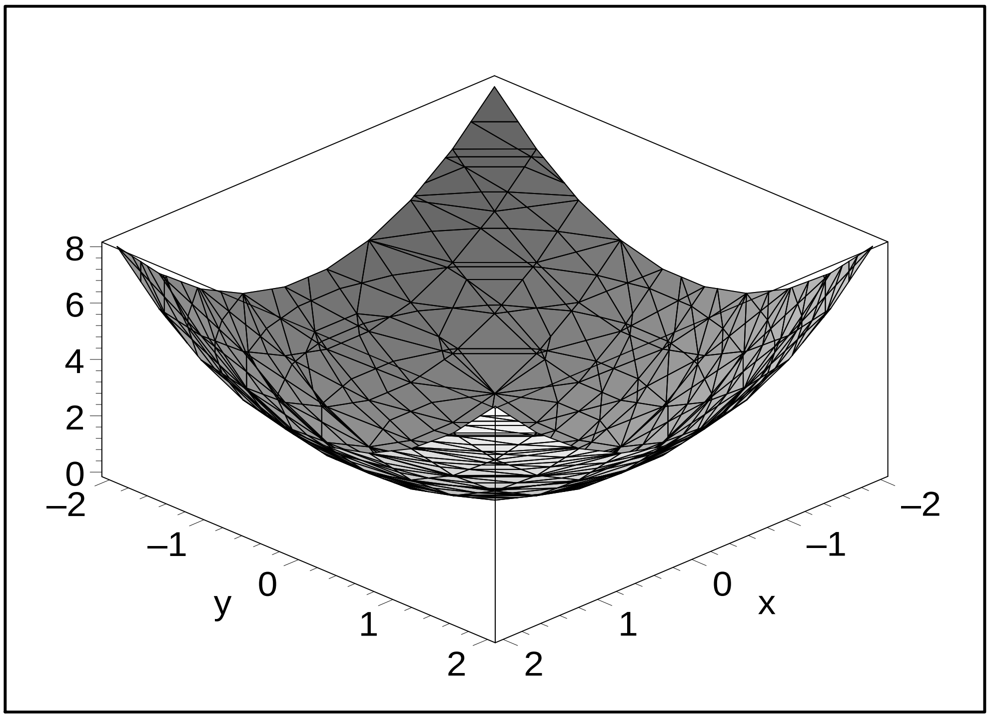
```
> gradplot(x^2+y^2,x=-2..2,y=-2..2);
```



Superficies en forma implícita

Si la superficie viene dada en forma implícita el comando es **implicitplot**

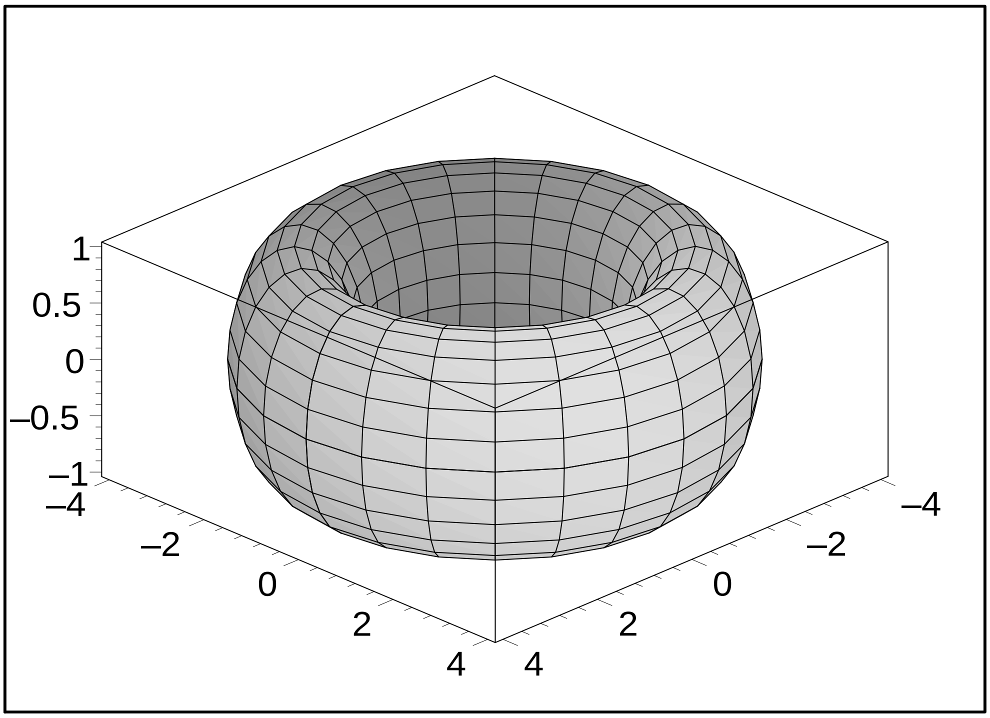
```
>  implicitplot3d(z=x^2+y^2,x=-2..2,y=-2..2,z=0..8,axes=boxed);
```



Superficies en forma paramétrica

De forma similar a las curvas dadas en paramétrica tenemos por ejemplo

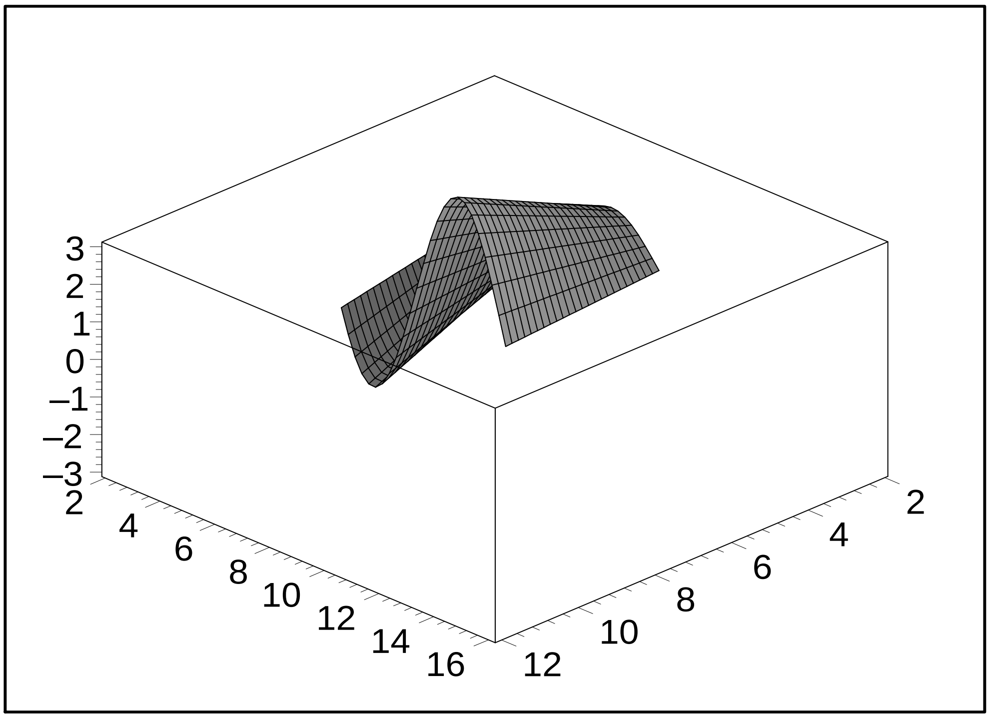
```
> plot3d([(3+cos(r))*cos(t),(3+cos(r))*sin(t),sin(r)],r=0..2*Pi,t=0..2*
> Pi,axes=boxed);
```



Animaciones en 3 dimensiones

También podemos hacer animaciones en 3 dimensiones. El comando es **animate3d. Por ejemplo:**

```
> animate3d([s*t,r*t,s*cos(r*t)],s=1..3,r=1..4,t=2..4,style=patch,axes=
> boxed,frames=30);
```



Ejercicios 8

(a) Pinta la superficie f(x,y)=x^2-y^2, y sus curvas de nivel.

(b) Pinta la superficie z^2+x^2+y^2=1.

(c) Pinta la superficie (exp(t)*cos(u),t,exp(t)*sin(u)) en 0<u<2*Pi, -2<t<2.

Ejercicio 9.

Pinta una esfera en paramétricas en el espacio.

Ejercicio 10.

Realiza una animación en 3d de la superficie [cos(x*t)*sin(y*t)], en el intervalo x=-Pi..Pi, y=-Pi..Pi, en el tiempo t=1..2.

Capítulo 3

Representación gráfica II

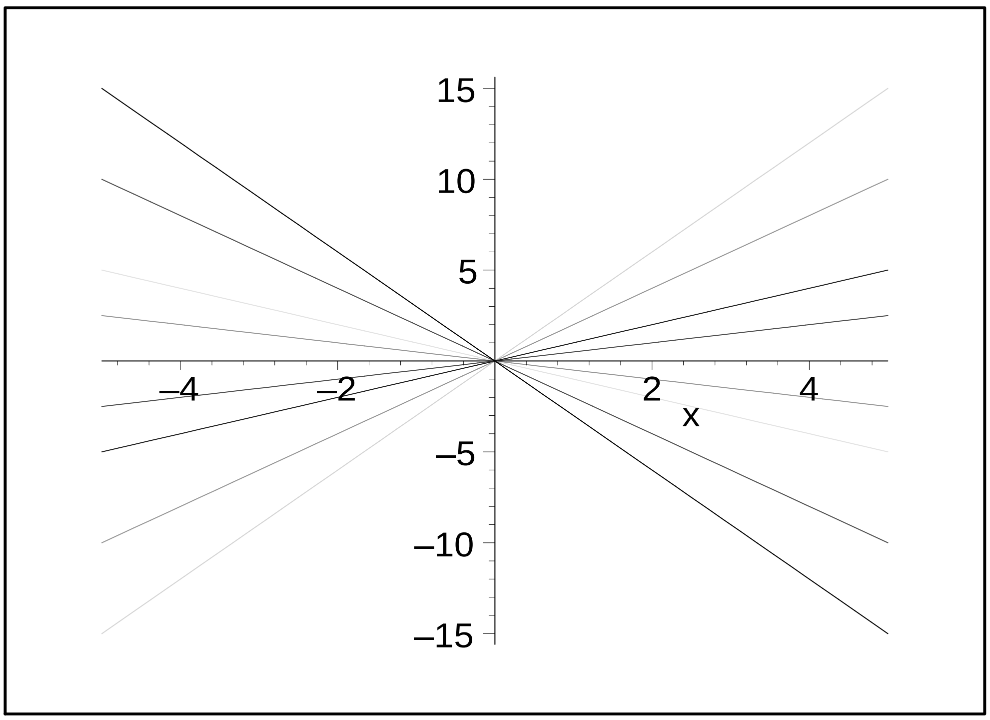
3.1 Familias de funciones en el plano

Muchas veces lo que queremos es estudiar el comportamiento de un conjunto de funciones que dependen de un parámetro (familia uniparamétrica de funciones) o dos (biparamétricas), etc. Pintar distintos valores del parçametro nos puede servir o incluso hacer una animación dependiente de ese parámetro.

Ejemplo 1

Por ejemplo: vamos a estudiar como varía la gráfica de la familia de funciones y=mx+b.Empezamos fijando b, y ariamos m, por ejemplo b=0, y m=1/2, m=1,m=2,m=3, m=-3,m=-2,m=-1,m=-1/2.

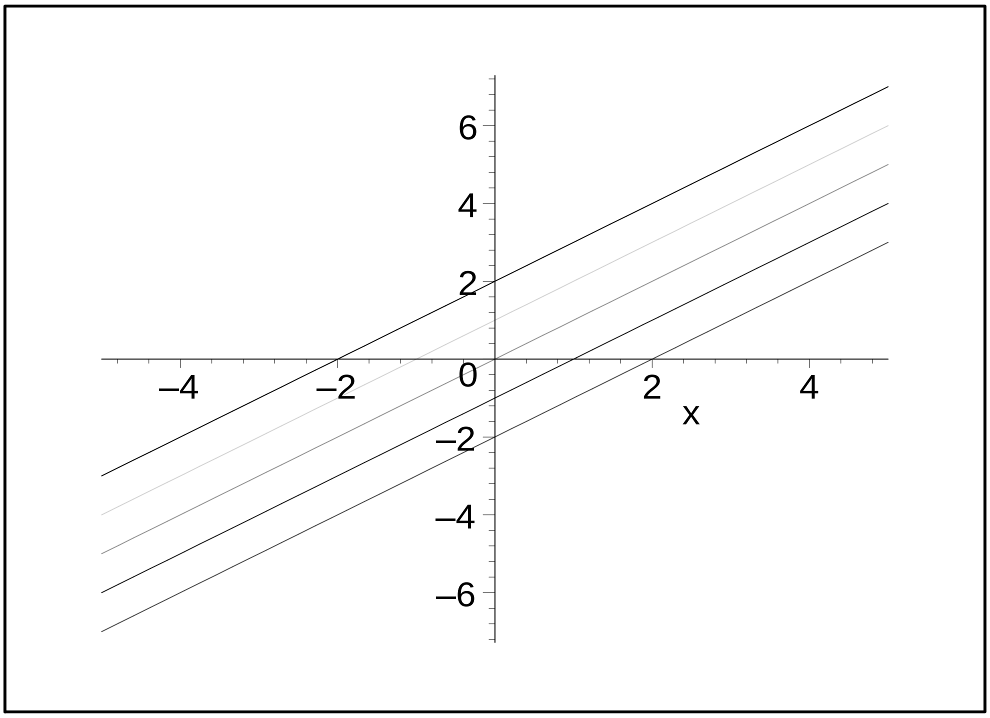
```
> restart:
> plot([1/2*x,x,2*x,3*x,-3*x,-2*x,-x,-1/2*x],x=-5..5,color=[red,blue,gr
> een,pink,black,brown, yellow,khaki ],thickness=2);
```



Observamos que si m (la pendiente) es positiva a medida que es mayor aunmenta la inclinación de la recta formando un ángulo menor de Pi/2, si m es negativa la inclinación de la recta disminuye a medida que dismuye el valor de m formando una ángulo mayor de Pi/2.

Ahora fijamos la m, m=1, por ejemplo y variamos la b=-2, b=-1, b=0, b=1,b=2:

```
> plot([x-2,x-1,x,x+1,x+2],x=-5..5,color=[red,blue,green,pink,black],th
> ickness=2);
```



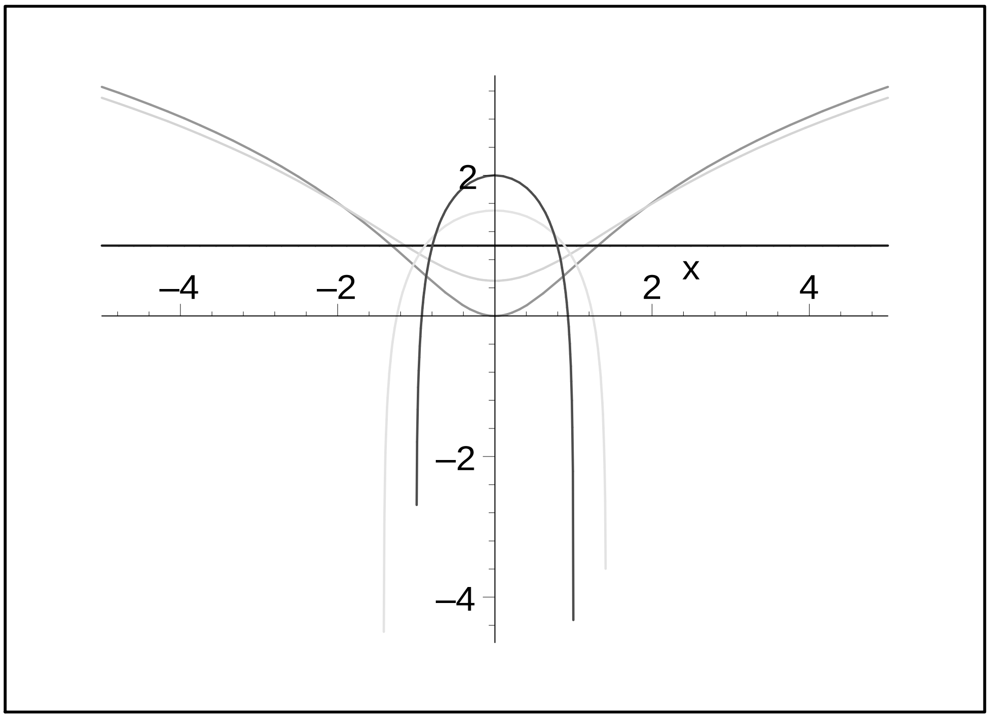
observamos que la recta corta con el eje y en el punto b, de manera que a medida que aumentamos b, el punto de corte va subiendo por el eje.

Ejemplo 2

Representa la familia de funciones y=t+log(1+(1-t)*x^2, con t variando en el intervalo [0,2]. Empezamos pintando algunos valores de t

```
> restart:
```

```
> plot([2+log(1+(1-2)*x^2),1.5+log(1+(1-1.5)*x^2),1+log(1+(1-1)*x^2),0.
> 5+log(1+(1-0.5)*x^2),log(1+x^2)],x=-5..5,color=[red,yellow,blue,pink,g
> reen],thickness=3);
```

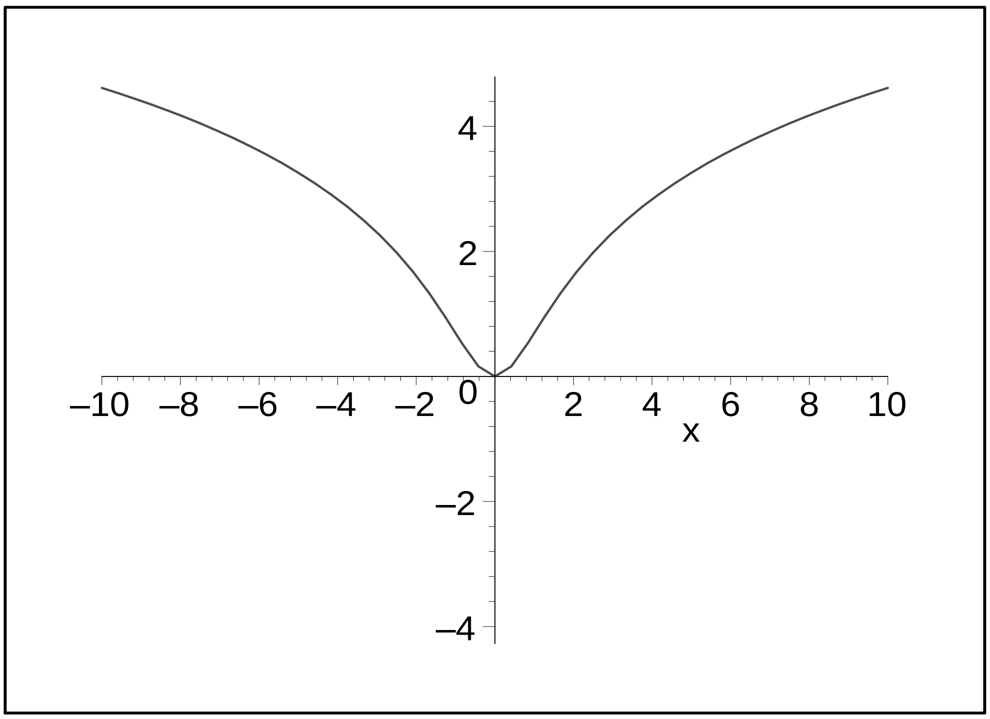


Ahora pintamos todos los valores de t mediante una animación:

```
>  with(plots):
```

```
Warning, the name changecoords has been redefined
```

```
>  animate(
>  t+log(1+(1-t)*x^2),x=-10..10,t=0..2,frames=100,thickness=3);
```



Ejercicios

Estudia las siguientes familias de funciones. Si tienen varios par ámetros tendras que fijar uno y variar el otro como en el ejemplo 1.Hacer una animación cuando creas conveniente.

Ejercicio 9: y=(x-a)^2+b

Ejercicio 10: y=ax^2+bx+c

Ejercicio 11: y=x^n para n pares.

Ejercicio 12: y=x^n para n impares.

Ejercicio 13: y=1/x^a, con a>=0.

Ejercicio 14: y=Asin(Bx)

Ejercicio 15: y=Acos(Bx). Observación: en estas gráficas a —A— se le llama amplitud y a 2Pi/—B— periodo y a —B—/2Pi frecuencia.

Ejercicio 16: y=Asin(Bx-C)

Ejercicio 17: y=Acos(Bx-C)

Capítulo 4

Inicio a la programación con Maple

4.1 Procedimientos

```
> restart:
```

Definimos un procedimiento que calcula el área de un triángulo. (ENTER+ MAYUSCULAS)

```
> area:=proc(b,a)
> (b*a)/2:
> end;
```

$$area := \mathbf{proc}(b,\, a)\, 1/2 * b * a\; \mathbf{end\ proc}$$

```
> area(3,4);
```

$$6$$

Observar en el procedimiento el *encabezamiento, el cuerpo y el terminador*

Dentro del procedimiento podemos definir dos tipos de variables **locales** y **globales**

Vamos a definir un procedimiento que calcule el área de un triángulo y el área de un rectángulo dando la base y la altura

```
> restart:
> area2:=proc(b,a)
> local at,ar:
> at:=(b*a)/2; #area del triángulo de base b y altura a
> ar:=b*a; #area del rectángulo de base b y altura a
> end:
> area2(2,3);
```

$$6$$

```
> at;
```

$$at$$

```
> ar;
```

$$ar$$

```
> restart:
> area2:=proc(b,a)
> global at,ar:
> at:=(b*a)/2; #área del triángulo de base b y altura a
> ar:=b*a; #área del rectángulo de base b y altura a
> end:
> area2(2,3);
```

$$6$$

```
> at;
```

$$3$$

```
> ar;
```

$$6$$

De esta manera tenemos definidas las variables fuera del procedimiento. Cuidado con la memoria. Observar que una linea precedida de**#** no se ejecuta, sirve para hacer un comentario dentro del procedimiento.

Otra manera de que se muestren varibles locales fuera del procedimiento es utilizando el comando **print**

```
> restart:
> area3:=proc(b,a)
> local t,r:
> t:=(b*a)/2; #area del triángulo de base b y altura a
> print(`área del triángulo`,t);
> r:=b*a; #area del rectángulo de base b y altura a
> print(`área del rectángulo`,r);
> end:
> area3(3,4);
```

$$rea\ del\ tringulo,\ 6$$

$$rea\ del\ rectngulo,\ 12$$

EJERCICIO 1. Escribir un procedimiento que dado el radio calcule el área del círculo y la longitud de la circunferencia.

4.2 Subprocedimientos

```
> restart:
```

Podemos utilizar un procedimiento dentro de otro.

```
> semisuma:=proc(s1,s2)
> (s1+s2)/2;
> end:
> semisuma(3,7);
```

$$5$$

```
> trapecio:=proc(b1,b2,a)
> semisuma(b1,b2)*a;
> end:
> trapecio(2,3,4);
```

$$10$$

EJERCICIO 2. Utilizar el procedimiento que habeis construido antes para calcular el área de un círculo como subprocedimiento de otro que calcule el volumen de un cilindro.

4.3 Condicionales

La estructura de un condicional:

if *condición* ***then*** *lo que queremos que pase* **fi**

```
> temperatura:=proc(t)
> if t>37.5 then print(`tienes fiebre`) fi
> end:
> temperatura(40);
```

tienes fiebre

```
> temperatura(36);
```

Observar que si no se cumple la condición el ordenador no contesta nada.

Observemos que para que imprima una frase, esta ha de ir entre acentos graves.

Pero podemos poner más condiciones

if *condición* ***then*** *lo que queremos que pase* **else** *en otro caso lo que pase* ***fi***

```
> temp2:=proc(t)
> if t>37.5
> then print(`tienes fiebre`);
> else print(`estas bien`);
> fi
> end:
> temp2(38);
```

tienes fiebre

```
> temp2(36);
```

estas bien

O más condiciones todavia

if *condición* ***then*** *lo que queremos que pase*

elif *condición* ***then*** *lo que queremos que pase*

elif *condición* ***then*** *lo que queremos que pase*

.

.

.

else *en otro caso lo que queremos que pase*

fi

También podemos pedir que las variables sean de un tipo específico.

```
> billete_tren:=proc(edad::integer)
> if edad<=3 then print(`los bebes no pagan`);
> elif edad>3 and edad<=12 then print(`los niños pagan la mitad`);
> elif edad>12 and edad<=65 then print(`son 3 euros`);
> else print(`tiene un descuento del 75%`):
> fi
> end:
> billete_tren(1);
```

los bebes no pagan

```
> billete_tren(5);
```

$$los\ ni\ os\ pagan\ la\ mitad$$

```
> billete_tren(18);
```

$$son\ 3\ euros$$

```
> billete_tren(70);
```

$$tiene\ un\ descuento\ del\ 75\%$$

Observar como hemos utilizado **and,** de manera similar a cuando definiamos intervalos.

Conectivos similares a **and** son **or** o **not.**

EJERCICIO 3. Escribir un procedimiento que decida según la velocidad a la que nades en que nivel de natación estas. Supon al menos 4 niveles distintos.

4.4 Procedimientos Recursivos

A este tipo de procedimientos les llamamos recursivos, porque son procedimientos que se envian así mismos. Atención que en este tipo de programas si no esta bien pensado el esquema lógico o los datos que se introducen no son adecuados es fácil caer en bucles infinitos, en errores.

```
> restart:
> fijo:=proc(f,x,k)
> local y:
> y:=f(x);
> if abs(f(x)-x)<k then print(x,k);
> else  print(y); fijo(f,y,k); fi
> end:
> f:=x->x^2:
> fijo(f,0.5,0.01);
```

$$.25$$
$$.0625$$
$$.00390625$$
$$.00390625,\ .01$$

EJERCICIO 4. Escribir un procedimiento recursivo que escriba los números naturales consecutivos desde un n dado hasta otro cierto m

4.5 Bucles

El esquema más sencillo de un bucle es

for ***variable*** **to** ***variable do*** *lo que queramos que haga* ***od***

```
> restart:
> for i to 5 do 3*i od;
```

$$3$$
$$6$$
$$9$$
$$12$$
$$15$$

Por defecto la variable empieza en 1

O dentro de un procedimiento:

```
> mult1:=proc(n)
> local i:
> for i to n do print(3*i) od:
> end:
> mult1(5);
```

$$3$$
$$6$$
$$9$$
$$12$$
$$15$$

Si en vez de empezar en 1 queremos empezar en otro numero entero, m, no tenemos más que añadir **from,** el esquema ahora será

for *variable from* *variable* ***to variable do*** *lo que queramos que haga* ***od***

```
> mult2:=proc(m,n)
> local i:
> for i from m to n do print(3*i) od:
> end:
> mult2(7,10);
```

$$21$$
$$24$$
$$27$$
$$30$$

Si además en vez de ir de 1 en 1 queremos ir con otro paso intercalamos **by**

for *variable from* *variable* ***by variable to variable do*** *lo que queramos que haga* ***od***

```
> mult3:=proc(m,p,n)
> local i:
> for i from m by p to n do print(3*i) od:
> end:
> mult3(5,2,11);
```

$$15$$
$$21$$
$$27$$
$$33$$

for *i while condición sobre i do* *lo que queramos que pase* ***od***

```
> mult4:=proc(n)
> local i:
> for i while i^2<n do print(3*i) od
> end:
> mult4(10);
```

$$3$$
$$6$$
$$9$$

Ejercicio 5: Escribe un procedimiento que calcule las potencias de 3, desde la potencia inicial que queramos a la final que queramos, y que vaya con el paso que definamos.

Ejemplo sobre la asignación x:=x+k

```
> x:=3:
> x:=x+3;
```

$$x := 6$$

Aqui vemos un procedimiento que suma los n primeros números naturales y presenta cada suma paracial.

```
> restart:
> suma:=proc(n)
> local i,s:
> s:=0:
> for i to n do s:=s+i; print(s); od;
> end:
> suma(5);
```

$$1$$
$$3$$
$$6$$
$$10$$
$$15$$

Ejercicio 6: Escribe un procedimiento que sume los 2n primeros números pares.

Bucles y condicionales

Vamos ahora a mezclar un bucle y un condicional, observar el ejemplo siguiente que nos da los terminos que le pidamos de una sucesión aritmética.

Ejemplo. Definimos un procedimiento que calcula y muestra los n primeros términos de una sucesión aritmética de diferencia d y término inicial a0

```
> restart:

> arit:=proc(a0,d,n)
> local j,an;
> print(a0);
> if n=1 then an:=a0;
> else for j to n-1 do an:=d*j+a0; print(an); od; fi;
> end:
> arit(1,3,5);
```

$$1$$
$$4$$

7
10
13

Ejercicio 7. Definir un procedimiento que calcula y muestra los n primeros términos de una sucesión geométrica de razón r y término inicial a0.

Problema

Utilizando, de todo lo anterior, lo que creas conveniente construye un procedimiento (un programa) que tenga como variables, f, l y e, donde f sea una sucesión de la que queremos calcular el límite, l el límite posible que creamos y e el error máximo que queramos cometer y que la respuesta del procedimiento consista en a partir de qué término de la sucesión todos los siguientes estan más cerca del límite que el error dado.

Construir otro procedimiento que os ayude a suponer el límite de una sucesión cualquiera.

Aplicar estos procedimientos a las sucesiones de los problemas 25, 26 y 27 de la hoja 3 de problemas.

Capítulo 5

Modelos

5.1 Ejemplo

Un campesino utilizó cuatro parcelas experimentales para determinar la relación entre la producción de trigo en kilogramos por área y la cantidad de fertilizante en litros por área. Los resultados se muestran en la tabla siguiente.

Fertilizante Producción

1.0 32

1.5 41

2.0 48

2.5 53

Vamos a modelizar esta situación de diferentes maneras para poder calcular de manera aproximada la producción para una aplicación de 1.6 litros por area y de 3 litros por area.

Empezamos con la recta de regresión

Construimos las matrices X eY, como vamos a manipular matricfes necesitamos cargar el paquete de algebra lineal **linalg**

```
>  with(linalg):
```

```
Warning, the protected names norm and trace have been redefined and
unprotected
```

```
>  X:=matrix(4,2,[1,1.0,1,1.5,1,2.0,1,2.5]);
```

$$X := \begin{bmatrix} 1 & 1.0 \\ 1 & 1.5 \\ 1 & 2.0 \\ 1 & 2.5 \end{bmatrix}$$

```
>  Y:=matrix(4,1,[32,41,48,53]);
```

$$Y := \begin{bmatrix} 32 \\ 41 \\ 48 \\ 53 \end{bmatrix}$$

Calculamos la matrix de los coeficientes:

A=(X^tX)^(-1)X^tY

```
> XT:=transpose(X);
```

$$XT := \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1.0 & 1.5 & 2.0 & 2.5 \end{bmatrix}$$

```
> P:=multiply(XT,X);
```

$$P := \begin{bmatrix} 4 & 7.0 \\ 7.0 & 13.50 \end{bmatrix}$$

```
> P1:=inverse(P);
```

$$P1 := \begin{bmatrix} 2.700000000 & -1.400000000 \\ -1.400000000 & .8000000000 \end{bmatrix}$$

```
> A:=multiply(P1,XT,Y);
```

$$A := \begin{bmatrix} 19.00000000 \\ 14.00000000 \end{bmatrix}$$

De manera que la recta de regresión que buscamos es

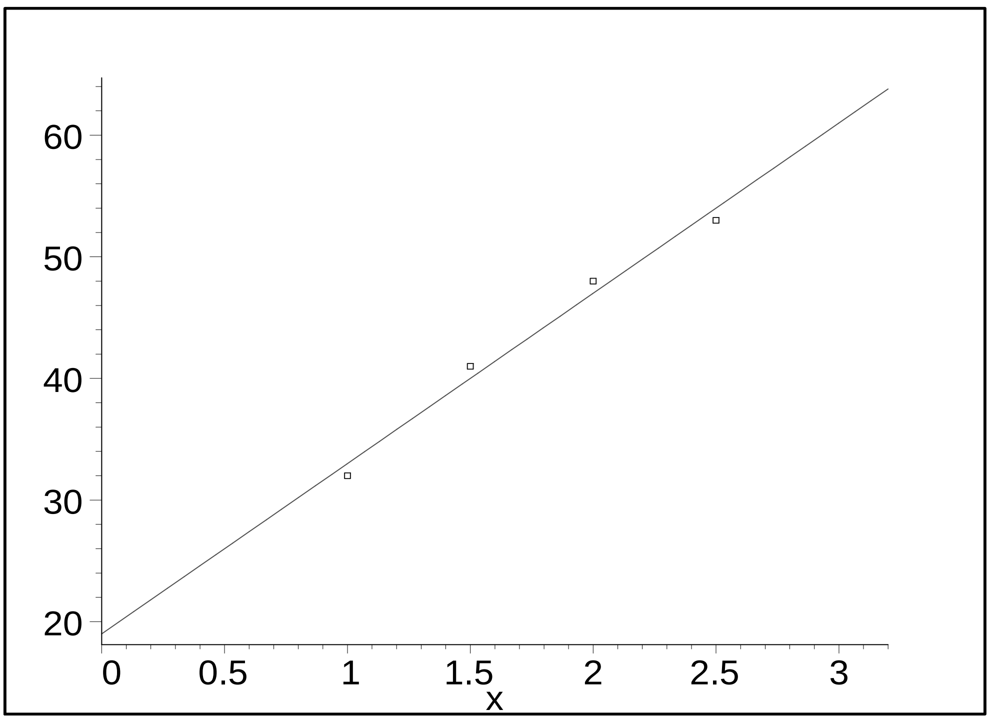
y=19+14x

vamos a pintar los puntos y la recta:

```
> with(plots):
```

```
Warning, the name changecoords has been redefined
```

```
> d1:=pointplot({[1.,32],[1.5,41],[2.0,48],[2.5,53]
> },symbol=box,color=blue):
> d2:=plot(19+14*x,x=0..3.2,thickness=2):
> display(d1,d2,thickness=3);
```



Si queremos saber cuanta producción habria con 1.6 de fertilizante y con 3, lo que hacemos es sustituirlo en el valor de la recta

```
> r1:=x->19+14*x:
```

```
> r1(1.6);
```

$$41.4$$

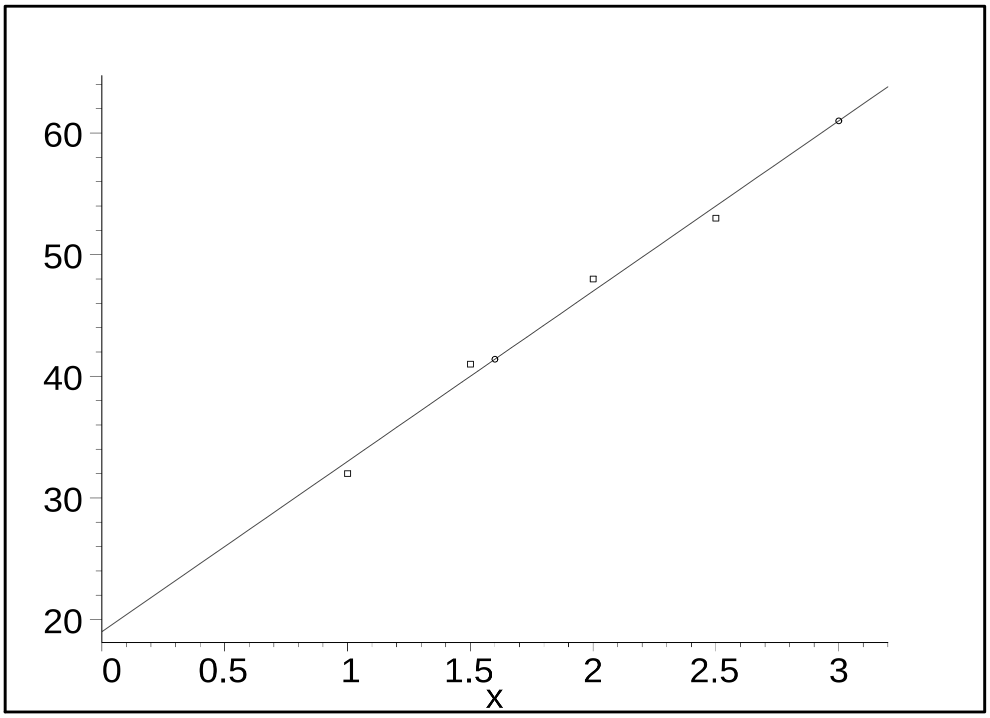
```
> r1(3.);
```

$$61.$$

Pintamos estos valores también:

```
> d3:=pointplot({[1.6,41.4],[3,61]},symbol=circle,color=black):
```

```
> display(d1,d2,d3);
```



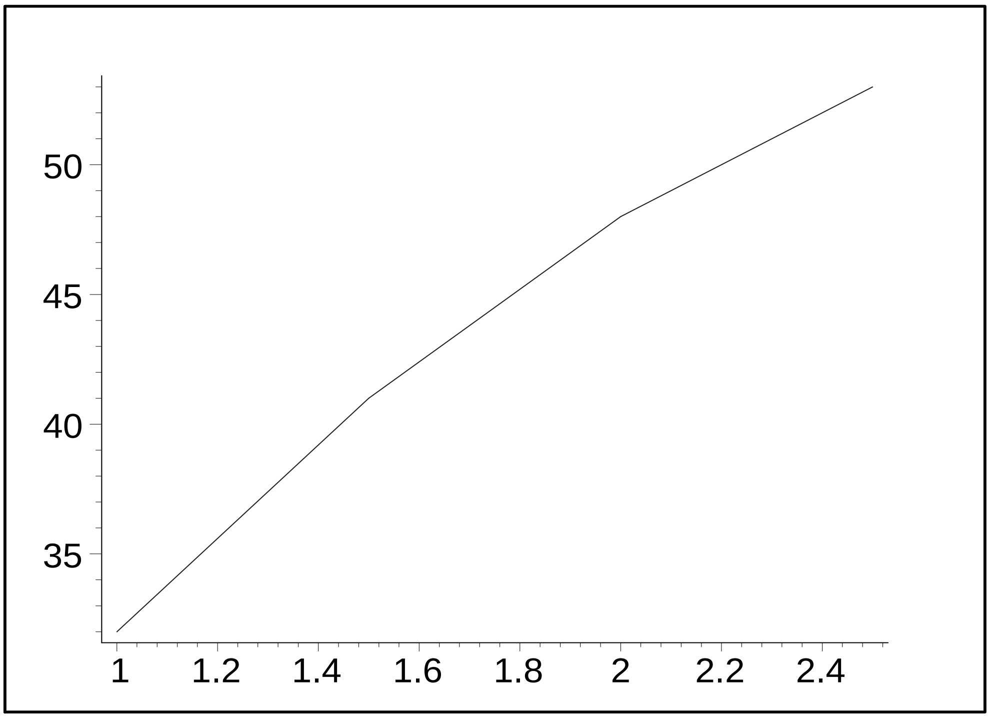
Resolvemos ahora mediante interpolación lineal

```
> restart:
```

```
> with(plots):
```

```
Warning, the name changecoords has been redefined
```

```
> plot({[1.,32],[1.5,41],[2.0,48],[2.5,53]},symbol=box,color=blue);
```



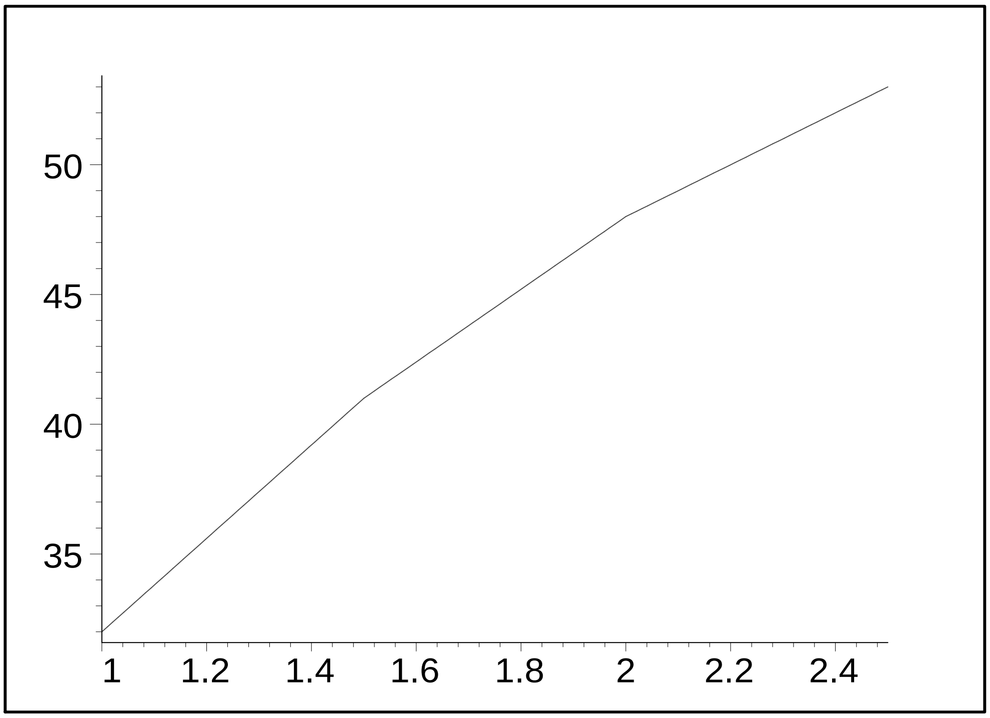
Definimos la función con rectas uniendo los puntos:

```
> f1:=x->piecewise(1.<=x and x<1.5,14+18*x,1.5<=x and
> x<2.0,20+14*x,2<=x ,28+10*x);
```

$$f1 := x \to \mathrm{piecewise}(1. \leq x \textbf{ and } x < 1.5,\ 14 + 18\,x,\ 1.5 \leq x \textbf{ and } x < 2.0,\ 20 + 14\,x,\ 2 \leq x,\ 28 + 10\,x)$$

Comprobamos que la hemos definido bien:

```
> plot(f1,1..2.5,discont=true);
```



Calculamos ahora los valores en 1.6 y 3

```
> f1(1.6);
```

$$42.4$$

```
> f1(3.);
```

$$58.$$

Resolvemos ahora mediante los polinomios de Lagrange

Como tenemos cuatro puntos, buscaremos un polinomio de tercer grado:

```
> restart:
> p:=x->a+b*x+c*x^2+d*x^3;
```

$$p := x \to a + b\,x + c\,x^2 + d\,x^3$$

Planteamos las ecuaciones: p(1.0)=32, p(1.5)=41,p(2.)=48, p(2.5)=53

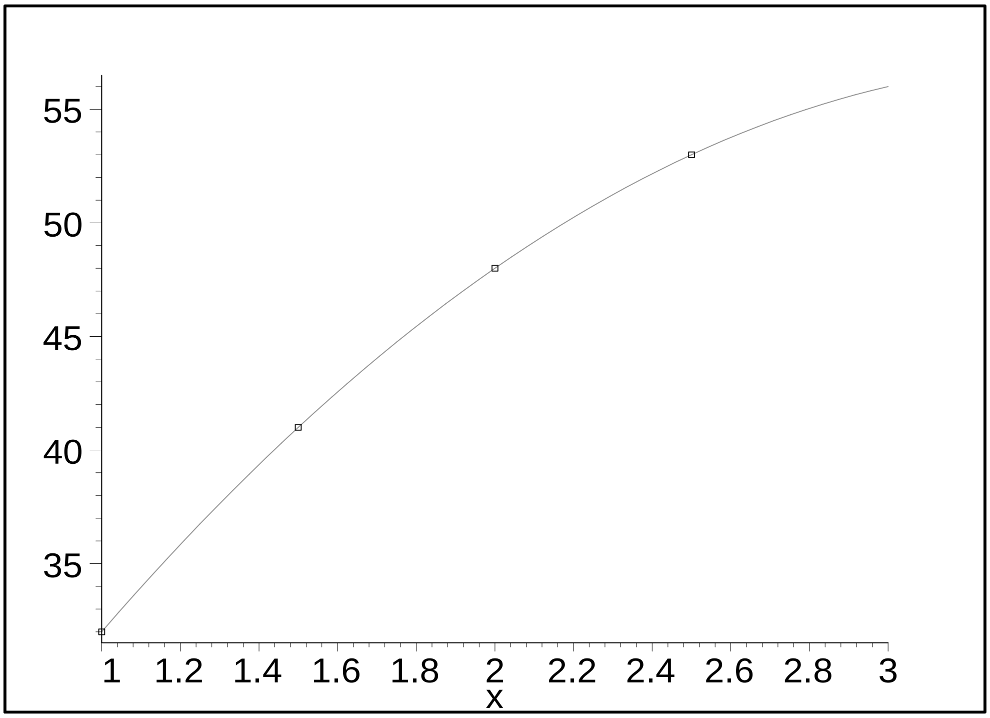
```
> solve({p(1.0)=32, p(1.5)=41,p(2.)=48, p(2.5)=53},{a,b,c,d});
```

$$\{c = -4., a = 8., d = 0., b = 28.\}$$

De manera que el polinomio que buscamos es

```
> p1:=8+28*x-4*x^2:
> with(plots):
Warning, the name changecoords has been redefined
```

```
> d1:=pointplot({[1.,32],[1.5,41],[2.0,48],[2.5,53]
> },symbol=box,color=blue):
> d4:=plot(p1,x=1..3,thickness=2,color=green):
> display(d1,d4);
```




```
> subs(x=1.6,p1);
```

$$42.56$$

```
> subs(x=3,p1);
```

$$56$$

Resolvemos ahora mediante splines cúbicas

Vamos a tener 3 polinomios de tercer grado ya que tenemos tres subintervalos

pi=ai+bix+cix^2+dix^3 definido en [xi,xi+1],

```
> restart:
> with(plots):
Warning, the name changecoords has been redefined
> dibu1:=pointplot({[1.,32],[1.5,41],[2.0,48],[2.5,53]
> },symbol=box,color=blue):
> p1:=x->a1+b1*x+c1*x^2+d1*x^3;
```

$$p1 := x \to a1 + b1\,x + c1\,x^2 + d1\,x^3$$

```
> p2:=x->a2+b2*x+c2*x^2+d2*x^3;
```

$$p2 := x \to a2 + b2\,x + c2\,x^2 + d2\,x^3$$

```
> p3:=x->a3+b3*x+c3*x^2+d3*x^3;
```

$$p3 := x \to a3 + b3\,x + c3\,x^2 + d3\,x^3$$

Imponemos las condiciones sobre los polinomios: pi(xi)=yi, i=0..n-1, sobre las derivadas pi'(xi+1)=pi+1'(xi+1) , i=0..n-2 y sobre la derivada segunda pi''(xi+1)=pi+1''(xi+1). Y ademas imponemos las condiciones frontera para completar el sistema: p1''(x0)=0=pn''(x0)

```
> ecu1:=p1(1.)=32;
```

$$ecu1 := a1 + 1.\,b1 + 1.\,c1 + 1.\,d1 = 32$$

```
> ecu2:=p1(1.5)=41;
```

$$ecu2 := a1 + 1.5\,b1 + 2.25\,c1 + 3.375\,d1 = 41$$

```
> ecu3:=p2(1.5)=41;
```

$$ecu3 := a2 + 1.5\,b2 + 2.25\,c2 + 3.375\,d2 = 41$$

```
> ecu4:=p2(2.)=48;
```

$$ecu4 := a2 + 2.\,b2 + 4.\,c2 + 8.\,d2 = 48$$

```
> ecu5:=p3(2.)=48;
```

$$ecu5 := a3 + 2.\,b3 + 4.\,c3 + 8.\,d3 = 48$$

```
> ecu6:=p3(2.5)=53;
```

$$ecu6 := a3 + 2.5\,b3 + 6.25\,c3 + 15.625\,d3 = 53$$

```
> ecu7:=D(p1)(1.5)=D(p2)(1.5);
```

$$ecu7 := b1 + 3.0\,c1 + 6.75\,d1 = b2 + 3.0\,c2 + 6.75\,d2$$

```
> ecu8:=D(p2)(2.)=D(p3)(2.);
```

$$ecu8 := b2 + 4.\,c2 + 12.\,d2 = b3 + 4.\,c3 + 12.\,d3$$

```
> ecu9:=D(D(p1))(1.5)=D(D(p2))(1.5);
```

$$ecu9 := 2\,c1 + 9.0\,d1 = 2\,c2 + 9.0\,d2$$

```
> ecu10:=D(D(p2))(2)=D(D(p3))(2.);
```

$$ecu10 := 2\,c2 + 12\,d2 = 2\,c3 + 12.\,d3$$

```
> ecu11:=D(D(p1))(1.)=0;
```

$$ecu11 := 2\,c1 + 6.\,d1 = 0$$

```
> ecu12:=D(D(p3))(2.5)=0;
```

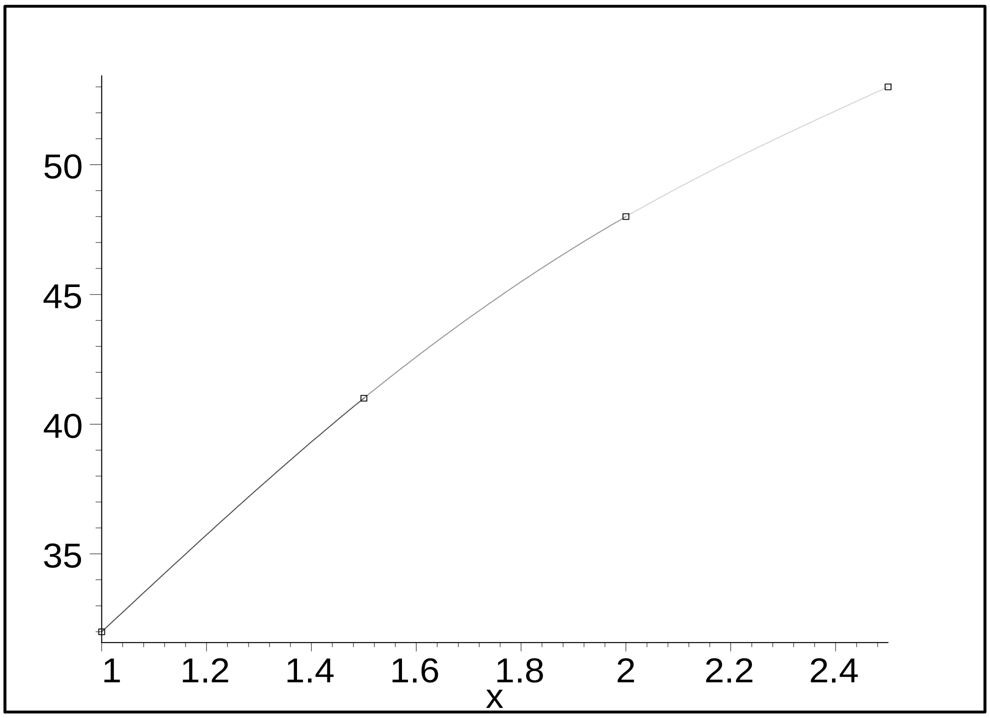
$$ecu12 := 2\,c3 + 15.0\,d3 = 0$$

```
> solve(
> {ecu1,ecu2,ecu3,ecu4,ecu5,ecu6,ecu7,ecu8,ecu9,ecu10,ecu11,ecu12},
> {a1,a2,a3,b1,b2,b3,c1,c2,c3,d1,d2,d3});
```

$$\{a3 = -20.,\ d1 = -3.200000000,\ c2 = -4.800000000,\ c1 = 9.600000000,\ a2 = 5.600000000,\ b1 = 9.200000000,\ a1 = 16.40000000,\ d3 = 3.200000000,\ b2 = 30.80000000,\ b3 = 69.20000000,\ c3 = -24.,\ d2 = 0.\}$$

Ahora lo pintamos

```
> p1:=16.40000000+9.200000000*x+9.600000000*x^2-3.200000000*x^3:
> p2:=5.600000000+30.80000000*x-4.800000000*x^2+0*x^3:
> p3:=-20.+69.20000000*x-24.*x^2+3.200000000*x^3:
> dibu2:=plot(p1,x=1..1.5,color=red):
> dibu3:=plot(p2,x=1.5..2,color=green):
> dibu4:=plot(p3,x=2..2.5,color=pink):
> display(dibu1,dibu2,dibu3,dibu4);
```




```
>   subs(x=1.6,p2);
```

$$42.59200000$$

```
>   subs(x=3.,p3);
```

$$58.00000000$$

Resolver cada uno de los siguientes problemas por los cuatro tipos de modelos como en el ejemplo. Comparar los distintos resultados. Intentar utilizar siempre el camino más corto.Intentar programar las partes más rutinarios.

5.2 Problema 1

El índice de precios al consumidor de todos los productos durante cinco años se da en la tabla siguiente

Año IPC

1980 247.0

1981 271.3

1982 288.6

1983 297.4

1984 310.7

Calcula el índice de precios al consumidor para el año 1990.

5.3 Problema 2

Encuentre y dibuje un modelo que relacione el periodo de los tres primeros planetas con su distancia media al Sol, como se muestra en la tabla más abajo. Luego probar la exactitud del modelo para calcular el periodo de Saturno. (La distancia se mide en unidades astronomicas y el periodo en años). Repita este problema con los cinco primeros y probar la exactitud del modelo para Saturno. Compara los dos modelos.

Planeta Distancia Periodo

Mercurio 0.387 0.241

Venus 0.723 0.615

Tierra 1.0 1.0

Marte 1.523 1.881

Jupiter 5.203 11.861

Saturno 9.541 29.457

5.4 Problema 3

En la tabla mostrada a continuación se observa la población de un pais en los años 1920, 1930, 1940 y 1950

Año Población (en millones)

1920 106

1930 123

1940 132

1950 151

Calcula la población de 1960. ¿Cómo se compara tu estimación con la población real de 179 millones en 1960?

5.5 Problema 4

Según la agencia de estadística la población mundial en millones para los años 1950 y 1985 es como se observa en la siguiente tabla

Año Población

1950 2530

1955 2770

1960 3050

1965 3360

1970 3720

1975 4100

1980 4470

1985 4870

Calcula la población de 1990 y 1995.

Capítulo 6

Resolución de ecuaciones f(x)=0

6.1 Algo sobre errores.

Error absoluto: Llamamos error absoluto a la diferencia, en valor absoluto, entre el valor real de un cierto resultado y la estimación que hagamos. Ej: valor real 1.3548, estimación 1.3532, error absoluto=—1.3548-1.3532—=0.0016.

Errores de redondeo y truncamiento. Llamamos *error de redondeo al que estamos comentiendo cuando efectuamos la diferencia, en valor absoluto, entre un número y su aproximación* por otro de manera que si la última cifra es mayor o igual que cinco aumentamos la anterior en una unidad y si es menor que cinco la disminuimos: (a esto se le lllama redondeo hacia arriba, si consideramos el cinco para disminuir, tendremos redondeo hacia abajo). Ej: 7.2374 lo podemos redondear con tres cifras decimales a 7.237 y con dos cifras decimales a 7.24. comentiendo en el primer caso un error de 0.0004 y en el segundo caso un error de 0.0026.

Llamamos *error de truncamiento* a la diferencia, en valor absoluto, entre una cifra y su aproximación construida mediante la eliminación de la parte final de sus dígitos: Por ejemplo: 12.36785 aproximamos por 12.36 y estamos cometiendo un error de 0.00785.

Errores de resta caótica: Es el que comete el ordenador debido a que sólo puede guardar, en este caso, 10 cifras significativas, observar el siguiente ejemplo:

```
> restart:
```

```
> (0.123456789-0.123456788)*1e9;
```

$$1.$$

```
> (0.1234567891-0.1234567890)*1e10;
```

$$1.$$

```
> (0.12345678912-0.12345678911)*1e11;
```

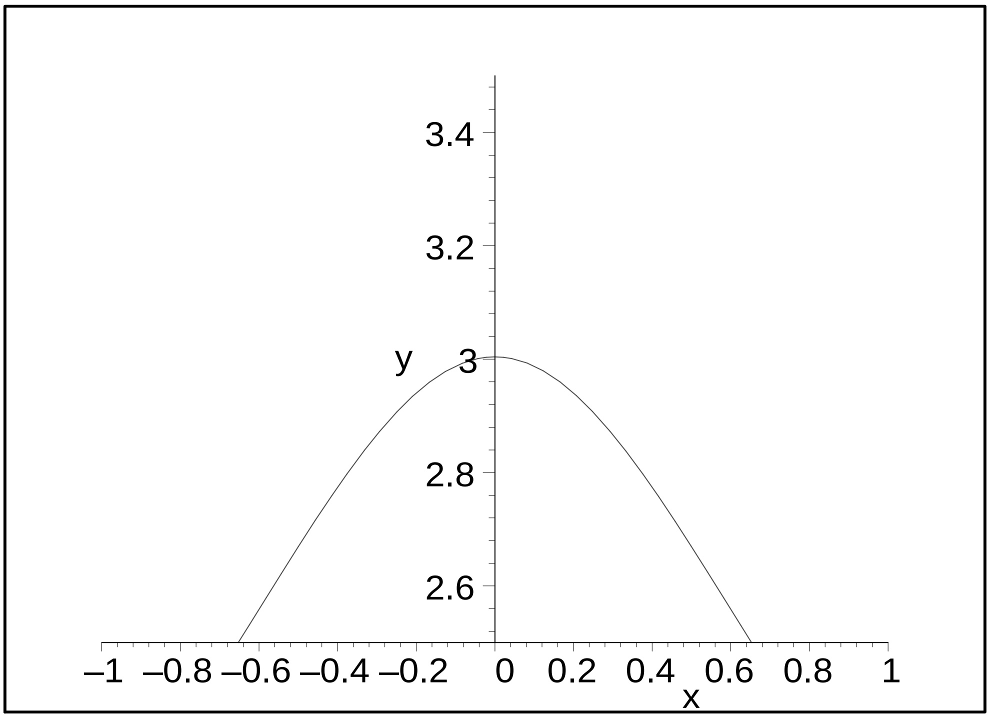
$$0.$$

Y esto puede modificar un resultado gráfico:

```
> f(x):=(1+x^2)^(1.1/x^2);
```

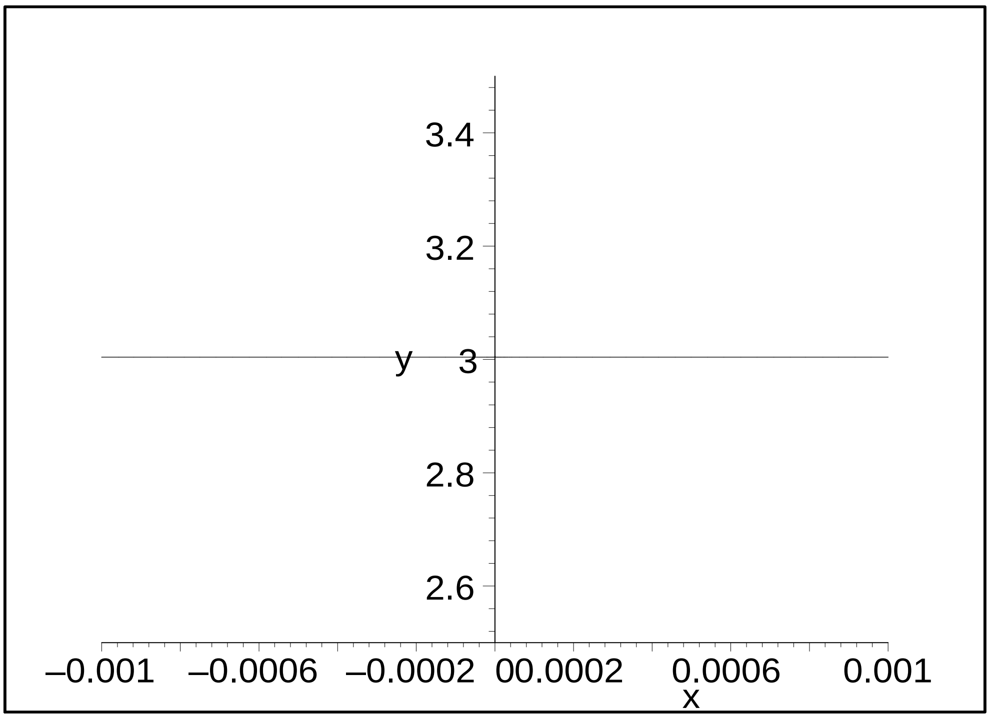
$$\mathrm{f}(x) := (1+x^2)^{\left(\frac{1.1}{x^2}\right)}$$

```
> plot(f(x),x=-1..1,y=2.5..3.5);
```

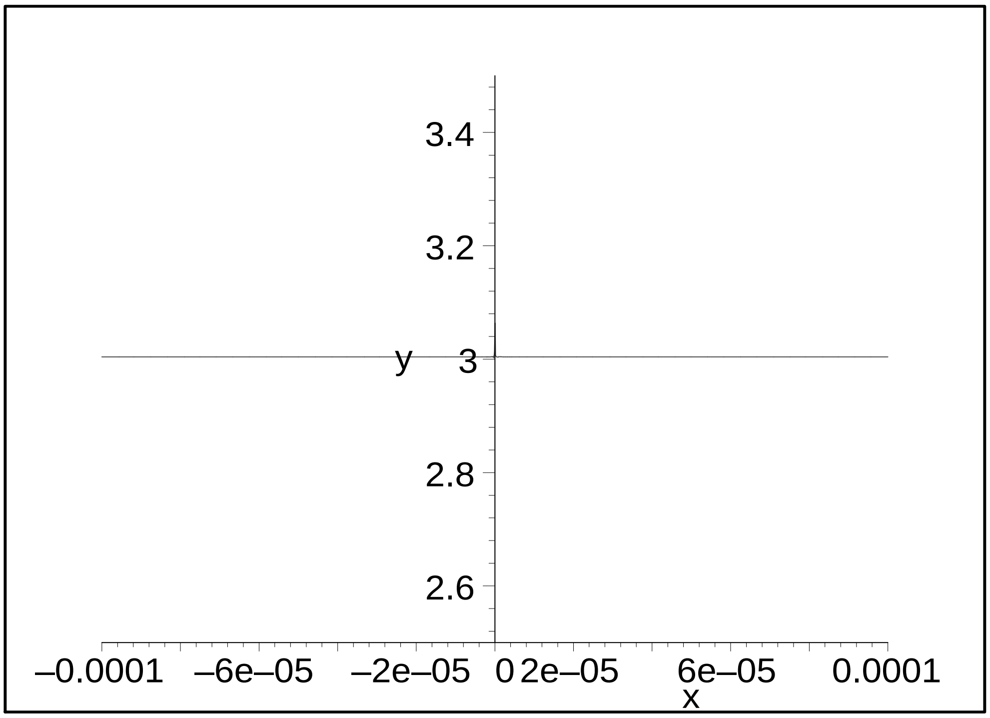


¿Cuál crees que es el limite de f cuando x ->0?

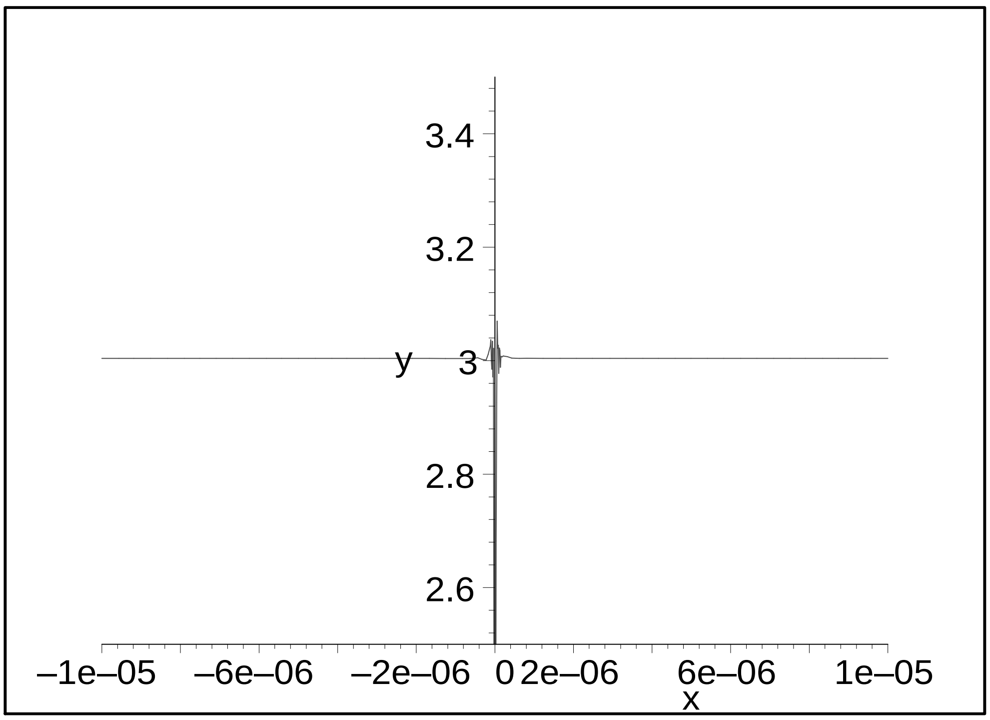
```
>  plot(f(x),x=-0.001..0.001,y=2.5..3.5);
```




```
>  plot(f(x),x=-0.0001..0.0001,y=2.5..3.5);
```




```
> plot(f(x),x=-0.00001..0.00001,y=2.5..3.5);
```



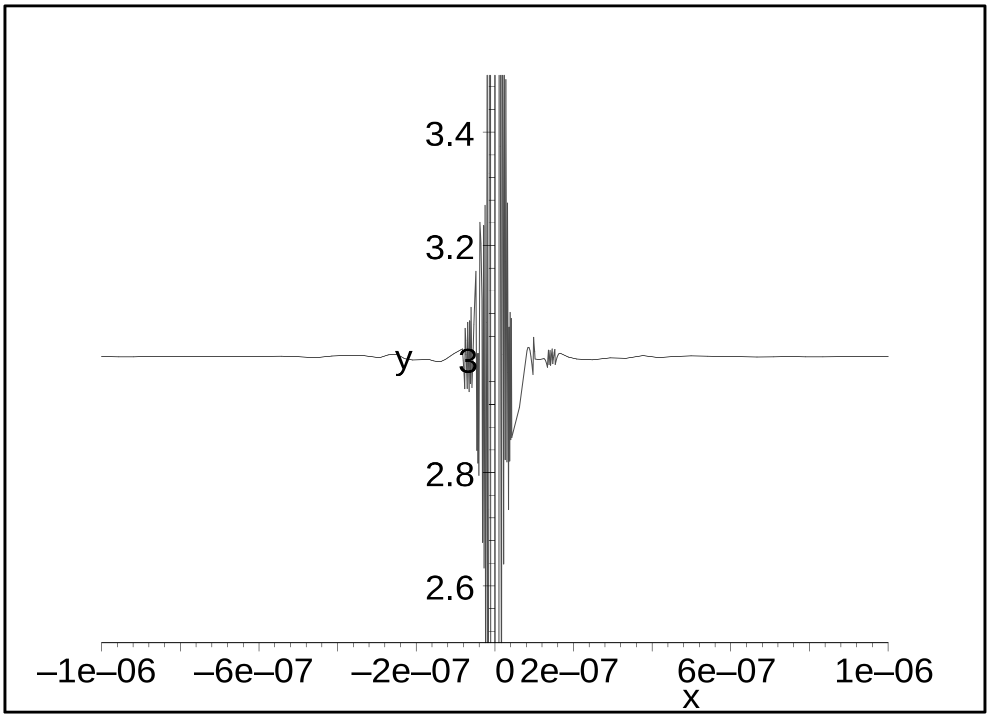
Si lo resolvemos mediante técnicas analíticas tenemos que el límite es:

```
> exp(1.1);
```

$$3.004166024$$

En cambio observamos que a medida que disminuimos la ventana de observación

```
>   plot(f(x),x=-0.000001..0.000001,y=2.5..3.5);
```



Nota muy importante: En general no podremos dar el error que cometemos con exactitud. Esto implicaria conocer la solución exacta al problema que estamos tratando y, si la conocemos, no tiene sentido que busquemos aproximaciones. De manera que tendremos que lo más, y fundamental, que podemos hacer para asegurarnos que nuestras aproximaciones estan bien, es saber una cota lo más pequeña posible para el error.

6.2 Ejercicio 1

(a) Sea f(x)=(x-sin(x))/x^3. Haz un conjetura del límite de f cuando nos acercamos a cero por la derecha evaluando f en x=0.1, 0.01,0.001,0.001.

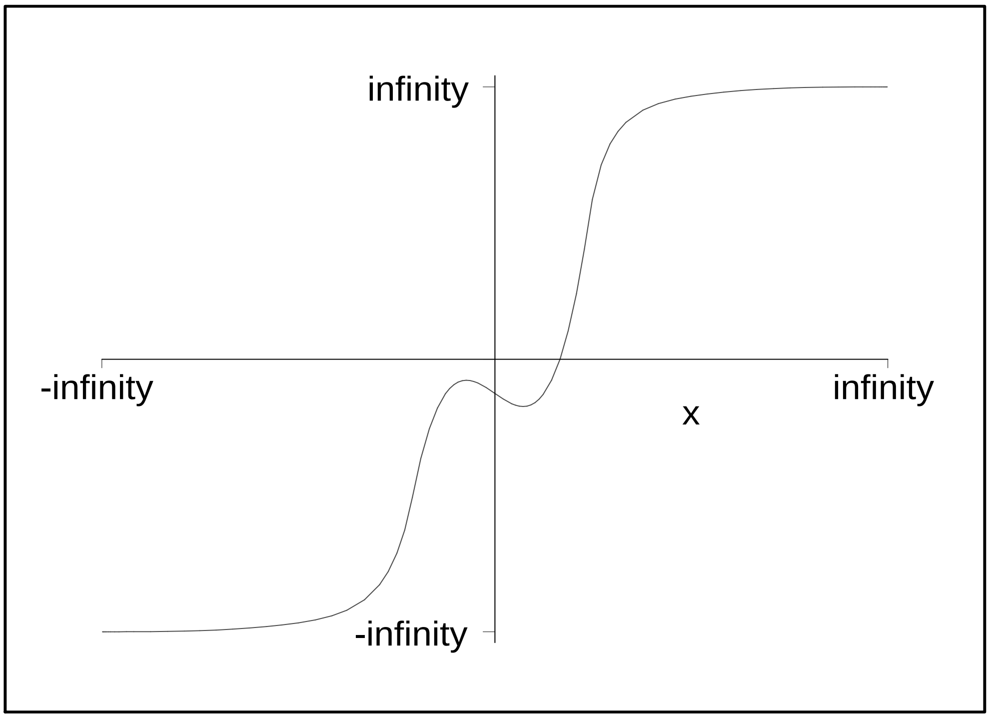
(b) Evalua f en los puntos x=0.00001, 1e-6,1e-7,1e-8,1e-9 y 1e-10 y haz otra conjetura.

(c) Pinta la gráfica de f entre x=-0.1..0.1 y otra entre x=-0.000001..0.000001.

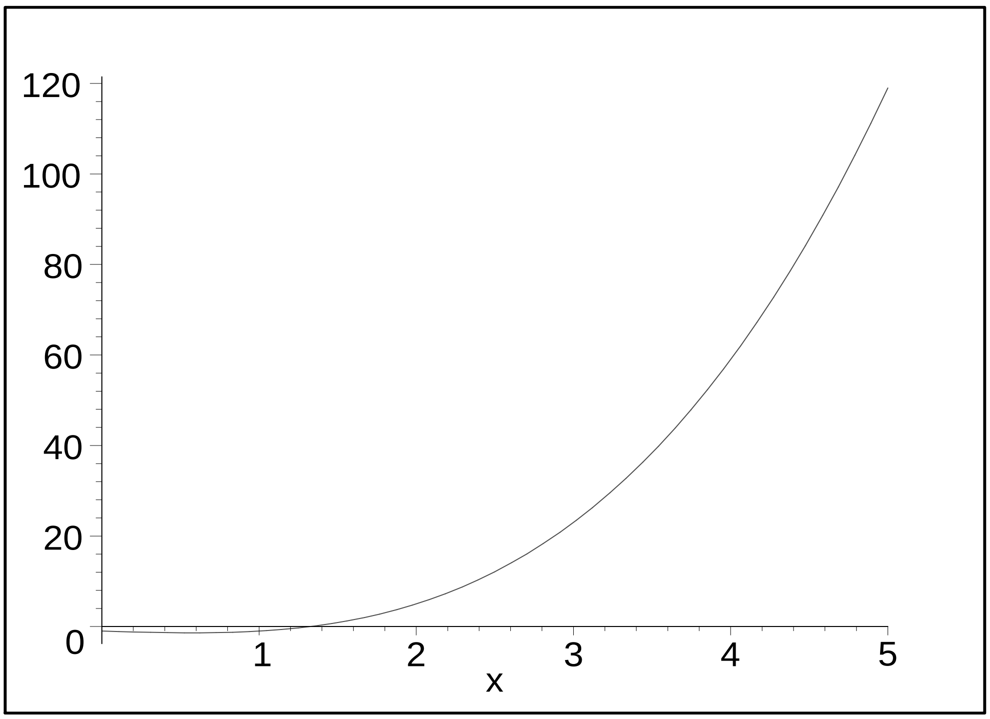
(d) ¿Cuál es el límite cuando nos acercamos a 0?

6.3 Resolución gráfica de ecuaciones.

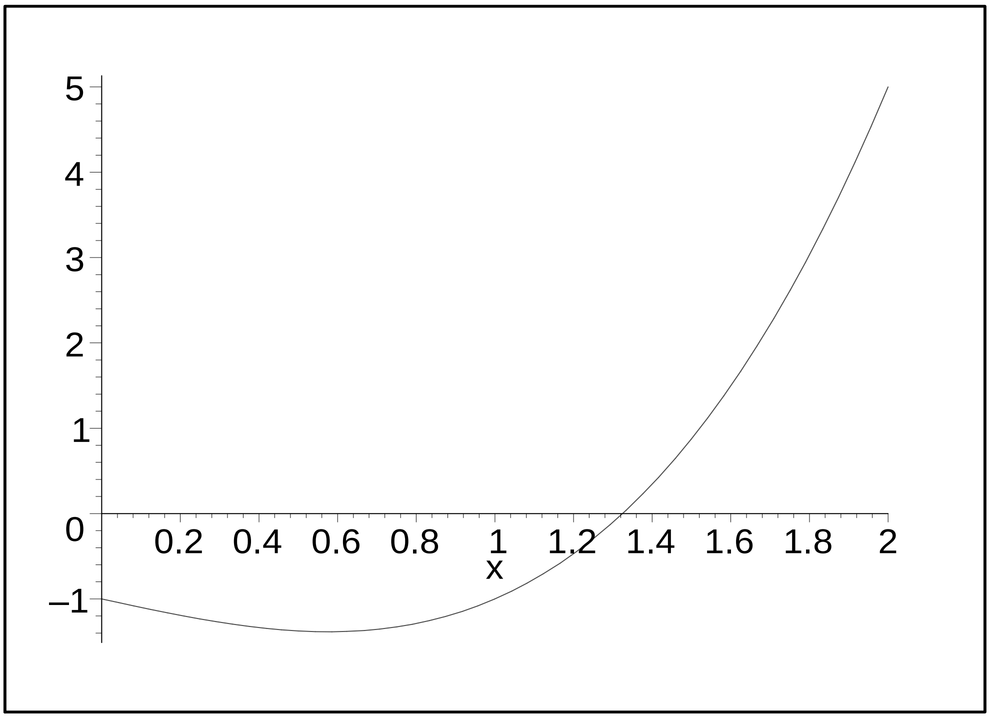
```
>   restart:
>   p1:=x^3-x-1:
>   plot(p1,x=-infinity..infinity);
```



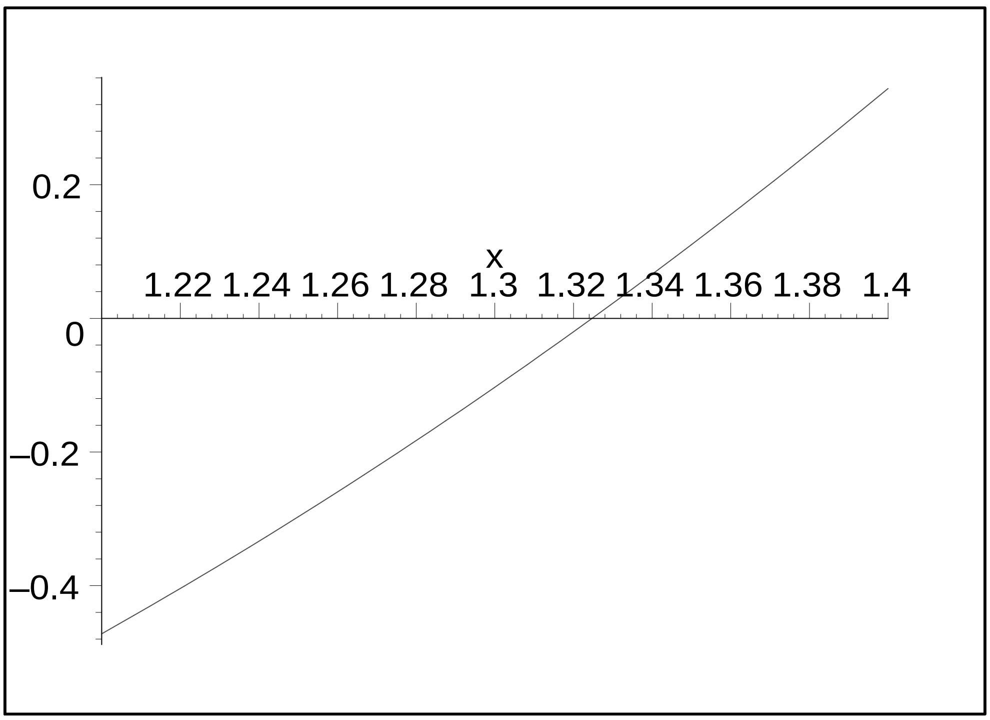

```
>  plot(p1,x=0..5);
```



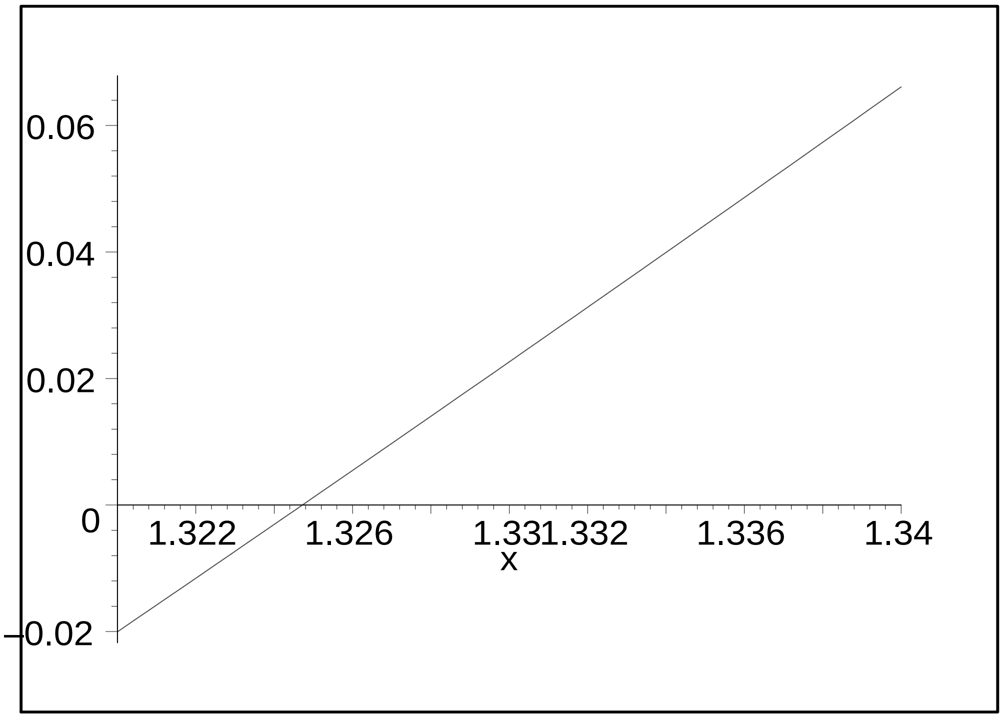

```
>  plot(p1,x=0..2);
```



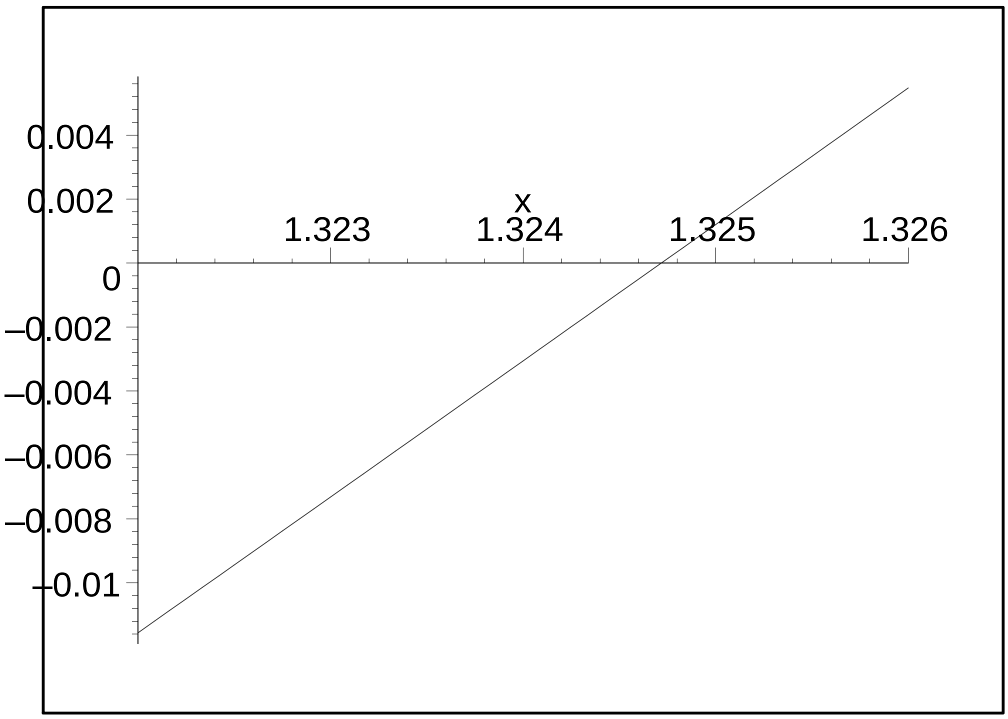

```
> plot(p1,x=1.2..1.4);
```



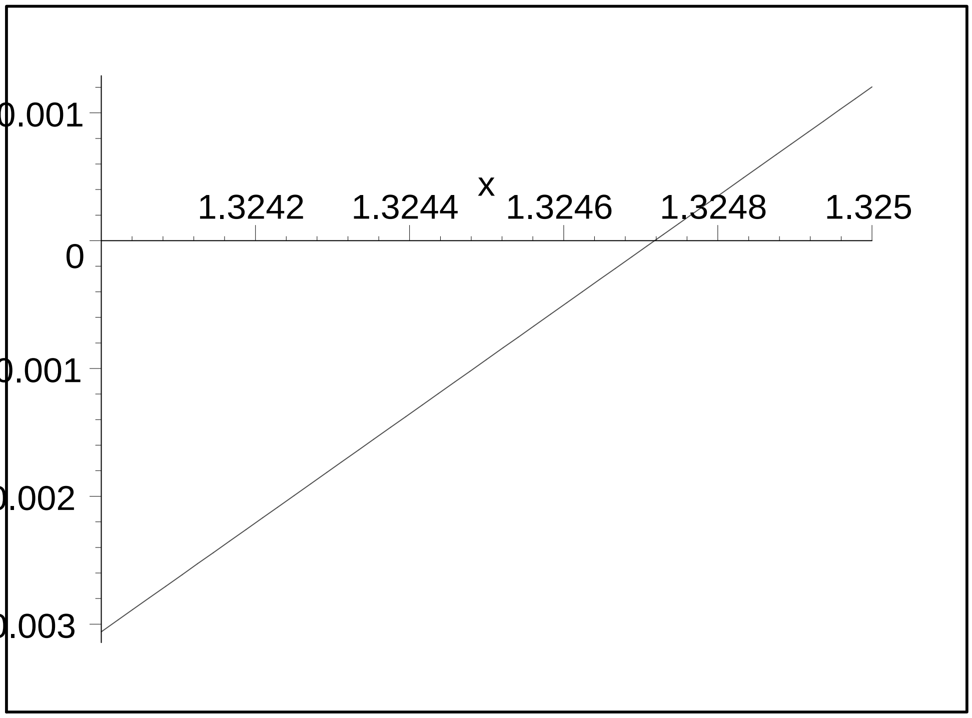

```
> plot(p1,x=1.32..1.34);
```




```
> plot(p1,x=1.322..1.326);
```




```
> plot(p1,x=1.324..1.325);
```



la **solución** es aproximadamente 1.324 y estamos cometiendo un error absoluto menor que 0.001, hemos dado tres cifras decimales exactas.

```
> evalf(solve(p1=0));
```

$$1.324717958,\ -0.6623589786 + 0.5622795125\,I,\ -0.6623589786 - 0.5622795125\,I$$

6.4 Ejercicio 2

Encuentra todas las raices del polinomio

p(x)=x^4+4x^3+x^2-6x+2

Encuentra tantos intervalos, disjuntos, como soluciones tenga p(x)=0 de manera que en cada intervalo haya una única raiz.

6.5 Teorema de Bozano: tanteo.

```
> p:=x->3*x^3-8*x^2+x+3:
> p(0);
```

$$3$$

```
> p(1);
```

$$-1$$

Voy a calcular la solución p(x)=0 que hay entre [0,1]

```
> p(0.5);
```

$$1.875$$

```
> p(0.75);
```

$$0.515625$$

```
> p(0.9);
```
$$-0.393$$
```
> p(0.8);
```
$$0.216$$
```
> p(0.85);
```
$$-0.087625$$
```
> p(0.82);
```
$$0.094904$$
```
> p(0.84);
```
$$-0.026688$$
```
> p(0.83);
```
$$0.034161$$
```
> p(0.835);
```
$$0.003748625$$
```
> p(0.837);
```
$$-0.008423241$$

La raiz es aproximadamente 0.83, estamos cometiendo un error absoluto menor o igual que 0.005 (o estrictamente menor que 0.01).

6.6 Ejercicio 3

Encuentra una solución de sin(x)=x+1 con un error menor que 0.001.

6.7 Teorema de Bolzano: bisección o bipartición.

```
> p2:=x->5-x-x^4:
> p2(0);
```
$$5$$
```
> p2(2);
```
$$-13$$

Al menos hay una raiz entre [0,2]

```
> p2(1);
```
$$3$$
```
> p2(1.5);
```
$$-1.5625$$
```
> p2(1.25);
```
$$1.30859375$$
```
> (1.5+1.25)/2;
```
$$1.375000000$$
```
> p2(1.375);
```
$$0.050537109$$
```
> (1.5+1.375)/2;
```
$$1.437500000$$
```
> p2(1.4375);
```
$$-0.707534790$$
```
> (1.375+1.4375)/2;
```
$$1.406250000$$

```
> p2(1.40625);
```
$$-0.316910744$$
```
> (1.375+1.40625)/2;
```
$$1.390625000$$
```
> p2(1.390625);
```
$$-0.130353987$$
```
> (1.390625+1.375)/2;
```
$$1.382812500$$
```
> p2(1.3828125);
```
$$-0.039208177$$
```
> (1.375+1.3828125)/2;
```
$$1.378906250$$
```
> p2(1.378906250);
```
$$0.005838543$$
```
> (1.3828125+1.378906250)/2;
```
$$1.380859375$$
```
> p2(1.380859375);
```
$$-0.016641175$$
```
> (1.378906250+1.380859375)/2;
```
$$1.379882812$$
```
> p2(1.379882812);
```
$$-0.005390415$$

Como p2(1.378906250)>0 y p2(1.379882812)<0 sabemos que la ecuación p2=0 va a tener una solución aproximada de 1.37, con dos cifras decimales exactas y cometiendo un error absoluto menor que 0.01 (o menor o igual que 0.005)

6.8 Ejercicio 5

Encuentra una solución de sin(x)=x+1 mediante el algoritmo de la bisección, con un error absoluto menor o igual que 0.01 ¿Cuántas subdivisiones del intervalo inicial has necesitado?.

6.9 Problemas

6.10 Programa 1

Programa un procedimiento decida si se puede o no aplicar Bolzano a una función *f en un cierto intervalo [a,b].*

Aplicaselo a las funciones de los ejercicios anteriores en los intervalos que hayas elegido.

6.11 Programa 2

Programa un procedimiento que dada una función continua ***f*** , a la que se le pueda aplicar Bolzano en el intervalo ***[a,b]*** te calcule la solución mediante el algoritmo de la bisección con tantas iteraciones ***n*** como tu le digas.

Aplicaselo a las funciones de los ejercicios anteriores en los intervalos que hayas elegido.

6.12 Programa 3

Programa un procedimiento que dada una función continua ***f*** , a la que se le pueda aplicar Bolzano en el intervalo ***[a,b]*** te calcule la solución con un error menor que el error ***err*** dado mediante el algoritmo de la bisección.

Aplicaselo a las funciones de los ejercicios anteriores en los intervalos que hayas elegido.

Capítulo 7

Método de Newton de Resolución de ecuaciones

7.1 Ejercicio 1. Método de Newton.

Programa un procedimiento que haga n iteraciones del método de Newton, para una función dada empezando en un cierto punto inicial.

7.2 Ejercicio 2. Comparación de los métodos de Bisección y Newton en la búsqueda de raices.

(a) Aplica el procedimiento de Bisección iterativo de la práctica anterior y el de Newton del problema anterior, con 10 iteraciones cada uno, a la función xˆ6-x-1, empezando en el intervalo [1,2] para el método de la bisección y en el punto 1.5 para el método de Newton.

(b) Compara los valores de la aproximación en ambos métodos. ¿Cuál es tu conclusión?

7.3 Ejercicio 3. Comparación del error en los métodos de Bipartición y Newton

(a) Sea f(x)=xˆ3-xˆ2-x-1. Calcula f(x)=0. Describe la solución con la mayor cantidad de cifras exactas utilizando las versiones iterativas vistas en clase de los métodos de Bisección en el intervalo [1,2] y Newton empezando en 1.5 utilizando tantas iteraciones como sean necesarias.

(b) Comprueba que el error en el método de Bisección es como (b-a)/2ˆn, mientras que en Newton es del mismo orden que el cuadrado del paso anterior.

Capítulo 8

Algunas soluciones

8.1 Solución Bolzano

```
> restart:
> bolz1:=proc(f,a,b,er)
> if evalf(f(a))=0 then print(`a es una raiz`);
> elif evalf(f(b))=0 then print (`b es una raiz`);
> elif evalf(f(a)*f(b))>0 then print (`no podemos aplicar Bolzano en
> ese intervalo`);
> else bolz2(f,a,b,er);
> fi:
> end:
> bolz2:=proc(f,a,b,er)
> local m;
> m:=(a+b)/2;
> if abs(m-a)<er then print(`la solución es`,m,`con un error menor
> de`,er); elif evalf(f(m))=0 then print(m,`es una raiz`); elif
> evalf(f(a)*f(m))>0 then bolz2(f,m,b,er):  else bolz2(f,a,m,er):  fi:
> end:
> f:=x->x^3-2*x+1:
> bolz1(f,-3.,3.,0.00001);
```

$$\textit{la solucin es}, -1.618028642, \textit{con un error menor de}, 0.00001$$

```
> bolz3:=proc(f,a,b,n) local m,i,k,l:  k:=a:
> l:=b:  for i to n do
> m:=(k+l)/2:
> if f(k)*f(m)<0 then l:=m:
> else k:=m fi:
> print(i,m):  od:  print(`Estamos cometiendo un error menor que
> :`,(b-a)/2^i);
> end:
> bolz3(f,-3.,0.,20);
```

$$1, -1.500000000$$
$$2, -2.250000000$$
$$3, -1.875000000$$
$$4, -1.687500000$$
$$5, -1.593750000$$
$$6, -1.640625000$$
$$7, -1.617187500$$
$$8, -1.628906250$$
$$9, -1.623046875$$

$$10, \ -1.620117188$$
$$11, \ -1.618652344$$
$$12, \ -1.617919922$$
$$13, \ -1.618286133$$
$$14, \ -1.618103028$$
$$15, \ -1.618011475$$
$$16, \ -1.618057252$$
$$17, \ -1.618034364$$
$$18, \ -1.618022920$$
$$19, \ -1.618028642$$
$$20, \ -1.618031503$$
$$\mathit{Estamos\ cometiendo\ un\ error\ menor\ que}\ :,\ 0.1430511475\,10^{-5}$$

8.2 Solución al programa que pinta una recta dados un par de puntos

```
> restart:
```

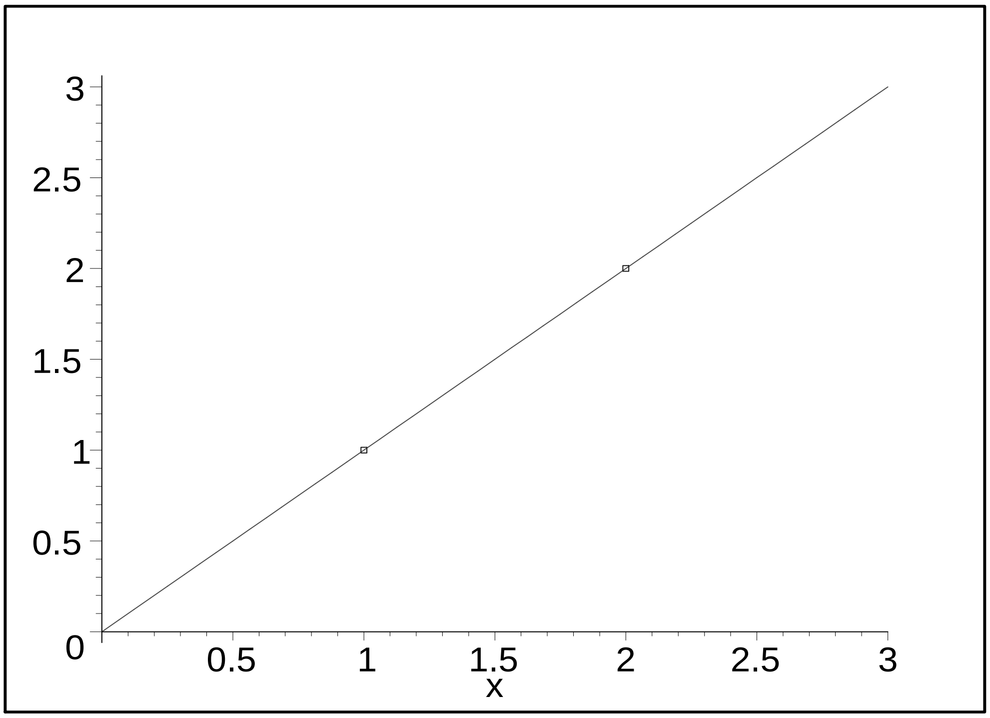
```
> with(plots):
```

```
Warning, the name changecoords has been redefined
```

```
> recta:=proc(a,b,c,d)
> local m, p, d1,d2:
> m:=(d-b)/(c-a); print(`pendiente:`,m);
> p:=b-m*a; print(`corte :`,p);
> d1:=plot(m*x+p,x=a-1..c+1,thickness=2):
> d2:=pointplot({[a,b],[c,d]},symbol=box,color=blue):
> display(d1,d2);
> end:
```

```
> recta(1,1,2,2);
```

$$\mathit{pendiente}\ :,\ 1$$
$$\mathit{corte}\ :,\ 0$$



8.3 Solución al programa de sucesiones

```
> restart:
```

Primero escribimos un progrtama que nos de idea de hacia donde va la sucesion, del posible limite. Comparar este programa con el comando **seq**

```
> calculo_limite:=proc(suc,n,paso)
> local i:
> for i to n by paso do print(evalf(suc(i))) od;
> end:
> suc1:=n->n/(1+n);
```

$$suc1 := n \to \frac{n}{1+n}$$

```
> calculo_limite(suc1,10,1);
```

$$0.5000000000$$
$$0.6666666667$$
$$0.7500000000$$
$$0.8000000000$$
$$0.8333333333$$
$$0.8571428571$$
$$0.8750000000$$
$$0.8888888889$$
$$0.9000000000$$
$$0.9090909091$$

```
> seq(n/(1+n),n=1..10);
```

$$\frac{1}{2}, \frac{2}{3}, \frac{3}{4}, \frac{4}{5}, \frac{5}{6}, \frac{6}{7}, \frac{7}{8}, \frac{8}{9}, \frac{9}{10}, \frac{10}{11}$$

```
> suc2:=n->(2/(3*n));
```

$$suc2 := n \to \frac{2}{3}\frac{1}{n}$$

```
> calculo_limite(suc2,10,1);
```

$$0.6666666667$$
$$0.3333333333$$
$$0.2222222222$$
$$0.1666666667$$
$$0.1333333333$$
$$0.1111111111$$
$$0.09523809524$$
$$0.08333333333$$
$$0.07407407407$$
$$0.06666666667$$

```
> seq(2/(3*n),n=1..10);
```

$$\frac{2}{3}, \frac{1}{3}, \frac{2}{9}, \frac{1}{6}, \frac{2}{15}, \frac{1}{9}, \frac{2}{21}, \frac{1}{12}, \frac{2}{27}, \frac{1}{15}$$

Una vez sospechamos el valor del limite construimos el procedimiento que me calcula a partir de que término (el N grande de la definición) de la sucesión dista del límite menos que el error (el epsilon de la definicion) que le demos.

```
> calculo_termino:=proc(suc,lim,err)
> local i:
> for i while abs(suc(i)-lim)>err do print(i,evalf(abs(suc(i)-lim)));
> od;
> end:
> calculo_termino(suc1,1,1e-1);
```

$$1, 0.5000000000$$
$$2, 0.3333333333$$
$$3, 0.2500000000$$
$$4, 0.2000000000$$
$$5, 0.1666666667$$
$$6, 0.1428571429$$
$$7, 0.1250000000$$
$$8, 0.1111111111$$

```
> calculo_termino(suc1,1,1);
> calculo_termino(suc1,1,1e-2);
```

$$1, 0.5000000000$$
$$2, 0.3333333333$$
$$3, 0.2500000000$$
$$4, 0.2000000000$$
$$5, 0.1666666667$$
$$6, 0.1428571429$$
$$7, 0.1250000000$$
$$8, 0.1111111111$$
$$9, 0.1000000000$$
$$10, 0.09090909091$$
$$11, 0.08333333333$$

12, 0.07692307692
13, 0.07142857143
14, 0.06666666667
15, 0.06250000000
16, 0.05882352941
17, 0.05555555556
18, 0.05263157895
19, 0.05000000000
20, 0.04761904762
21, 0.04545454545
22, 0.04347826087
23, 0.04166666667
24, 0.04000000000
25, 0.03846153846
26, 0.03703703704
27, 0.03571428571
28, 0.03448275862
29, 0.03333333333
30, 0.03225806452
31, 0.03125000000
32, 0.03030303030
33, 0.02941176471
34, 0.02857142857
35, 0.02777777778
36, 0.02702702703
37, 0.02631578947
38, 0.02564102564
39, 0.02500000000
40, 0.02439024390
41, 0.02380952381
42, 0.02325581395
43, 0.02272727273
44, 0.02222222222
45, 0.02173913043
46, 0.02127659574
47, 0.02083333333
48, 0.02040816327
49, 0.02000000000
50, 0.01960784314
51, 0.01923076923
52, 0.01886792453
53, 0.01851851852
54, 0.01818181818
55, 0.01785714286

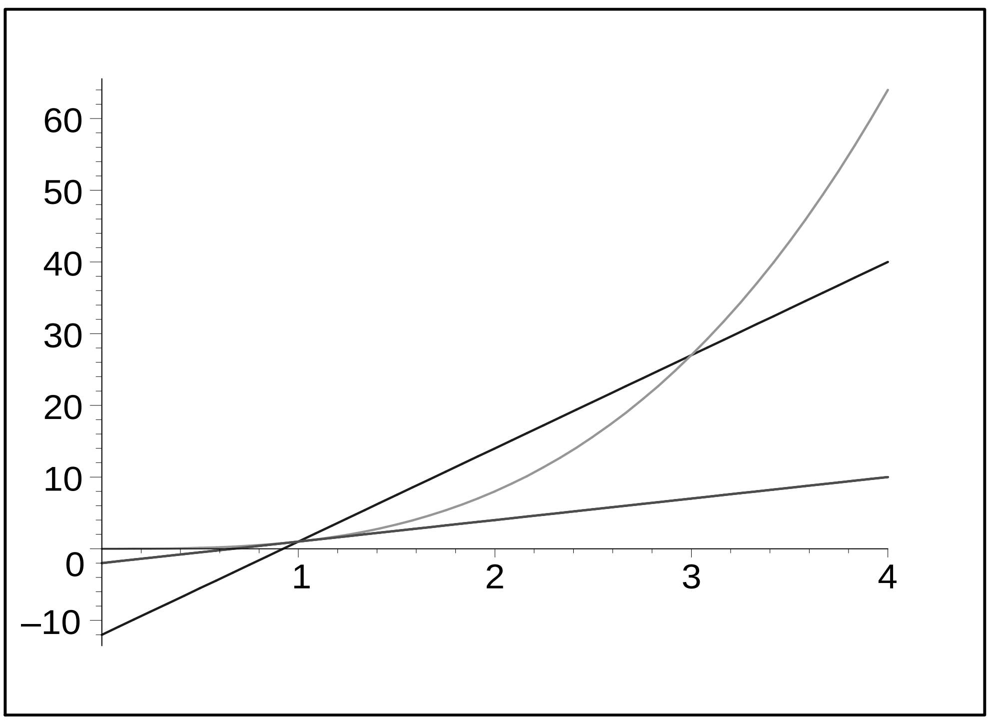
56, 0.01754385965
57, 0.01724137931
58, 0.01694915254
59, 0.01666666667
60, 0.01639344262
61, 0.01612903226
62, 0.01587301587
63, 0.01562500000
64, 0.01538461538
65, 0.01515151515
66, 0.01492537313
67, 0.01470588235
68, 0.01449275362
69, 0.01428571429
70, 0.01408450704
71, 0.01388888889
72, 0.01369863014
73, 0.01351351351
74, 0.01333333333
75, 0.01315789474
76, 0.01298701299
77, 0.01282051282
78, 0.01265822785
79, 0.01250000000
80, 0.01234567901
81, 0.01219512195
82, 0.01204819277
83, 0.01190476190
84, 0.01176470588
85, 0.01162790698
86, 0.01149425287
87, 0.01136363636
88, 0.01123595506
89, 0.01111111111
90, 0.01098901099
91, 0.01086956522
92, 0.01075268817
93, 0.01063829787
94, 0.01052631579
95, 0.01041666667
96, 0.01030927835
97, 0.01020408163
98, 0.01010101010

```
> calculo_termino(suc2,0,1e-1);
```

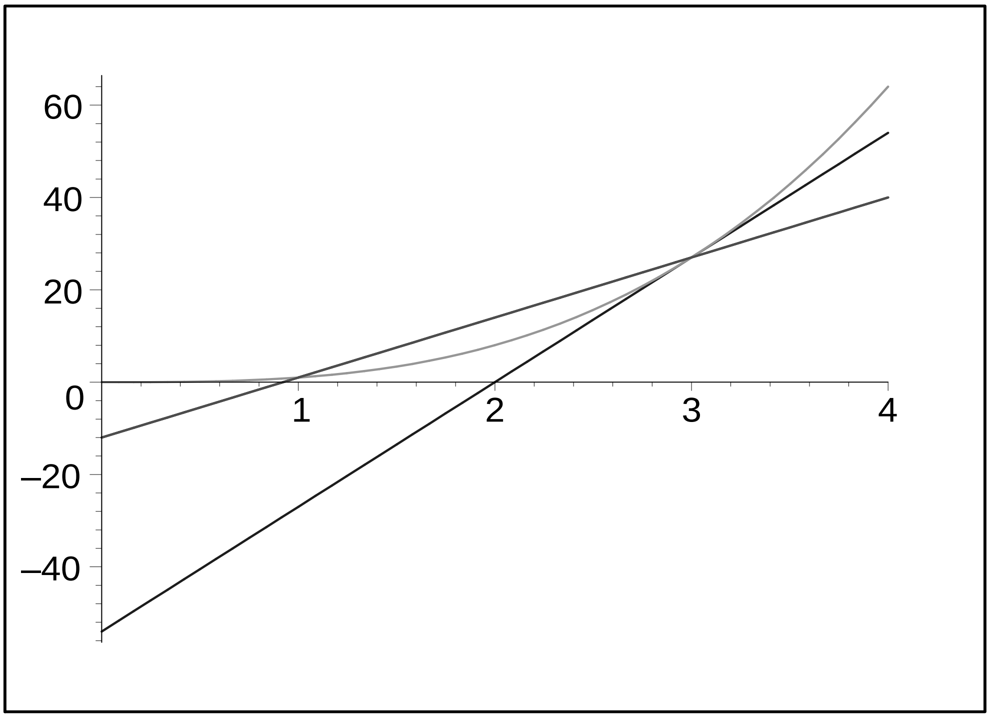
$$\begin{array}{c} 1,\ 0.6666666667 \\ 2,\ 0.3333333333 \\ 3,\ 0.2222222222 \\ 4,\ 0.1666666667 \\ 5,\ 0.1333333333 \\ 6,\ 0.1111111111 \end{array}$$

8.4 Solución al programa de las secantes

```
> restart:
> with(plots):
Warning, the name changecoords has been redefined
> tangente:=proc(f,a,c)
> local mt,ms,mi,md,izq,dcha,d1,d2,d3,d4:
> mt:=D(f)(a):
> ms:=(f(a)-f(c))/(a-c):
> mi:=min(ms,mt):
> md:=max(ms,mt):
> izq:=min(a,c)-1:
> dcha:=max(a,c)+1:
> d1:=plot(f,izq..dcha,thickness=3,color=green):
> d2:=plot(mt*(x-a)+f(a),x=izq..dcha,thickness=3,color=blue):
> d3:=plot(ms*(x-a)+f(a),x=izq..dcha,thickness=3,color=blue):
> d4:=animate(m*(x-a)+f(a),x=izq..dcha,m=mi..md,thickness=3):
> display(d1,d2,d3,d4):
> end:
> f:=x->x^3:
> tangente(f,1,3);
```



```
> tangente(f,3,1);
```



8.5 Solución al método de Newton

```
> restart:
> MetodoNewton:=proc(f,x,n) local i,x0,x1:
> x0:=x:  for i to n do x1:=x0-f(x0)/(D(f)(x0)):  x0:=x1:  print(`En la
> iteración:  `,i,`la solución es aproximadamente:  `,x1):  od:
> end:
> f:=x->x^3-2*x+1:
> MetodoNewton(f,3.,10);
```

$$\mathit{En\ la\ iteracin} : , 1, \mathit{la\ solucin\ es\ aproximadamente} : , 2.120000000$$
$$\mathit{En\ la\ iteracin} : , 2, \mathit{la\ solucin\ es\ aproximadamente} : , 1.572406298$$
$$\mathit{En\ la\ iteracin} : , 3, \mathit{la\ solucin\ es\ aproximadamente} : , 1.250682507$$
$$\mathit{En\ la\ iteracin} : , 4, \mathit{la\ solucin\ es\ aproximadamente} : , 1.081716611$$
$$\mathit{En\ la\ iteracin} : , 5, \mathit{la\ solucin\ es\ aproximadamente} : , 1.013986428$$
$$\mathit{En\ la\ iteracin} : , 6, \mathit{la\ solucin\ es\ aproximadamente} : , 1.000546177$$
$$\mathit{En\ la\ iteracin} : , 7, \mathit{la\ solucin\ es\ aproximadamente} : , 1.000000892$$
$$\mathit{En\ la\ iteracin} : , 8, \mathit{la\ solucin\ es\ aproximadamente} : , 1.000000000$$
$$\mathit{En\ la\ iteracin} : , 9, \mathit{la\ solucin\ es\ aproximadamente} : , 1.000000000$$
$$\mathit{En\ la\ iteracin} : , 10, \mathit{la\ solucin\ es\ aproximadamente} : , 1.000000000$$

Capítulo 9

Inversas, Polinomio de Taylor en una variable, Límites en una y varias variables, Polinomio de Taylor en varias variables, Derivación implícita en una y varias variables

9.1 Inversas

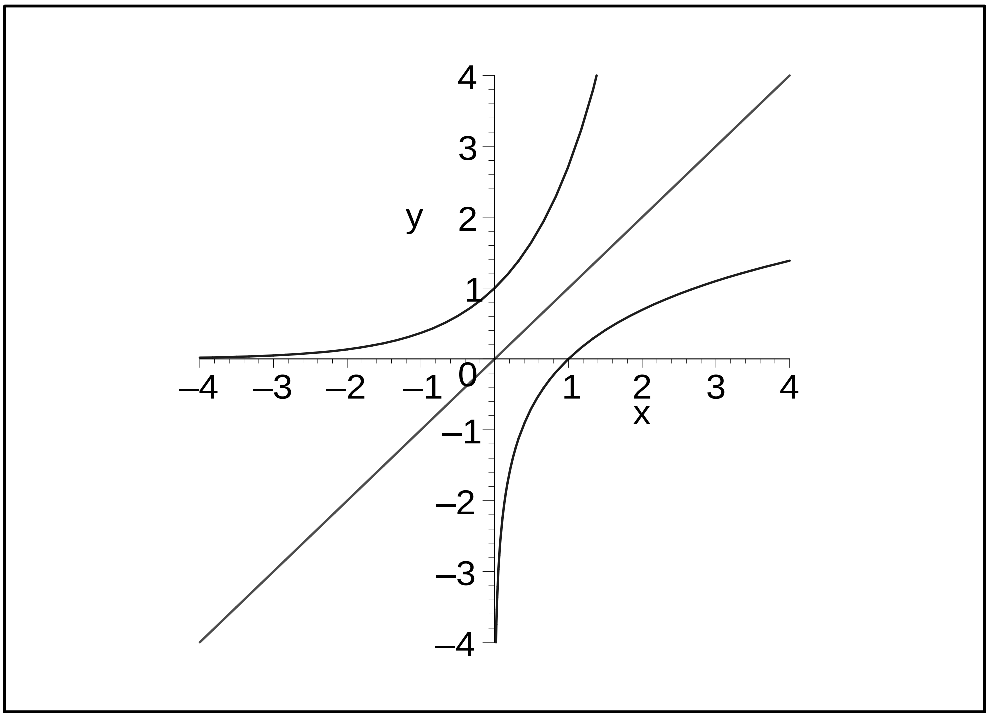
```
> restart:
```

9.2 Problema 1:

Observando las siguientes gráficas de una función, su inversa y la función y=x, ¿podrías explicar que relación hay entre ellas?

También pintaremos la tangente en un punto y es su correspondiente en la función inversa. ¿Podráis explicar que relación hay entre ellas?

```
> plot([x,log(x),exp(x)],x=-4..4,y=-4..4,color=[red,blue,blue],thicknes
> s=3,scaling=constrained);
```



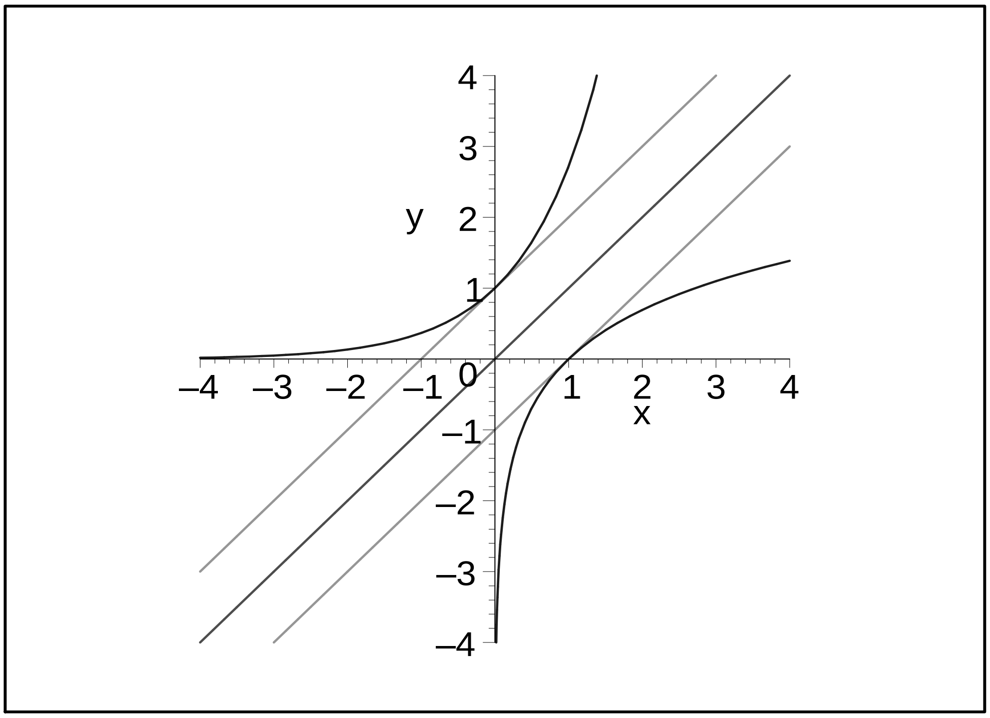

```
> with(plots):
```

```
Warning, the name changecoords has been redefined
```

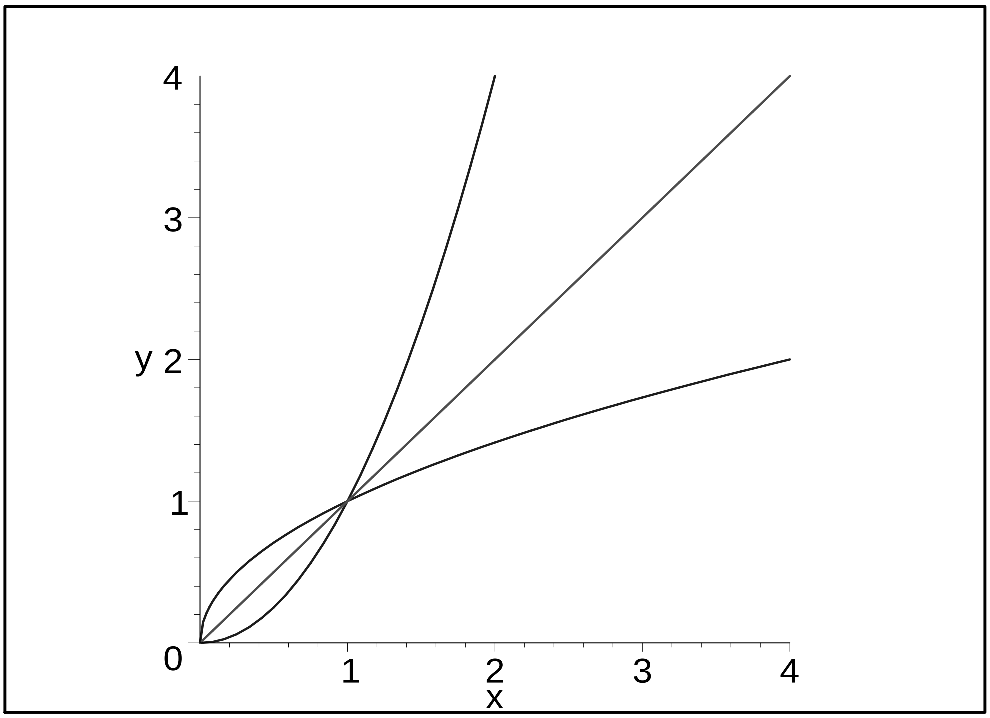
```
> d1:=plot([x,log(x),exp(x)],x=-4..4,y=-4..4,color=[red,blue,blue],thic
> kness=3,scaling=constrained):
```

```
> d2:=plot([x+1,x-1],x=-4..4,y=-4..4,color=[green,green],thickness=3,sc
> aling=constrained):
```

```
> display(d1,d2);
```

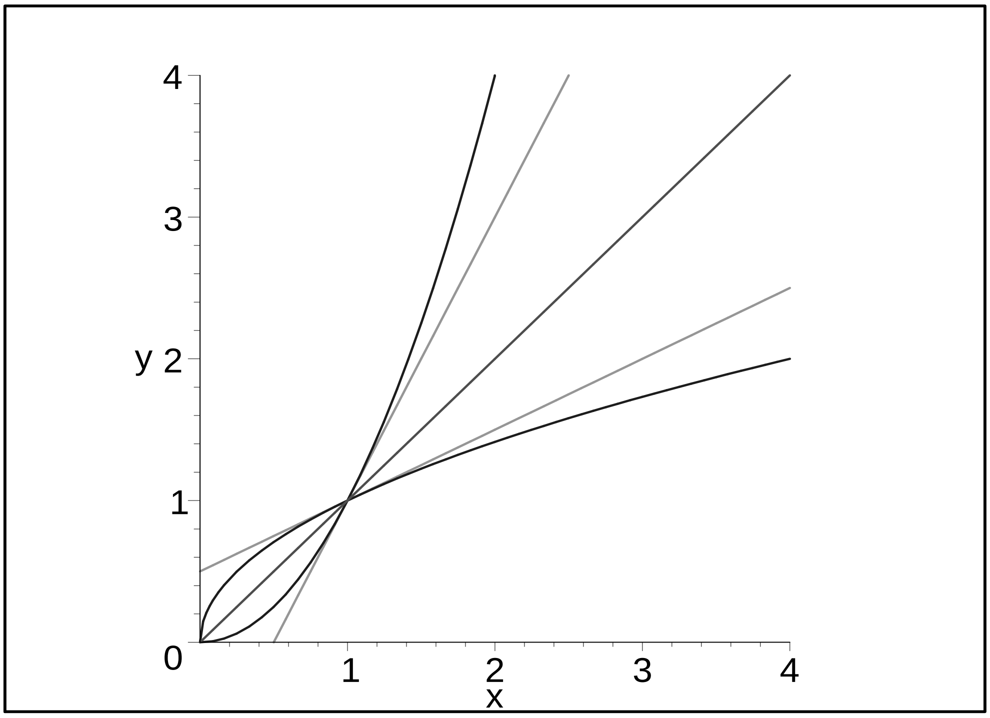



```
> plot([x,sqrt(x),x^2],x=0..4,y=0..4,color=[red,blue,blue],thickness=3,
> scaling=constrained);
```




```
> d3:=plot([x,sqrt(x),x^2],x=0..4,y=0..4,color=[red,blue,blue],thicknes
> s=3,scaling=constrained):
> d4:=plot([1/2*x+1/2,2*x-1],x=0..4,y=0..4,color=[green,green],thicknes
> s=3,scaling=constrained):
```

```
> display(d3,d4);
```



9.3 Problema 2

Resuelve el problema 46 de la hoja 5 (pintar las gráficas entre 0 y 20), y calcula las rectas tangentes a f en los puntos (1/2, 1/2), (2,4) y (4.5, 16.97056275) y en los correspondientes puntos de la inversa.

9.4 Polinomios de Taylor

El comando que calcula el polinomio de Taylor de grado n de una función f en el punto a es taylor(f(x),x=a,n)

```
> taylor(cos(x),x=0,8);
```

$$1 - \frac{1}{2}x^2 + \frac{1}{24}x^4 - \frac{1}{720}x^6 + \mathrm{O}(x^8)$$

9.5 Problema 3

Calcula los 5 primeros polinomios de Taylor de la función log(1+x) en x=0, pintalos en una gráfica junto con la función y da para cada polinomio el intervalo de mejor aproximación.

9.6 Problema 4

Resuelve el problema 55 de las hojas.

9.7 Límites en una y varias variables

Maple en el cálculo de límites no es demasiado eficaz, en una variable si

```
> limit(sin(x)/x,x=0);
```

$$1$$

```
> limit(log(x),x=0);
```

$$-\infty$$

```
> limit(abs(x)/x,x=0);
```

$$\mathit{undefined}$$

Pero en varias variables límites muy sencillos no son resueltos.

```
> limit((x*y)/(x^2+y^2),{x=0,y=0});
```

$$\mathrm{limit}(\frac{x\,y}{x^2+y^2},\,\{x=0,\,y=0\})$$

```
> limit((x^2*y^2)/(x^2+y^2),{x=0,y=0});
```

$$\mathrm{limit}(\frac{x^2\,y^2}{x^2+y^2},\,\{x=0,\,y=0\})$$

9.8 Polinomios de Taylor en varias variables. Plano tangente.

El polinomio de Taylor en varias variables de una función adecuada f(x,y) en el punto (a,b) de grado 2 es

$$\mathrm{P}(x,\,y)=\mathrm{f}(a,\,b)+\mathrm{D}_1(f)(a,\,b)\,(x-a)+\mathrm{D}_2(f)(a,\,b)\,(y-b)+\frac{1\,\mathrm{D}_{1,\,1}(f)(a,\,b)\,(x-a)^2}{2}+(x-a)\,\mathrm{D}_{1,\,2}(f)(a,\,b)\,(y-b)+\frac{1\,\mathrm{D}_{2,\,2}(f)(a,\,b)\,(y-b)^2}{2}$$

La orden que calcula el polinomio de Taylor en varias variables es **mtaylor,** similar al comando **taylor** que conociamos para una variable.

```
> mtaylor(exp(x*y),{x=0,y=0},5);
```

$$1+x\,y+\frac{1}{2}\,x^2\,y^2$$

El polinomio de Taylor de Primer grado nos da una aproximación lineal a la función, el polinomio de Taylor de segundo grado nos da una aproximación cuadrática de la función, etc.

Al plano definido por esta aproximación lineal z=P(x,y), con P el polinomio de Taylor de primer grado de la función f en el punto (a,b), se le llama **plano tangente a f en el punto (a,b)**.

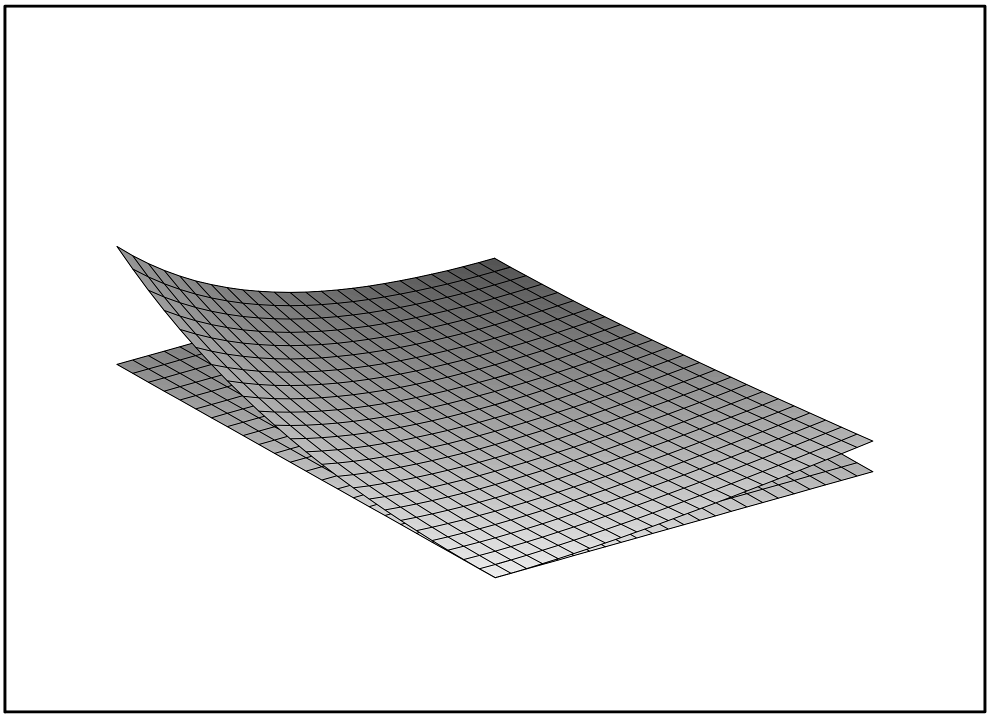
9.9 Ejemplo:

Vamos a calcular el plano tangente en el origen a z=exp(x-y) y lo dibujamos junto a la función

```
> p1:=mtaylor(exp(x-y),{x=0,y=0},2);
```

$$\mathit{p1} := 1+x-y$$

```
> plot3d({exp(x-y),p1},x=-1..1,y=-1..1);
```



9.10 Problemas 5, 6, 7 y 8

Calcular el plano tangente de las siguientes funciones en el punto que se indica. Pinta la función y su plano tangente.

(5) z=cos(x+y) en el origen

(6) z=x^2-y^2 en (5,4,9)

(7) z=exp(sin(y)+1) en (0,Pi/2,2)

(8) z=x^3+y^3-3xy en (1,2,3)

9.11 Derivación implicita. Regla de la cadena.

Al igual que vimos en una variable, a veces queremos obtener la derivada de una función que viene definida implicitamente dependiendo de dos o más variables. Para resolver esto utilizamos el comando **implicitdiff,** por ejemplo: supongamos que queremos derivar con respecto a x, la función z(x,y) definida implicitamente por la ecuación f(x,y,z)=0, de manera que por la regla de la cadena tenemos que D_xf(x,y,z(x,y))+D_zf(x,y,z(x,y)*D_z=0, y esto es lo que nos da Maple

```
> restart:
```

Aqui con una variable

```
> implicitdiff(x^2+y^2=1,y(x),x);
```

$$-\frac{x}{y}$$

Aquí con más variables

```
> implicitdiff(f(x,y,z)=0,z(x,y),x);
```

$$-\frac{D_1(f)(x,\, y,\, z)}{D_3(f)(x,\, y,\, z)}$$

Por ejemplo: si x^2+y^2+z^2=0 define implicitamente una cierta función z(x,y) y queremos clacular la derivada con respecto a y, sería: 2y+2zD_yz(x,y)=0, de manera que D_yz(x,y)=-y/z y asi es como lo hacemos con Maple.

```
> implicitdiff(x^2+y^2+z^2=0,z(x,y),y);
```

$$-\frac{y}{z}$$

Otro ejemplo z^2-2*x^2-2*y^2=12

La derivada de z con respecto a x, z_x, será 2z*z_x-4*x=0, es decir: z_x=2*x/z. Lo vemos con Maple:

```
> implicitdiff(z^2-2*x^2-2*y^2=12,z(x,y),x);
```

$$\frac{2\,x}{z}$$

9.12 Problemas 9 y 10

Deriva con respecto a x e y, y comprueba los resultados con Maple, las siguientes funciones z(x,y) definidas implicitamente:

(9) x^2+4^*y^2+z^2=36

(10) x^2+2*z^2=y^2

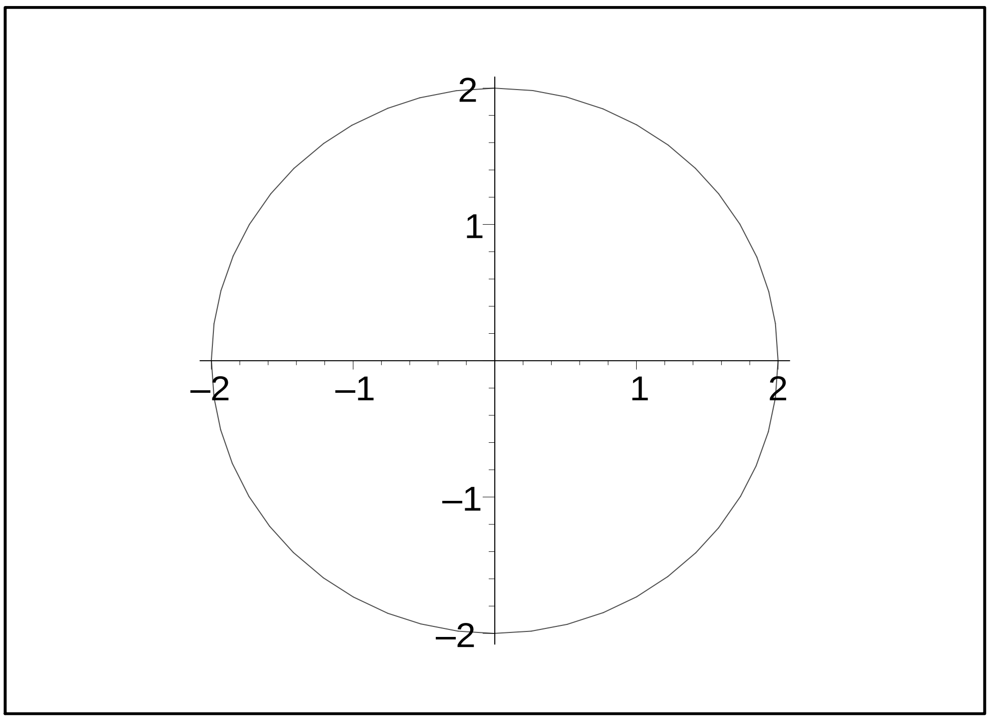
Capítulo 10

Coordenadas polares, cilíndricas y esféricas.

10.1 Coordenadas polares

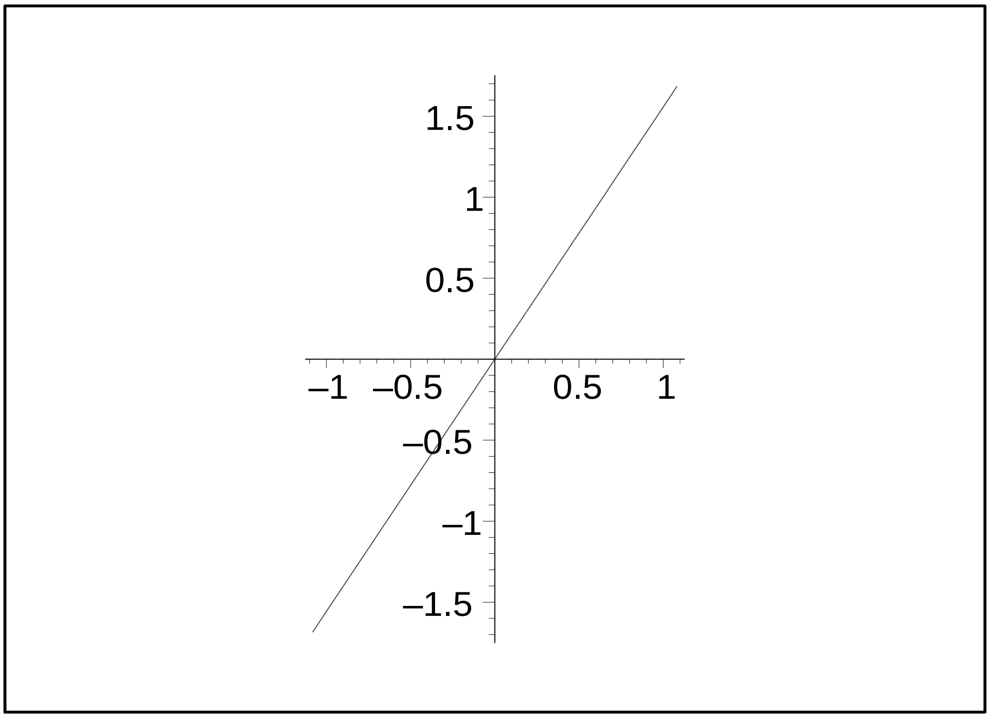
En Maple podemos pintar con el tipo de coordenadas que queramos. Por ejemplo para pintar en coordenadas polares, basta con que añadamos la opción **coords=polar** en el comando plot. Vamos a pintar los ejemplos vistos en clase:

```
> restart:
> plot(2,theta=0..2*Pi,coords=polar,scaling=constrained);
```

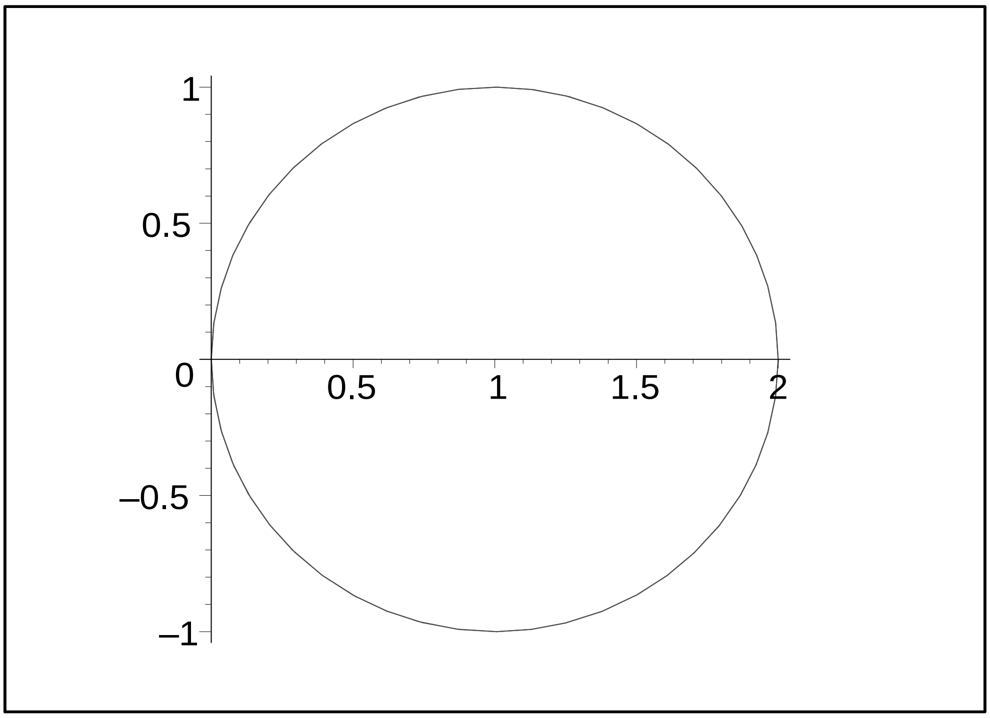



```
> with(plots):
Warning, the name changecoords has been redefined
```

```
> implicitplot(theta=1,r=-2..2,theta=1..2,coords=polar,scaling=constrai
> ned);
```



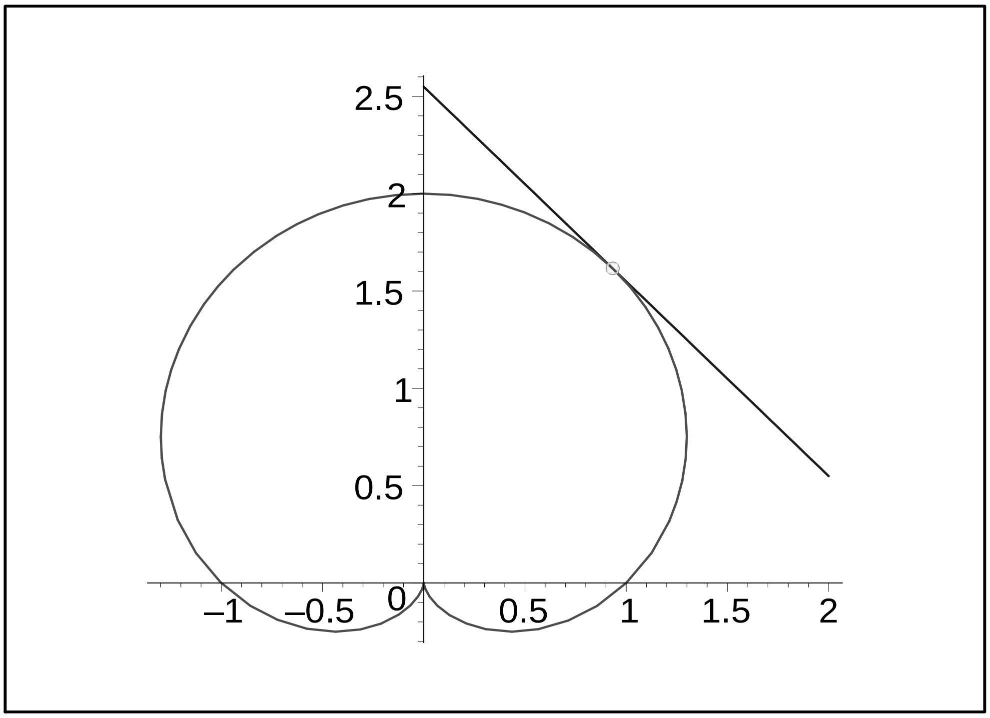

```
> plot(2*cos(theta),theta=0..2*Pi,coords=polar,scaling=constrained);
```



También podemos calcular la recta tangente en un punto.

Ejemplo: Calculamos la recta tangente a r=1+sin(a) en el punto a=Pi/3. Calculamos la pendiente de la curva en ese punto teniendo en cuenta que viene dada en polares y nos da que dy/dx=-1 (como vimos en clase), pintamos la recta tangente en cartesianas, para ello calculamos el punto en cartesianas y lo pintamos para comprobar que no nos hemos equivocado y finalmente pintamos la recta:

```
> d1:=plot(1+sin(theta),theta=0..2*Pi,coords=polar,scaling=constrained,
> thickness=3,color=red):
> d2:=plot(-x+(3*sqrt(3)+5)/4,x=0..2,coords=cartesian,color=blue,thickn
> ess=3):
> d3:=pointplot({[1+sqrt(3)/2,Pi/3]
> },coords=polar,symbol=box,symbolsize=16,color=yellow):
> d4:=pointplot({[(2+sqrt(3))/4,(3+2*sqrt(3))/4]
> },coords=cartesian,symbol=circle,symbolsize=22,color=green):
> display(d1,d2,d3,d4);
```



10.2 Ejercicio 1:

(a) Dibuja la curva en polares: r=2sen(a). Estudia a partir de que valores de **a** la curva se repite.

(b) Idem: r=cos(2a).

(c) Idem: r=sen(8a/5).

10.3 Ejercicio 2:

Estudia la familia de curvas en polares dada por r=1+csen(a) siendo c un número real cualquiera.

10.4 Ejercicio 3:

Encuentra y dibuja la recta tangente a la curva siguiente en el punto que se indica

(a) r=3cos(a), a=Pi/3

(b) r=cos(a)+sen(a), a=Pi/4

10.5 Coordenadas cilíndricas

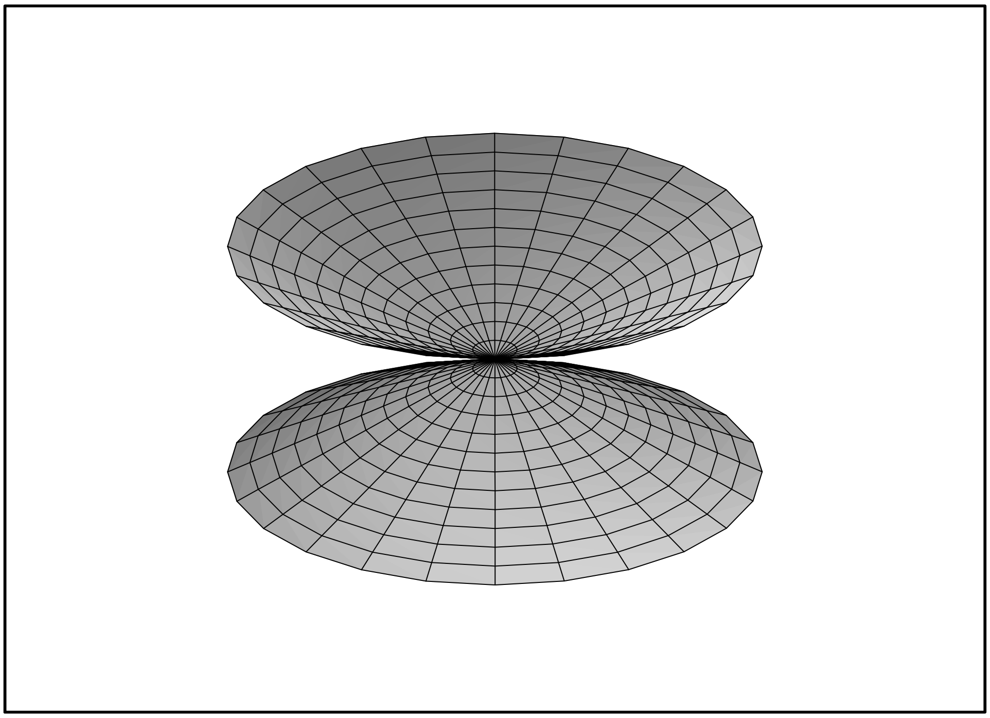
Para pintar en coordenadas cilíndricas, basta con que añadamos la opción **coords=cylindrical** en el comando plot. Vamos a pintar los ejemplos vistos en clase:

```
> restart:
```

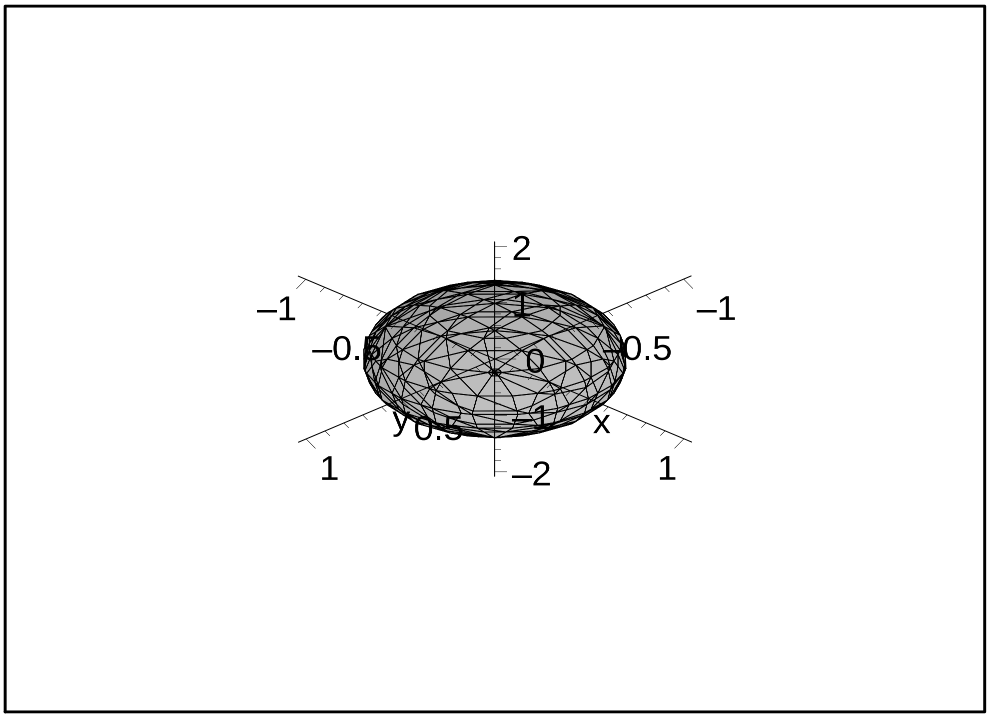
```
> with(plots):
```

```
Warning, the name changecoords has been redefined
```

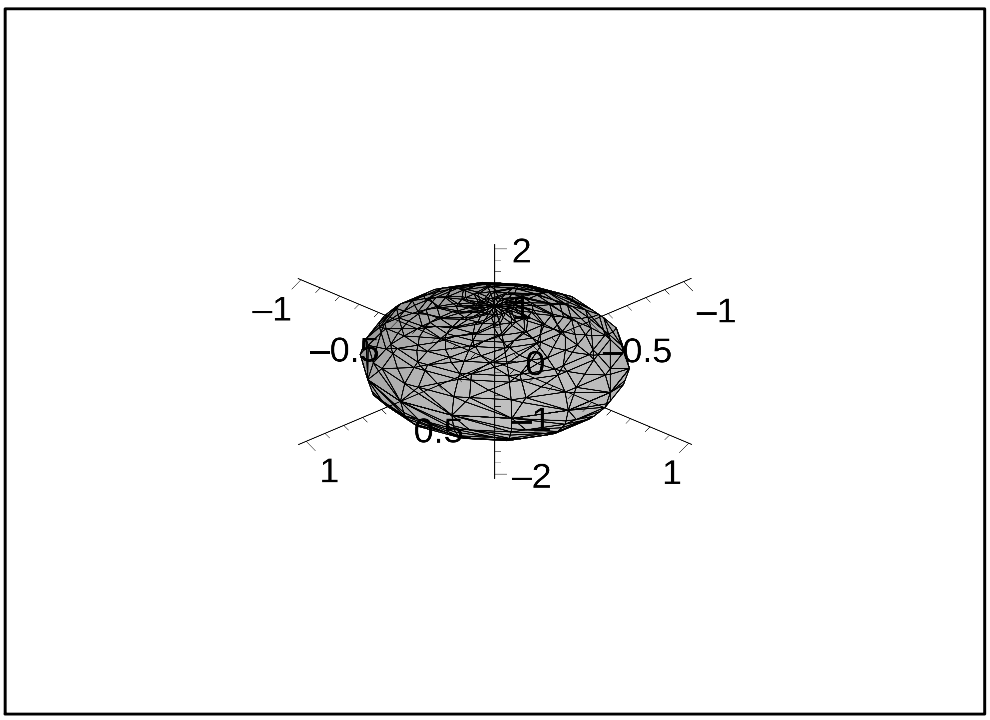
```
> plot3d(a,r=0..2*Pi,a=-2*Pi..2*Pi,coords=cylindrical);
```



```
> implicitplot3d(4*x^2 +4* y^2 + z^2 = 1,x=-1..1,y=-1..1,
> z=-2..2,axes=normal,grid=[15,15,15]);
```




```
> implicitplot3d(z^2=1-4*r^2,r=0..1,a=0..2*Pi,z=-2..2,coords=cylindrica
> l,axes=normal,grid=[15,15,15]);
```



10.6 Coordenadas esféricas

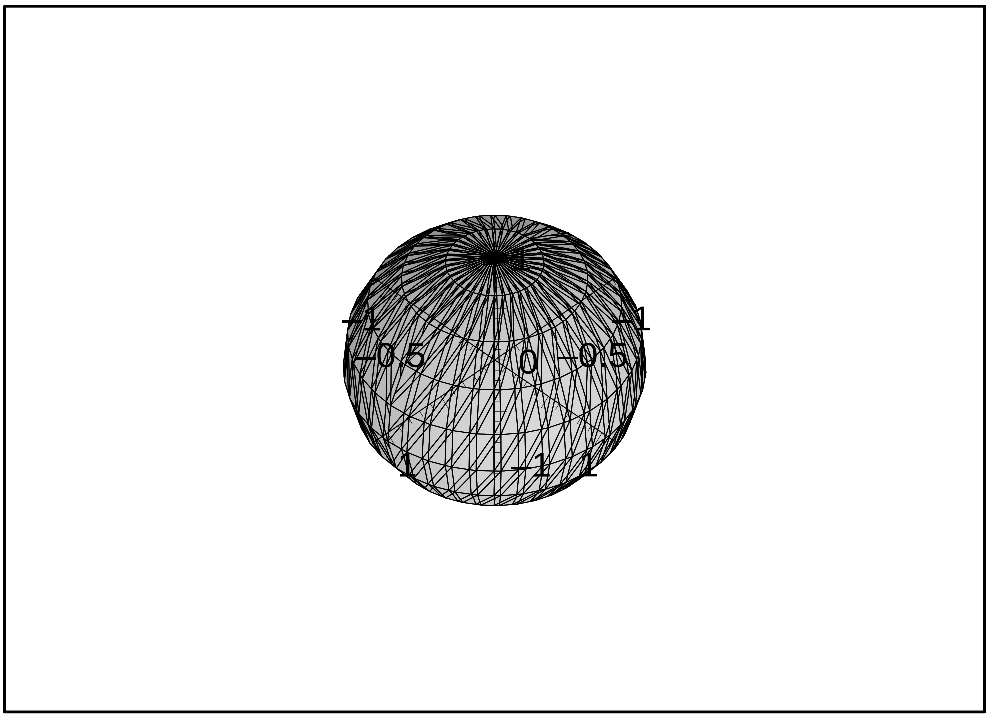
Para pintar en coordenadas esféricas, basta con que añadamos la opción **coords=spherical** en el comando plot. Vamos a pintar los ejemplos vistos en clase:

```
>  restart:
```

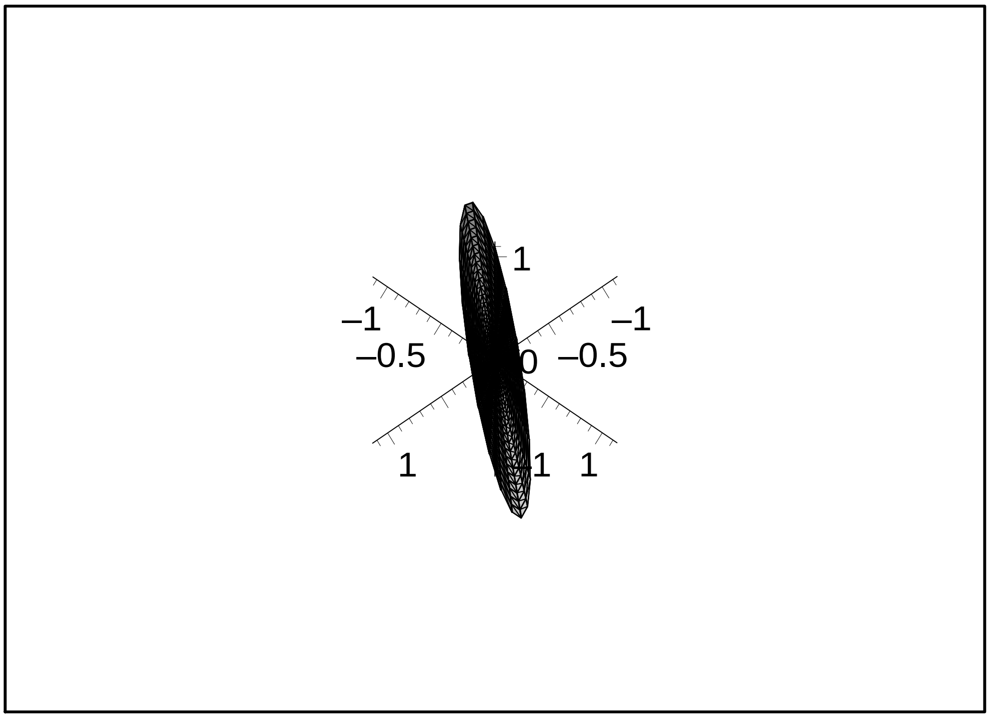
```
>  with(plots):
```

```
Warning, the name changecoords has been redefined
```

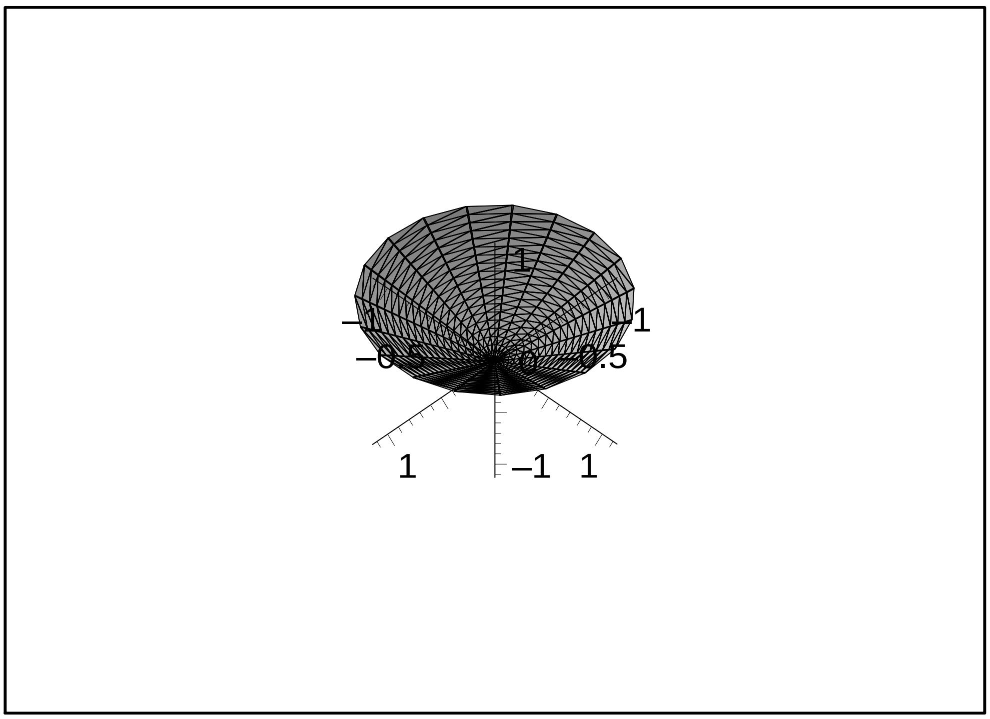
```
>  implicitplot3d(r=1,r=0..1.1,u=0..Pi,v=0..2*Pi,coords=spherical,axes=n
>  ormal,grid=[20,20,20],scaling=constrained);
```



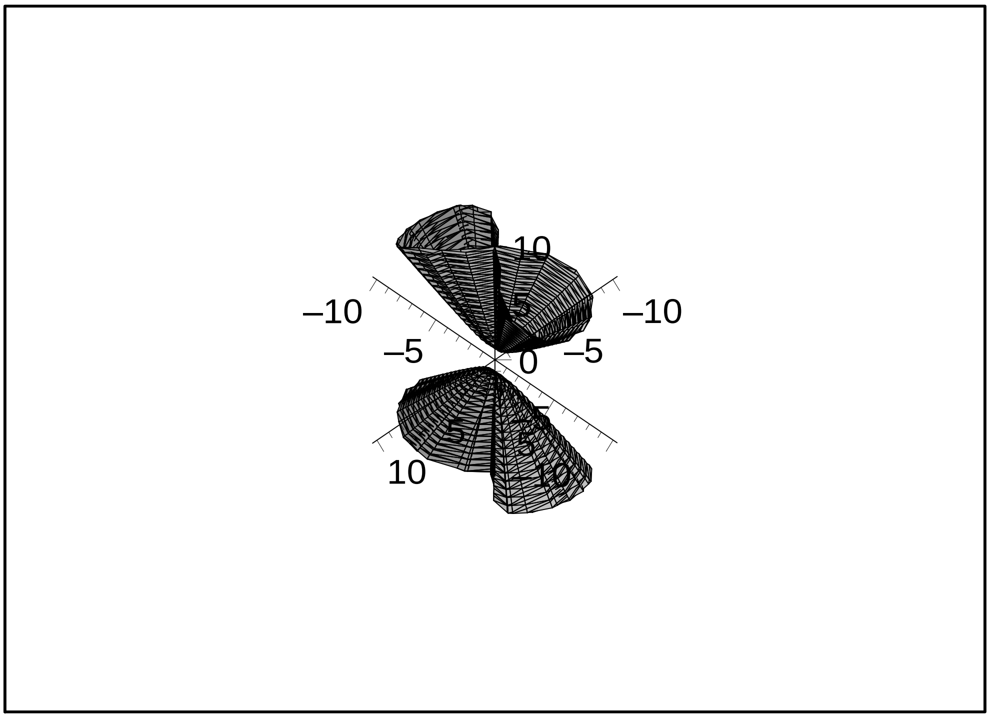
```
>  implicitplot3d(u=1,r=0..1.1,u=0..Pi,v=0..2*Pi,coords=spherical,axes=n
>  ormal,grid=[20,20,20],scaling=constrained);
```



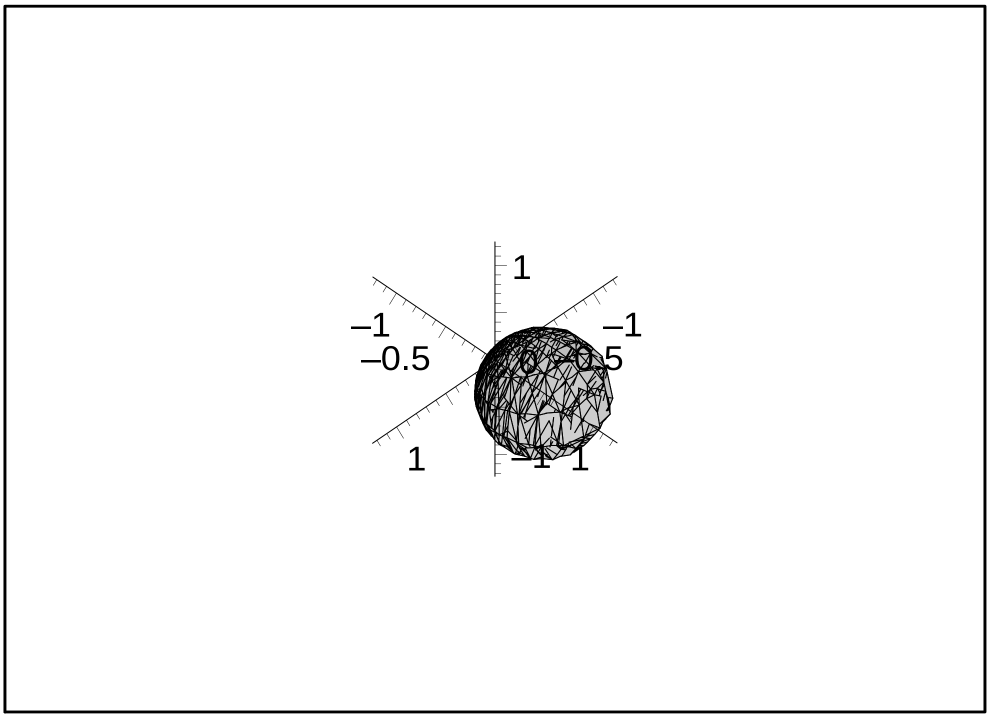

```
> implicitplot3d(v=1,r=0..1.1,u=-Pi..Pi,v=0..2*Pi,coords=spherical,axes
> =normal,grid=[20,20,20],scaling=constrained);
```




```
> implicitplot3d(r^2*(sin(u)^2*cos(2*v)-cos(u)^2)=1,r=0..10,u=-Pi..Pi,v
> =0..2*Pi,coords=spherical,axes=normal,grid=[20,20,20],scaling=constrai
> ned);
```




```
> implicitplot3d(r=sin(u)*sin(v),r=0..1.2,u=-Pi..Pi,v=0..2*Pi,coords=sp
> herical,axes=normal,grid=[20,20,20],scaling=constrained);
```



10.7 Ejercicios cilíndricas y esféricas

10.8 Ejercicio 4

Escribe y dibuja las siguientes ecuaciones rectangulares en cil índricas y esféricas

(a) $x^2+y^2+z^2=16$

(b) $x^2+y^2=2y$

(c) $x^2+y^2-z^2=16$

(d) $z=x^2-y^2$

10.9 Ejercicio 5

Dibuja el sólido que queda cuando se taladra un agujero de radio 3 a traves del centro de una esfera de radio 4.

10.10 Ejercicio 6

Dibuja un silo que conste de un cilíndro de radio 3 y altura 10 y este cerrado por un hemisferio.

Capítulo 11

Gráficas de funciones en el espacio.

11.1 Superficies en el espacio

```
> restart:
> with(geom3d):with(plots):with(linalg):with(plottools):
```

```
Warning, the name polar has been redefined

Warning, the name changecoords has been redefined

Warning, the name inverse has been redefined

Warning, the protected names norm and trace have been redefined and
unprotected

Warning, these names have been redefined: arrow, circle, dodecahedron,
hexahedron, homothety, icosahedron, line, octahedron, point, reflect,
rotate, sphere, stellate, tetrahedron, transform, translate
```

11.2 Ejercicio 1

Pinta las gráficas y las curvas de nivel de:

(a) z=sin(sqrt(x^2+y^2))

(b) z=x^2*y^2*exp(-x^2-y^2)

(c) z=1/(x^2+4*y^2)

(d) z=x^3-3*x*y^2

(e) z=sin(x)*sin(y)

(f) z=sin(x)^2+1/4*y^2

11.3 Curvas en el espacio. Movimiento en el espacio.

Si definimos una función **r** que va de R a R^3, de manera que para cada valor de **t** defina un punto en el espacio **(x(t), y(t),z(t))**, podemos pensar en ella como el movimiento de una partícula en el espacio. A veces a estas funciones le llamaremos **trayectoria.** Llamamos a r(t)=(x(t),y(t),z(t)) función **posición**, si derivamos con respecto al tiempo de manera similar a como hariamos con una función de una variable pero para cada componente, tendremos la función **velocidad** v(t)=r′(t)=(x′(t),y′(t),z′(t)) y la **rapidez** de la partícula en el tiempo t que es el módulo de la función velocidad, —v(t)— y la **aceleración**, la segunda derivada de la posición o la derivada de la velocidad, a(t)=v′(t)=r″(t)=(x″(t),y″(t),z″(t)).

Ejemplo: Determina la velocidad, la aceleración y la rapidez de una partícula con un vector de posición r(t)=(t^2,exp(t),t*exp(t))

```
> restart:
```

```
> with(geom3d):with(plots):with(linalg):with(plottools):
```

```
Warning, the name polar has been redefined

Warning, the name changecoords has been redefined

Warning, the name inverse has been redefined

Warning, the protected names norm and trace have been redefined and
unprotected

Warning, these names have been redefined: arrow, circle, dodecahedron,
hexahedron, homothety, icosahedron, line, octahedron, point, reflect,
rotate, sphere, stellate, tetrahedron, transform, translate
```

r(t)=(t^2,exp(t),t*exp(t)), posición

v(t)=(2*t,exp(t),(t+1)*exp(t)), velocidad

a(t)=(2,exp(t),(t+2)*exp(t)) aceleración, calculamos la rapidez

```
> rap:=sqrt((2*t)^2+(exp(t))^2+((t+1)*exp(t))^2);
```

$$rap := \sqrt{4\,t^2 + (e^t)^2 + (t+1)^2\,(e^t)^2}$$

Vamos a calcular la velocidad, aceleración y rapidez en t=1. Pintamos la curva y el vector tangente en t=1:

v(1)=(2,e,2*e);

a(1)=(2,e,3e)

y la rapidez en t=1 es:

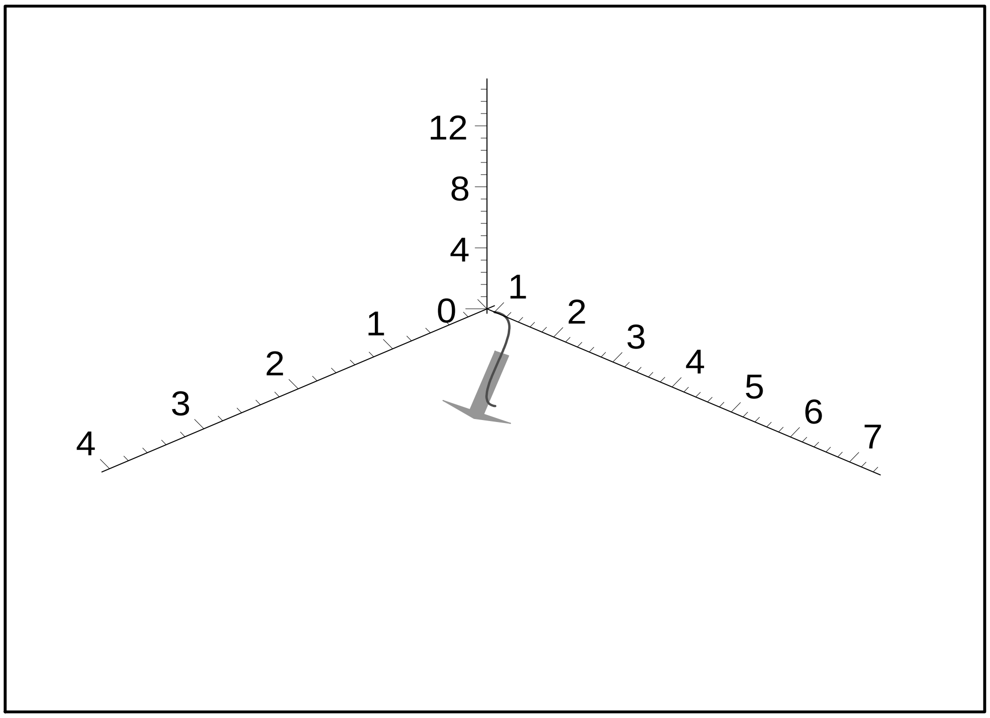
```
> evalf(subs(t=1.,rap));
```

$$6.398849934$$

```
> d2:=spacecurve([t^2,exp(t),t*exp(t)],t=0..2,color=red,thickness=3,axe
> s=NORMAL,numpoints=80):
```

```
> d3 := arrow([1,exp(1),exp(1)], [3,2*exp(1),3*exp(1)], .2, 1., .1,
> color=green):
```

```
> display(d2,d3);
```



11.4 Ejercicio 2

Encuentra la velocidad, la aceleración y la rapidez de una partícula con una función de posición dada. Pinta la curva y el vector tangente en cada punto indicado.:

(a) r(t)=(t^2-1,t,0), t=1

(b) r(t)=(sin(t),t,cos(t)), t=0

(c) r(t)=(t,t^2,t^3), t=1

11.5 Familias de superficies en el espacio

Vamos a estudiar funciones dependiendo del valor de uno o varios par ámetros. Vamos a estudiar cómo varíará la superficie que definen según los valores del parámetro.

11.6 Ejemplo

La familia de superficies vendrá dada por

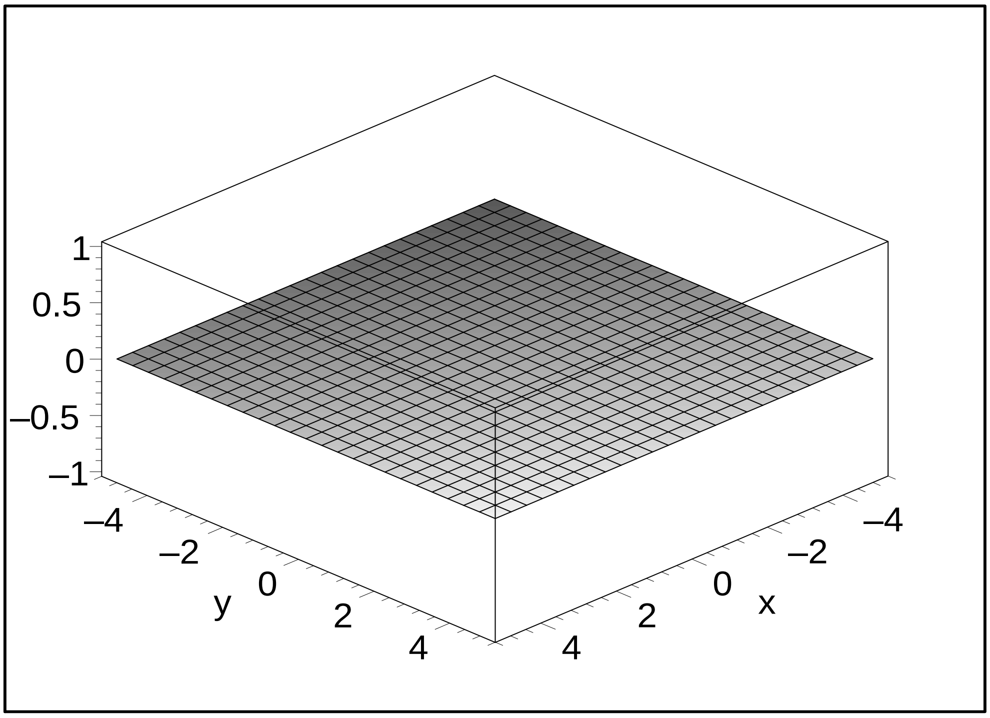
f(x,y)=(*a*x^2+*b*y^2)exp ^(-x^2-y^2)

```
> restart:
> with(plots):
Warning, the name changecoords has been redefined
```

```
> a:=0:
```

```
> b:=0:
```

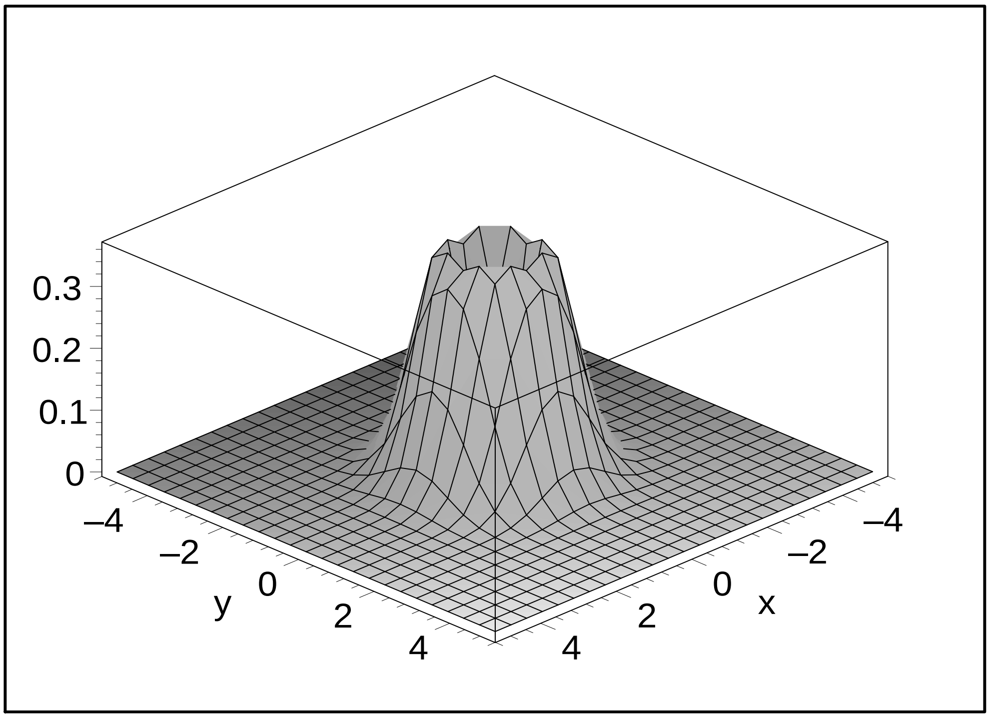
```
> plot3d((a*x^2+b*y^2)*exp(-x^2-y^2),x=-5..5,y=-5..5,axes=boxed);
```



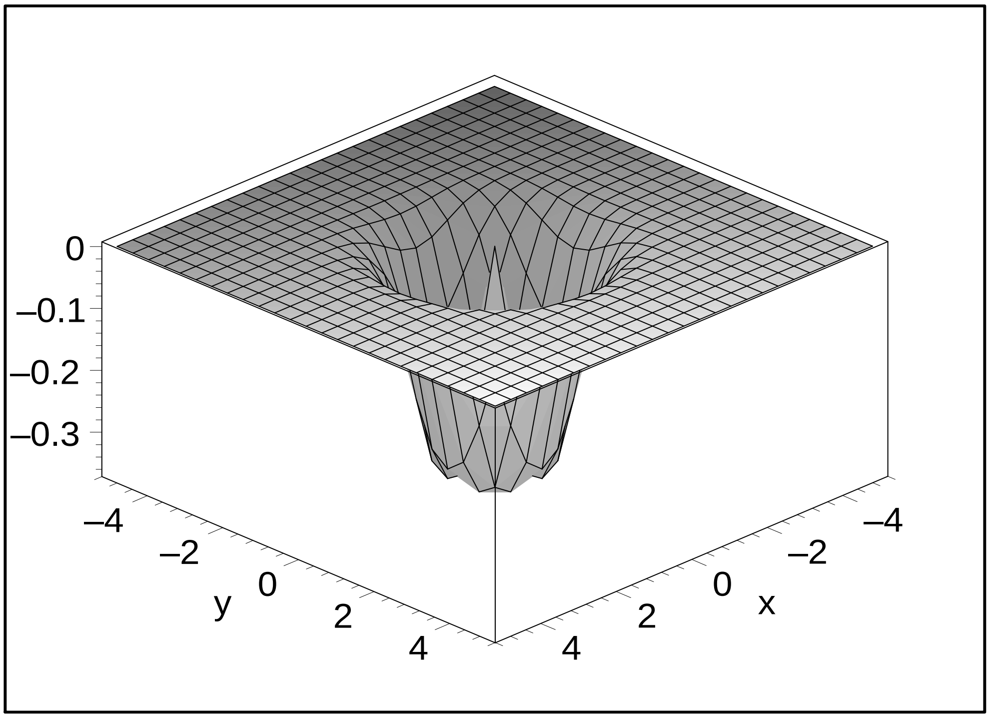

```
> a:=1:
```

```
> b:=1:
```

```
> plot3d((a*x^2+b*y^2)*exp(-x^2-y^2),x=-5..5,y=-5..5,axes=boxed);
```

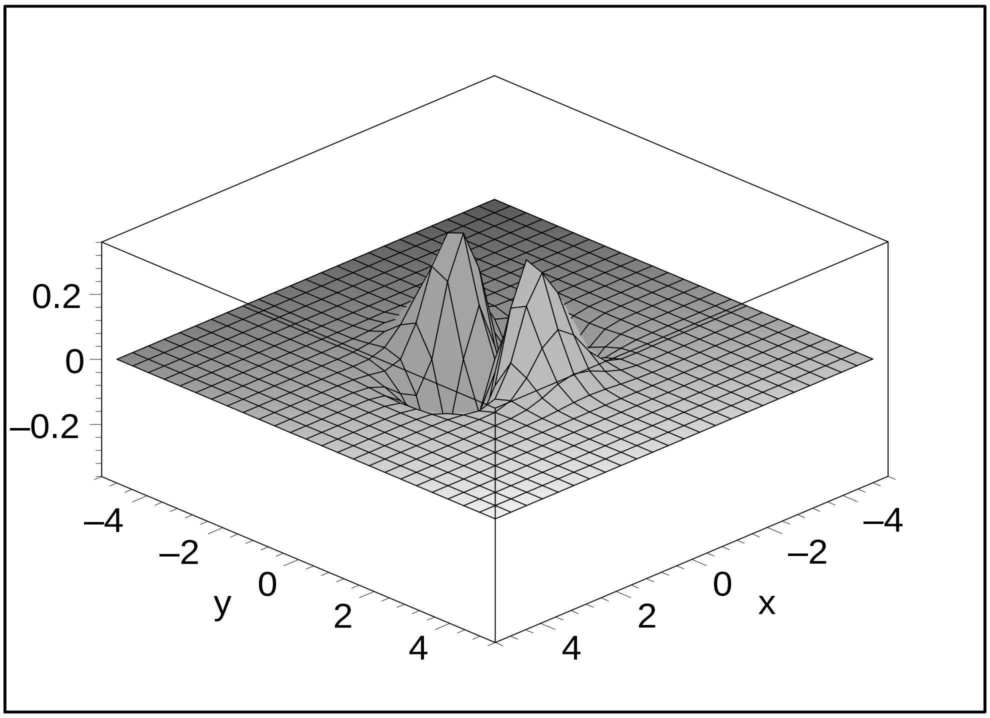



```
> a:=-1:
> b:=-1:
> plot3d((a*x^2+b*y^2)*exp(-x^2-y^2),x=-5..5,y=-5..5,axes=boxed);
```

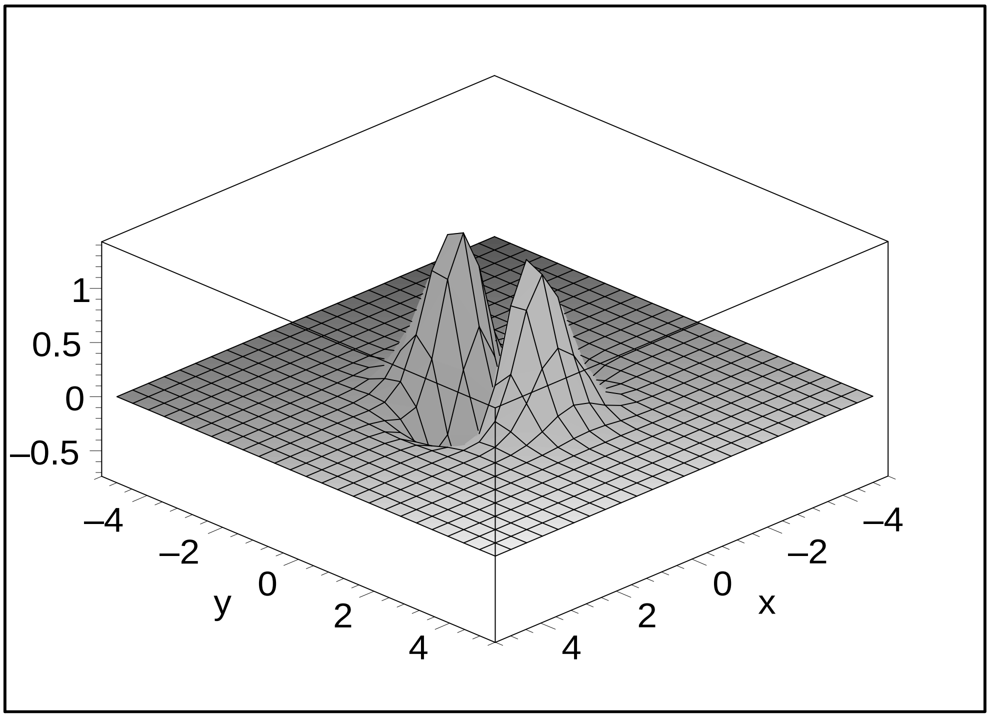



```
> a:=-1:
```

```
> b:=1:
> plot3d((a*x^2+b*y^2)*exp(-x^2-y^2),x=-5..5,y=-5..5,axes=boxed);
```




```
> a:=-2:
> b:=4:
> plot3d((a*x^2+b*y^2)*exp(-x^2-y^2),x=-5..5,y=-5..5,axes=boxed);
```



Se observa como la existencia o no de picos depende de que los parámetros a y b sean o no distintos de cero. El valor de un parámetro ya sea positivo o negativo da origen a un pico o un valle respectivamente. También la altura o profundidad viene dada por la magnitud en valor absoluto del parámetro, cuanto más grande más alto o profundo.

11.7 Ejercicios

Problema 3.

Analizar la familia de superficies

z=x^2+y^2+*cxy*

En particular, determine los valores de transición de c para los que la superficie cambia de un tipo de superficie a otro:

Problema 4.

Los miembros de la familia de las superficies, dadas en coordenadas esféricas, por la ecuación

R=1+0.2*sen(m*u)*sen(n*v)

se han sugerido como modelos para tumores y se han llamado *esferas irregulares o esferas fruncidas. Analiza esta familia de superficies, suponiendo que m y n son enteros positivos. ¿Qué papel desenpeñan los valores m y n en la configuración de la superficie?*

Capítulo 12

Máximos y mínimos: libres, condicionados y absolutos

12.1 Clasificación de puntos críticos. Gráficas. Extremos relativos.

Vamos a centrarnos en funciones escalares, esto es, funciones que van de R^n a R (En general n=2 o 3). Lo primero que tenemos que hacer es calcular el gradiente. La orden **grad** dentro del paquete **linalg** que calcula el gradiente de una función:

Cargamos el paquete **linalg**

```
> with(linalg):
```

```
Warning, the protected names norm and trace have been redefined and
unprotected
```

```
> G1:=grad(x^2+y^2,[x,y]);
```

$$G1 := [2\,x,\,2\,y]$$

Como el gradiente es un vector y lo que queremos es manejar cada coordenada, vemos como se hace en Maple para obtener cada coordenada, la primera:

```
> G1[1];
```

$$2\,x$$

La segunda:

```
> G1[2];
```

$$2\,y$$

Ejemplo de manipulación de coordenadas de vectores con Maple.

```
> v:=vector([3,5,2]);
```

$$v := [3,\,5,\,2]$$

```
> v[1];
```

$$3$$

```
> v[2];
```

$$5$$

```
> v[3];
```

$$2$$

Para calcular puntos críticos igualamos el gradiente a cero, cada componente, y resolvemos con solve por ejemplo, (siempre que exista el gradiente, si no habrá que estudiar los puntos de no diferenciabilidad incluso no continuidad aparte)

```
> solve({G1[1]=0,G1[2]=0},{x,y});
```

$$\{x = 0,\, y = 0\}$$

Para decidir si es máximo o mínimo o ninguno de los dos utilizamos e*l criterio de las segundas derivadas parciales*. Para calcular la matriz hessiana utilizamos el comando **hessian**, también del paquete **linalg**

```
> hessian(x^2+y^2,[x,y]);
```

$$\begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

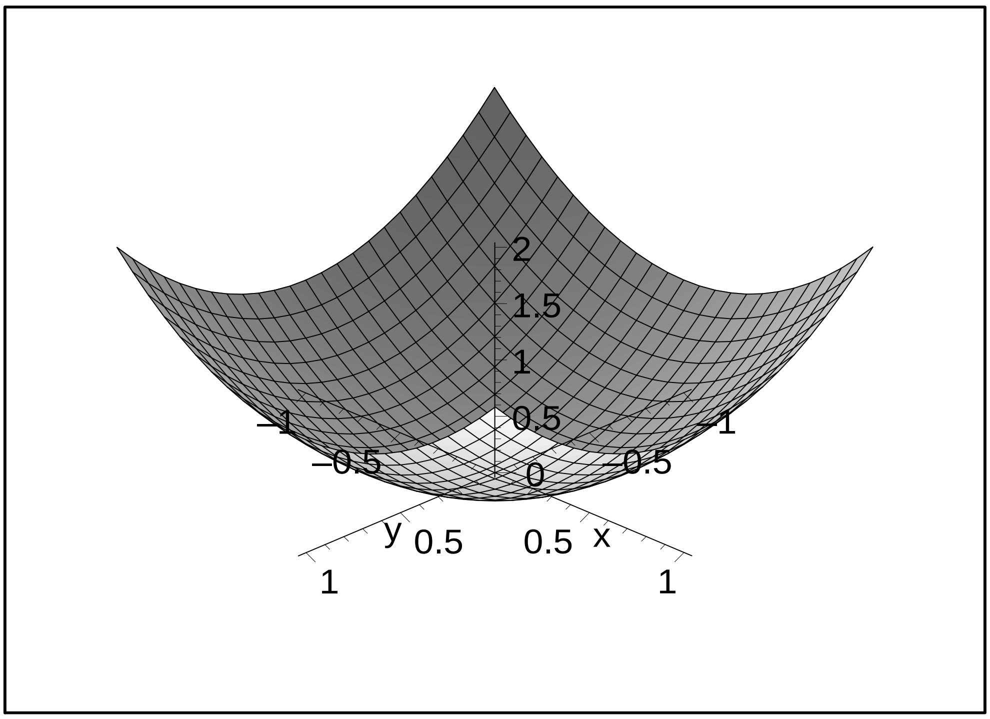
Aqui observamos que como no depende del punto y es definido positivo el punto (0,0) es un **mínimo** de nuestra función.

Suele ser interesante pintar la gráfica de la función entorno a los puntos críticos, esto vendrá especialmente bien cuando la función no tenga definido el gradiente o el criterio de la hessiana no nos de información. También pintarlas curvas de nivel puede ser interesante:

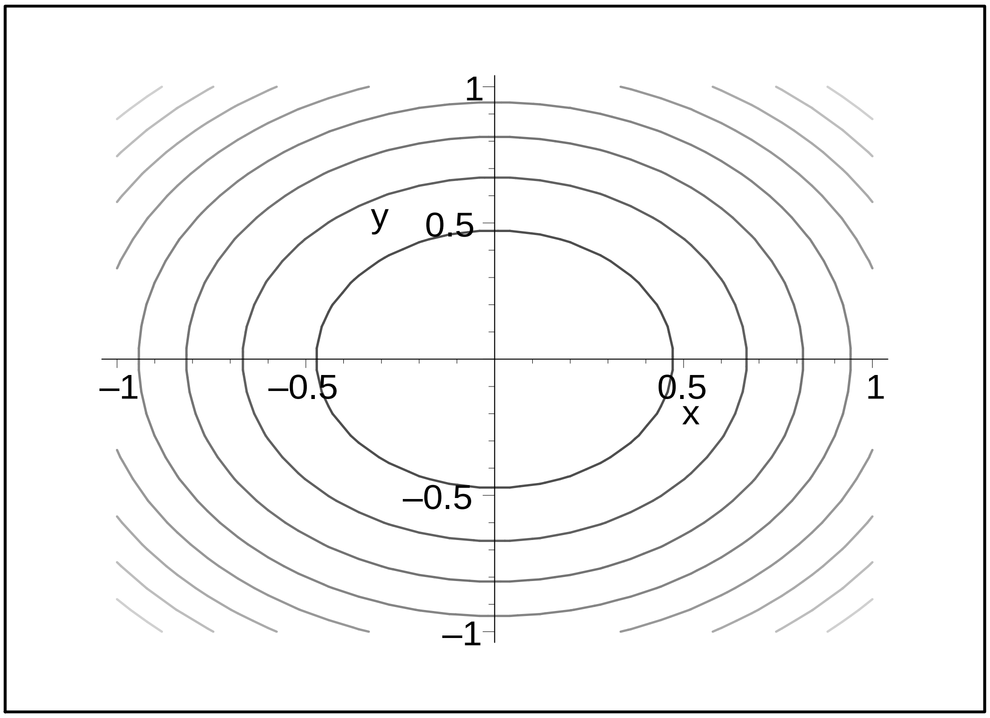
```
> with(plots):
```

```
Warning, the name changecoords has been redefined
```

```
> plot3d(x^2+y^2,x=-1..1,y=-1..1,axes=normal);
```




```
> contourplot(x^2+y^2,x=-1..1,y=-1..1,thickness=3);
```



Otro ejemplo: Calculamos y clasificamos los puntos críticos de x^2-y^2

```
> G2:=grad(x^2-y^2,[x,y]);
```

$$G2 := [2\,x,\ -2\,y]$$

```
> solve({G2[1]=0,G2[2]=0},{x,y});
```

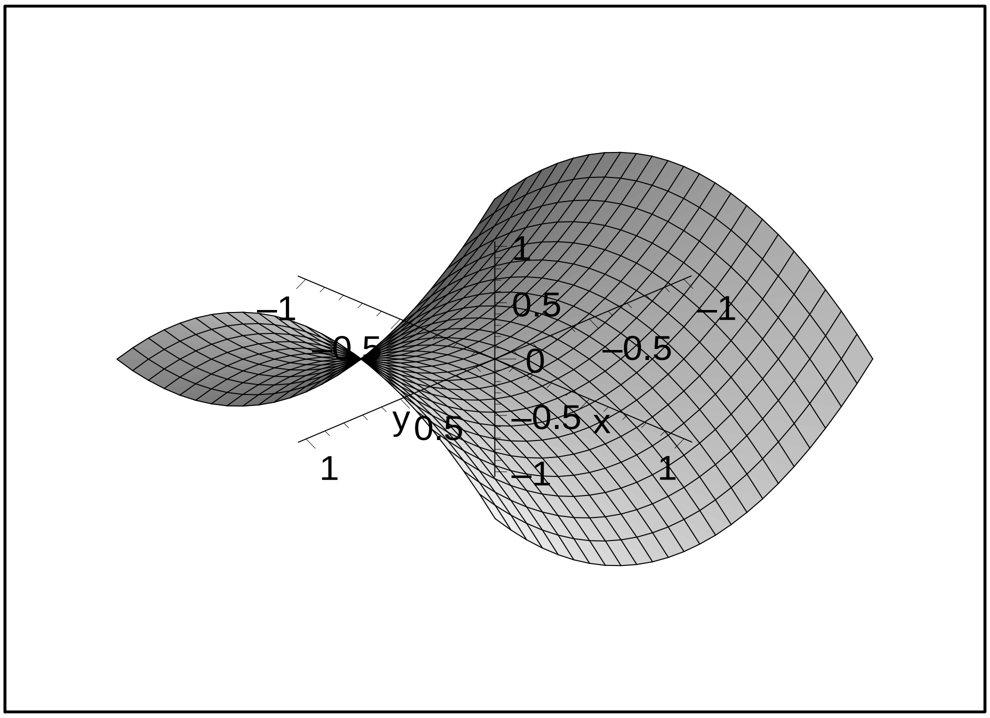
$$\{x = 0,\ y = 0\}$$

```
> hessian(x^2-y^2,[x,y]);
```

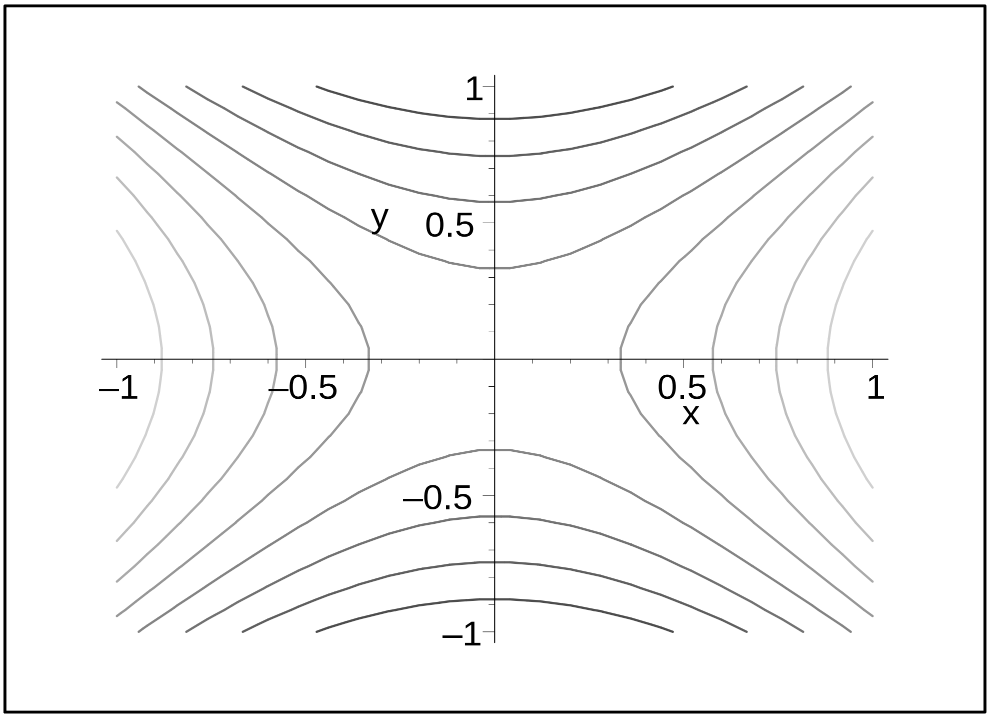
$$\begin{bmatrix} 2 & 0 \\ 0 & -2 \end{bmatrix}$$

Aqui el hessiano no es definido positivo ni negativo de manera que este criterio no me soluciona.
Pintamos la función y sus curvas de nivel

```
> plot3d(x^2-y^2,x=-1..1,y=-1..1,axes=normal);
```




```
>  contourplot(x^2-y^2,x=-1..1,y=-1..1,thickness=3);
```



Observando la gráfica y las curvas de nivel vemos que si nos acercamos por x=0, el origen podría ser un mínimo pero en cambio si nos acercamos por y=0 lo que tenemos es un máximo. De manera que en este caso lo que tenemos en el origen es un **punto silla.**

Un último ejemplo: Calculamos los puntos críticos de g(x,y)=x^2-3*x^4-3*x*y^2+2*x-5

```
> G3:=grad(x^2-2*x*y^2+2*x-5,[x,y]);
```

$$G3 := [2\,x - 2\,y^2 + 2,\ -4\,x\,y]$$

```
> solve({G3[1]=0,G3[2]=0},{x,y});
```

$$\{y = 0,\ x = -1\},\ \{x = 0,\ y = 1\},\ \{x = 0,\ y = -1\}$$

```
> hessian(x^2-2*x*y^2+2*x-5,[x,y]);
```

$$\begin{bmatrix} 2 & -4\,y \\ -4\,y & -4\,x \end{bmatrix}$$

Substituimos el valor en cada punto crítico.

```
> H1:=subs(x=-1,y=0,hessian(x^2-2*x*y^2+2*x-5,[x,y]) );
```

$$H1 := \begin{bmatrix} 2 & 0 \\ 0 & 4 \end{bmatrix}$$

```
> det(H1);
```

$$8$$

tenemos entonces 2 y 8, el hessiano es definido positivo así que tenemos un **mínimo**

```
> H2:=subs(x=0,y=1,hessian(x^2-2*x*y^2+2*x-5,[x,y]) );
```

$$H2 := \begin{bmatrix} 2 & -4 \\ -4 & 0 \end{bmatrix}$$

```
> det(H2);
```

$$-16$$

tenemos 2 y -16 asi que el hessiano es indefinido y por tanto tenemos un **punto silla**

```
> H3:=subs(x=0,y=-1,hessian(x^2-2*x*y^2+2*x-5,[x,y]) );
```

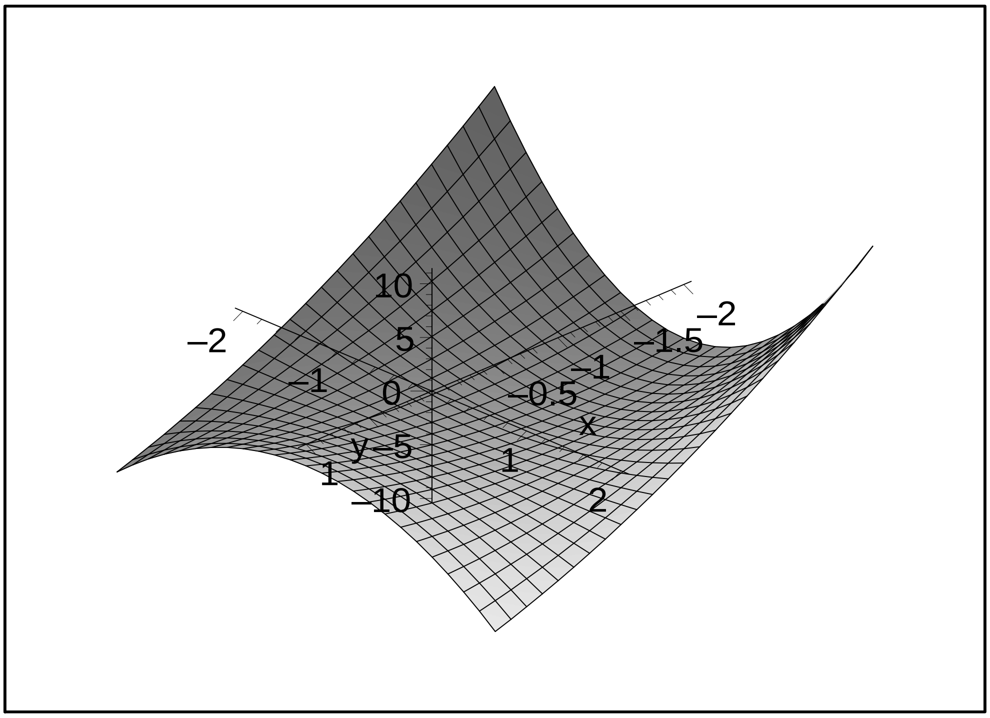
$$H3 := \begin{bmatrix} 2 & 4 \\ 4 & 0 \end{bmatrix}$$

```
> det(H3);
```

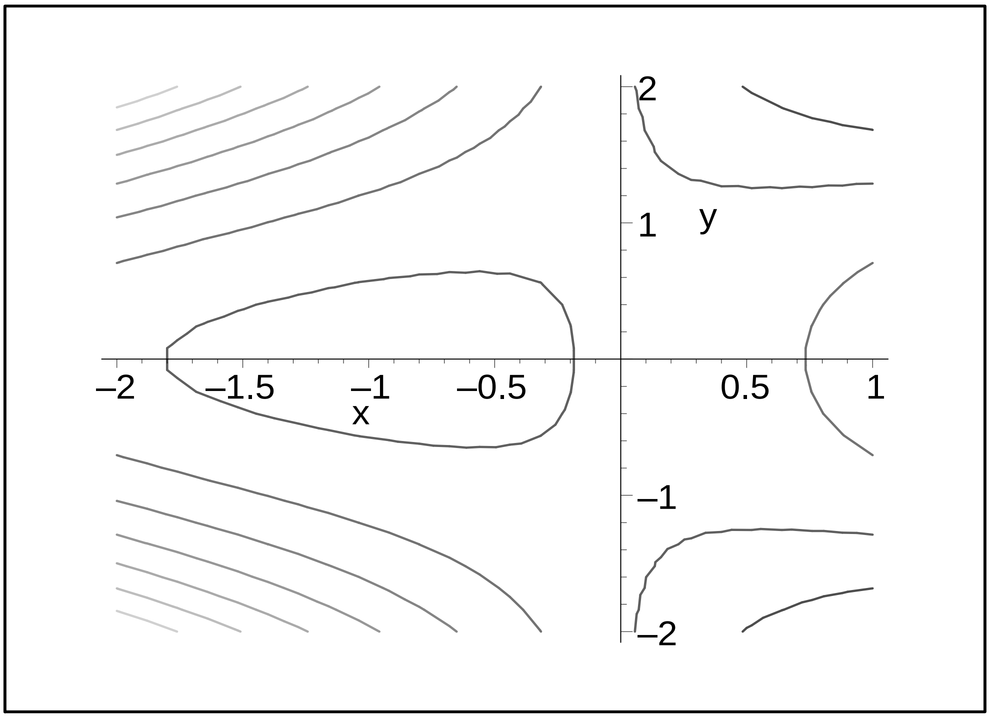
$$-16$$

También **punto silla**. Pintamos la función y sus curvas de nivel para comprobar.

```
> plot3d(x^2-2*x*y^2+2*x-5,x=-2..1,y=-2..2,axes=normal);
```




```
> contourplot(x^2-2*x*y^2+2*x-5,x=-2..1,y=-2..2,thickness=3);
```



12.2 Ejercicios

Calcula y clasifica los puntos críticos de las siguientes funciones, pinta la gráfica de la función y sus curvas de nivel.

(1) f(x,y)=x^2-sin(xy)

(2) f(x,y)=x+y^2-x^3y^2-3y

(3) f(x,y)=x+y^2-x^3y^2

(4) f(x,y)=2x^3-3xy+y^2

(5) f(x,y)=2x^3-3xy+6y^2-3y+2

12.3 Maximos y mínimos condicionados

Muchas veces la naturaleza del problema nos pone limítaciones a la zona donde debemos calcular los máximos y los mínimos. Si la función es continua y el conjunto sobre el que tenemos que calcular máximos y mínimos es un compacto (cerrado y acotado) sabemos seguro que en el conjunto habrá al menos un máximo y un mínimo.

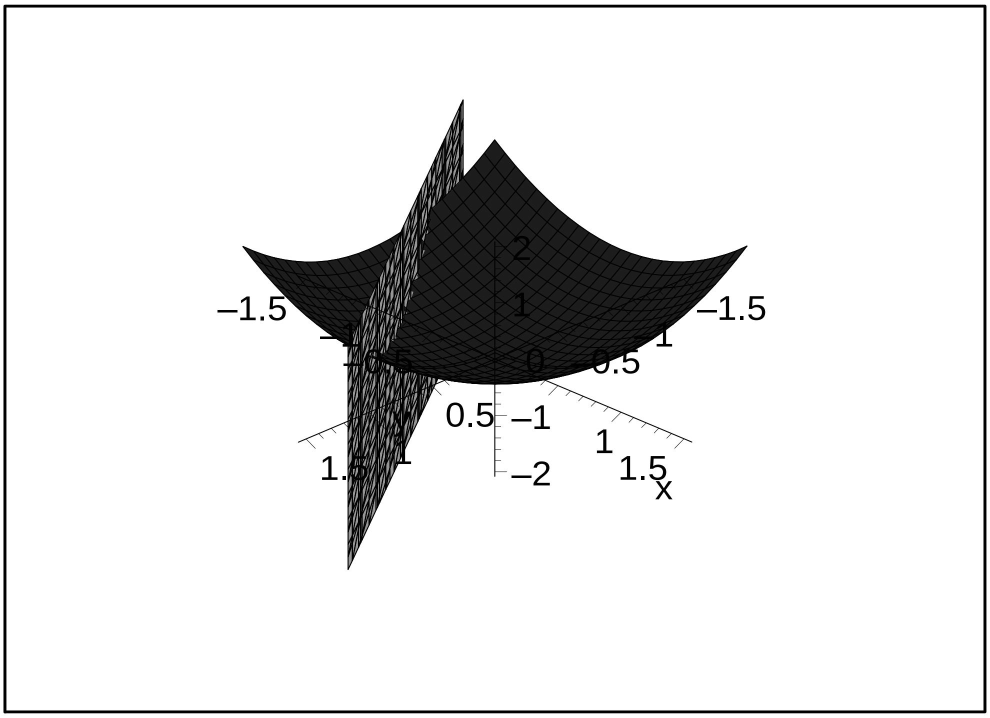
La manera de calcularlos consistirá en buscar los máximos y mínimos de la función sin restricciones y quedarnos de estos con lo que pertenezcan al conjunto y después estudiar la frontera del conjunto en donde dependiendo de su naturaleza utilizaremos unos métodos u otros: reducir a una variable menos, multiplicadores de Lagrange, estudio de las curvas de nivel, etc.

Como en el apartado anterior calculamos los extremos sin restricciones ahora nos centraremos en las restricciones, en los bordes.

12.4 Ejemplo 1

Calcular máximos y mínimos de la función x^2+y^2 sobre el plano 2*x+3*y=4

```
> restart:

> with(plots):

Warning, the name changecoords has been redefined

> d1:=plot3d(x^2+y^2,x=-1..1,y=-1..1,color=blue,axes=normal):

> d2:=implicitplot3d(2*x-3*y=2,x=-1.5..1.5,y=-1.5..1.5,z=-2..2,color=gr
> een,axes=normal):

> display(d1,d2);
```



Observamos que calcular los extremos de la función (en azul) sobre la restricción (en verde) es lo mismo que calcular extremos sobre la curva interesección.

```
> f:=(x,y)->x^2+y^2:
> solve(2*x+3*y=4,y);
```

$$-\frac{2\,x}{3}+\frac{4}{3}$$

```
> y:=-2/3*x+4/3:
> g:=x->simplify(f(x,y));
```

$$g := x \to \mathrm{simplify}(\mathrm{f}(x,\, y))$$

```
> g(x);
```

$$\frac{13}{9}\,x^2-\frac{16}{9}\,x+\frac{16}{9}$$

```
> diff(g(x),x);
```

$$\frac{26\,x}{9}-\frac{16}{9}$$

```
> solve(diff(g(x),x)=0,x);
```

$$\frac{8}{13}$$

```
> x:=8/13:
> y;
```

$$\frac{12}{13}$$

De manera que mi punto crítico es (8/13,12/13) y como la segunda derivada de g es positiva, lo que tenemos es un **mínimo**

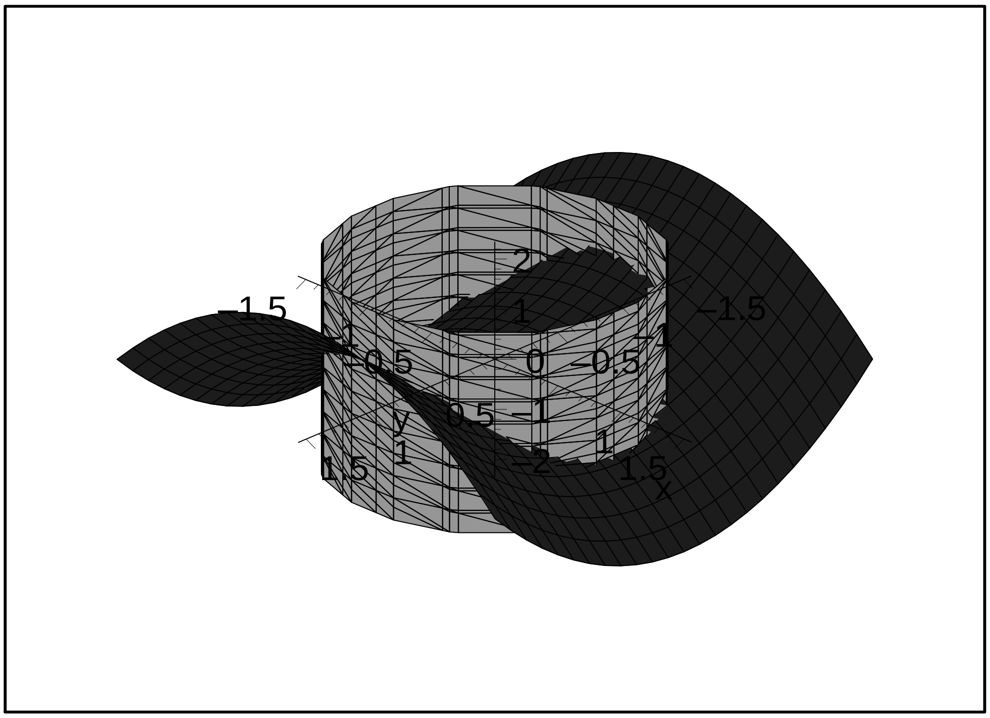
12.5 Ejemplo 2

```
> restart:
> with(plots):
```

```
Warning, the name changecoords has been redefined
```

Calcular los máximos y mínimos de f(x,y)=xˆ2-yˆ2 sobre la circunferencia unidad

```
> p1:=plot3d(x^2-y^2,x=-1.5..1.5,y=-1.5..1.5,color=blue,axes=normal):
> p2:=implicitplot3d(x^2+y^2=1,x=-1.5..1.5,y=-1.5..1.5,z=-2..2,color=gr
> een,axes=normal):
> display(p1,p2);
```



Tenemos que calcular máximos y mínimos de la función (en azul) cuando esta en la restricción (en verde). Es fácil observar que hay dos m áximos y dos mínimos.

Lo calculamos resudiendo a una variable. Las (x,y) que nos interesan tienen que cumplir xˆ2+yˆ2=1, De aquí despejamos la y:

```
> solve(x^2+y^2=1,y);
```

$$\sqrt{-x^2+1},\ -\sqrt{-x^2+1}$$

Sustituimos en la función:

```
> f:=(x,y)->x^2-y^2:
> y:=sqrt(-x^2+1):
> f(x,y);
```

$$2\,x^2-1$$

```
> diff(f(x,y),x);
```

$$4\,x$$

De manera que en (0,1),(0,-1) son puntos extremos.

```
> restart:
```

Si despejamos la x

```
> solve(x^2+y^2=1,x);
```

$$\sqrt{1-y^2},\ -\sqrt{1-y^2}$$

```
> f:=(x,y)->x^2-y^2:
> x:=sqrt(1-y^2):
> f(x,y);
```

$$1-2\,y^2$$

```
> diff(f(x,y),y);
```

$$-4\,y$$

De manera que (1,0) y (-1,0) también son extremos. Para saber si son m áximos o mínimos en este caso simplemente substituimos en la función.

```
> f(0,1);
```

$$-1$$

```
> f(0,-1);
```

$$-1$$

```
> f(1,0);
```

$$1$$

```
> f(-1,0);
```

$$1$$

Que son los que habiamos observado en la gráfica. Normalmente un problema de este tipo se resolveria más fácilmente por los multiplicadores de Lagrange que veremos en la próxima sección.

12.6 Ejercicios

(1) **Problema 3** del 31 de enero de 2003

(2) Analiza la existencia de máximos y mínimos de la función f(x,y)=(x+y)^2 con la condición xy=4

(3) Obten la distancia mínima del punto (5,-1) a los puntos de la par ábola y=x^2

12.7 Multiplicadores de Lagrange

El comando **extrema** proporciona los valores de los máximos y m ínimos relativos de la función alcanzados en los puntos que corresponde de entre esos puntos críticos que tiene gradiente nulo. En la última variable guardamos los valores donde se alcanzan los extremos. Esta orden esta ulitizando los multiplicadores de Lagrange.

```
> restart:
> with(plots):
Warning, the name changecoords has been redefined
```

```
> extrema(x^2-2*x*y^2+2*x-5,{},{x,y},sol1);
```

$$\{-6, -5\}$$

Para saber donde se alcanzan:

```
> allvalues(sol1);
```

$$\{\{x=-1, y=0\}, \{x=0, y=1\}, \{x=0, y=-1\}\}$$

Comprobamos los valores del máximo y mínimo

```
> f:=(x,y)->x^2-2*x*y^2+2*x-5;
```

$$f := (x, y) \to x^2 - 2\,x\,y^2 + 2\,x - 5$$

```
> f(-1,0);
```

$$-6$$

```
> f(0,-1);
```

$$-5$$

```
> f(0,1);
```

$$-5$$

El paréntesis vacío es necesario ponerlo porque esta orden también la vamos a utilizar cuando calculemos máximos y mínimos condicionados. **Por ejemplo** imaginemos que queremos calcular los extremos de la función anterior con la restricción x^2+y^2=1:

```
> extrema(x^2-2*x*y^2+2*x-5,{x^2+y^2=1},{x,y},sol2);
```

$$\{-6, -2\}$$

```
> allvalues(sol2);
```

$$\{\{x=-1, y=0\}, \{x=0, y=1\}, \{x=0, y=-1\}, \{y=0, x=1\}, \{x=\frac{-1}{3}, y=\frac{2\sqrt{2}}{3}\}\}, \{$$
$$\{x=-1, y=0\}, \{x=0, y=1\}, \{x=0, y=-1\}, \{y=0, x=1\}, \{x=\frac{-1}{3}, y=-\frac{2\sqrt{2}}{3}\}\}$$

```
> f(1,0);
```

$$-2$$

```
> f(-1/3.,(2/3)*sqrt(2));
```

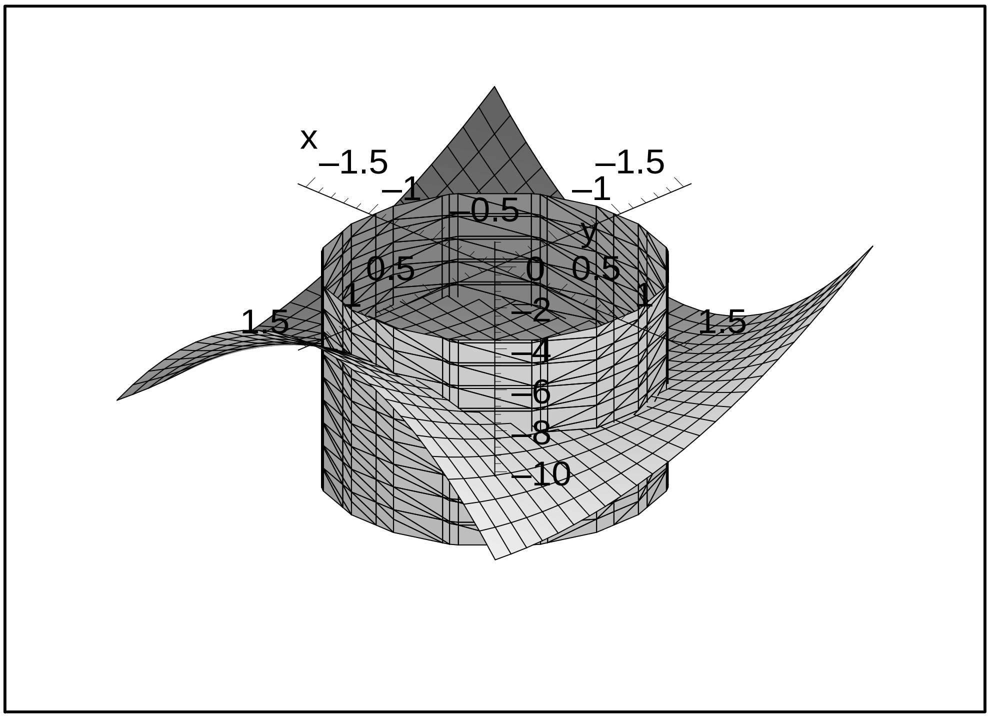
$$-4.962962963$$

```
> f(-1/3.,-2/3*sqrt(2));
```

$$-4.962962963$$

Vamos a pintar la curva donde hemos calculado los extremos:

```
> P1:=plot3d(x^2-2*x*y^2+2*x-5,x=-1.5..1.5,y=-1.5..1.5,axes=normal):
> P2:=implicitplot3d(x^2+y^2=1,x=-1.5..1.5,y=-1.5..1.5,z=-10..0,axes=no
> rmal):
> display(P1,P2);
```



Otro ejemplo: Sea T(x,y,z)=20+2*x+2*y*z^2 la temperatura en cada punto de la esfera x^2+y^2+z^2=11. calcular las temperaturas estremas a lo largo de la curva intersección de la esfera con el plano x+y+z=3.

```
> T:=(x,y,z)->20+2*x+2*y+z^2:
> extrema(T(x,y,z),{x^2+y^2+z^2=11,x+y+z=3},{x,y,z},sol3);
```

$$\{25, \frac{91}{3}\}$$

```
> allvalues(sol3);
```

$$\{\{y = 1 - \frac{2\sqrt{3}}{3},\, x = 1 - \frac{2\sqrt{3}}{3},\, z = 1 + \frac{4\sqrt{3}}{3}\},\, \{y = -1,\, z = 1,\, x = 3\},\, \{x = -1,\, z = 1,\, y = 3\}\}$$
$$,\{\{y = 1 + \frac{2\sqrt{3}}{3},\, z = 1 - \frac{4\sqrt{3}}{3},\, x = 1 + \frac{2\sqrt{3}}{3}\},\, \{y = -1,\, z = 1,\, x = 3\},$$
$$\{x = -1,\, z = 1,\, y = 3\}\}$$

```
> T(3,-1,1);
```

$$25$$

```
> T(-1,3,1);
```

$$25$$

```
> simplify(T(1-2/3*sqrt(3),1-2/3*sqrt(3),1+4/3*sqrt(3)));
```

$$\frac{91}{3}$$

```
> simplify(T(1+2/3*sqrt(3),1+2/3*sqrt(3),1-4/3*sqrt(3)));
```

$$\frac{91}{3}$$

Ejercicio: Resolver los ejemplos del apartado anterior utilizando los multiplicadores de Lagrange. Y utiliza la orden **extrema** para comprobar resultados.

12.8 Máximos y mínimos absolutos

Resolver los **ejercicios 77** y **80**

Bibliografía

[1] Apostol, T.M. Calculus 2a ed. Reverté. Barcelona, 1972

[2] Anton, H. Calculus: a new horizon. 6th ed. John Wiley & Sons. New York, 1999

[3] Bartle, R. and Sherbert, D. Introducción al Análisis matemático de una variable. Ed. Limusa. Mejico D. F. 1991.

[4] García et al. Calculo I. ????

[5] Larson et al. Algebra Lineal. Ed Pirámide. Madrid 2004

[6] Cohen, M. et al. Proof wihtout words: Geometrics Series. Mathematics Magazine. Vol 74. nº4 page 320, October 2001

[7] Pérez, C. Métodos matemáticos y programación con Maple V. 1a ed. ra-ma. Madrid, 1998.

[8] Roanes, E. et al. Cálculos matemáticos por ordenador con Maple V.5, 1a ed. Rubiños-1860, S.A. Madrid, 1999.

[9] Murray R. Spiegel, et al. Fórmulas y tablas de matemática aplicada. 2ª ed. Madrid 2000.

[10] Spivak, M. Calculus : cálculo infinitesimal 2a ed. Reverté. Barcelona, 1988.