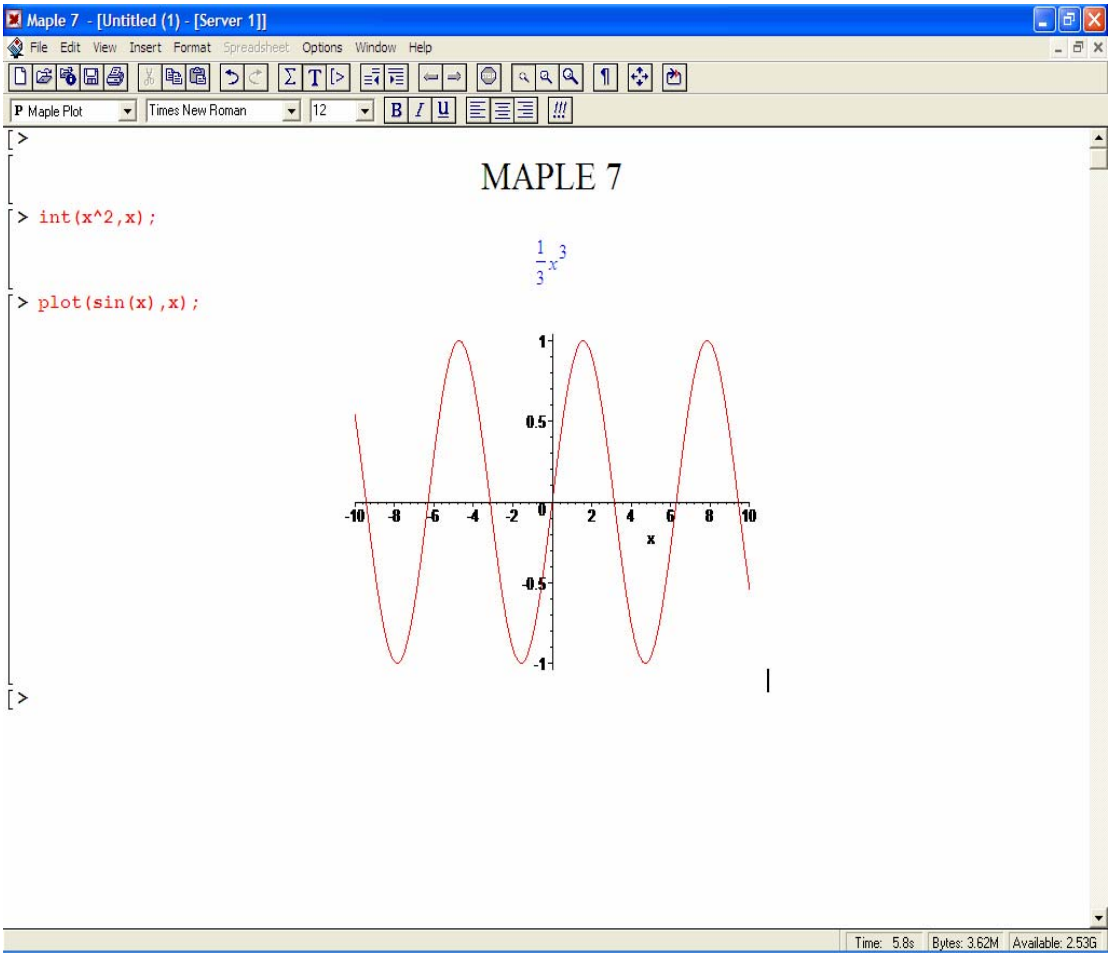
UNIVERSIDAD PONTIFICIA DE SALAMANCA

Ampliación de Matemáticas, Curso 2005/06

Preparado por: **Lic. Raúl Martín Martín**



UNIVERSIDAD PONTIFICIA DE SALAMANCA
Ampliación de Matemáticas, Curso 2005/06
Preparado por: **Lic. Raúl Martín Martín**

Práctica 1: Introducción a Maple

Maple es sistema de cálculo simbólico o *Computer Algebra System*, es decir, Maple puede manipular información de manera algebraica o simbólica. De este modo, podemos obtener soluciones analíticas exactas para diferentes problemas matemáticos: integrales, sistemas de ecuaciones, ecuaciones diferenciales, problemas de álgebra lineal… Además contiene rutinas para gráficas, algoritmos numéricos y un lenguaje de programación. Ingenieros y profesionales en la industria usan Maple como una gran herramienta sustituyendo a calculadoras, hojas de cálculo e incluso lenguajes de programación.

Matemáticas con Maple

El propósito de esta primera práctica consiste en aprender los primeros pasos con Maple, para ello, será muy útil que pruebes todos los ejemplos que se proponen según vayas leyendo.

Maple puede funcionar como una calculadora con enteros o números en punto flotante.
En pantalla aparece el símbolo “>” como una indicación de que maple está esperando nuestras instrucciones. Terminadas éstas (escritas con la sintaxis adecuada), debemos poner ; seguido de INTRO y entonces Maple responderá.
Puedes comenzar realizando algunas operaciones habituales (suma, producto, división, exponenciación):

```
> 2+3;
                                  5

> 4*6; 12/6; 2^5;
                                 24
                                  2
                                 32
```

Maple trabaja con cantidades exactas:

```
> 12/8;
```

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

```
> 12/8 * 2/3;
```

$$1$$

Maple permite hacer aproximaciones de un número, esto se hace así:

```
> evalf(11/43);
```

$$.2558139535$$

Generalmente las órdenes son expresiones en inglés, encerrando entre paréntesis los argumentos.

Para aclarar las dudas sobre los comandos, puedes usar:

? command y Maple te abre la pantalla de ayuda sobre el comando indicado.

Para una mejor aproximación de un número, indicamos entre corchetes [ ] el número de cifras decimales que queremos, así:

```
> evalf[50](11/43);
```

$$.25581395348837209302325581395348837209302325581395$$

La raíz cuadrada tiene orden propia:

```
> sqrt(12);
```

$$2\sqrt{3}$$

(De nuevo observa la forma exacta de operar de Maple).

El símbolo % hace referencia al último resultado y se usa a veces proque nos permite no tener que volver a escribir éste. Por ejemplo, obtengamos una aproximación con 60 decimales del resultado anterior:

```
> evalf[60](%);
```

$$3.464101615137754587054892683011744733885610507620761256111 62$$

¿Qué ocurre si volvemos a poner %? (Compruébalo)

Cuando Maple opera con decimales, da el resultado con decimales, aunque el número sea entero. Por eso, para obtener una aproximación decimal de 11/43 también podemos escribir 11.0/43 ó sin más 11./43 u 11/43..

Siempre que a algún número de los que intervienen en una operación le acompañemos de un punto, obtendremos el resultado aproximado:

```
> 11./43;
```

$$.2558139535$$

```
> 0.5*2;
```

$$1.0$$

```
> 11/43.;
```

$$.2558139535$$

```
> sqrt(3.);
```

$$1.732050808$$

Para conseguir saltos de línea, se pulsa la tecla de mayúsculas e INTRO.

Cálculo simbólico.-

Esta clase de programas pueden operar con símbolos y hacer simplificaciones como uno las haría en un papel. Por ejemplo:

Todos sabemos que $(x^2-1)/(x+1)=x-1$.

```
> (x^2-1)/(x+1);
```

$$\frac{x^2-1}{x+1}$$

```
> simplify(%);
```

$$x-1$$

Si manejamos expresiones más complicadas, puede que Maple no simplifique como lo haríamos nosotros.

No hay dificultad en usar dos o más variables:

```
> (a^2+a*b+b^2)*(a-b);
```

$$(a^2+a\,b+b^2)\,(a-b)$$

```
> simplify(%);
```

$$(a^2+a\,b+b^2)\,(a-b)$$

Para Maple la expresión anterior está simplificada. Observa que vamos buscando a^3-b^3, para ello usamos otras ordenes similares:

```
> collect(%,a);
```

$$a^3-b^3$$

Esta última orden resalta el valor de a en la expresión, poniendo esta como un polinomio en a. Otros ejemplos:

```
> (a-b)*(a^2-2)+(b-3)*(5-a);
```

$$(a-b)\,(a^2-2)+(b-3)\,(5-a)$$

```
> simplify(%);
```

$$a^3 + a - b\,a^2 + 7\,b - a\,b - 15$$

```
> collect((a-b)*(a^2-2)+(b-3)*(5-a),a);
```

$$a^3 - b\,a^2 + (1 - b)\,a + 7\,b - 15$$

```
> collect((a-b)*(a^2-2)+(b-3)*(5-a),b);
```

$$(-a^2 + 7 - a)\,b + a\,(a^2 - 2) - 15 + 3\,a$$

También podemos factorizar un polinomio:

```
> x^2+2*x-3;
```

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

```
> factor(%);
```

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

Asignación de valores a variables.-

Lo hacemos con la sintaxis: variable:=valor.

```
> a:=1;
(a-b)*(a^2-2)+(b-3)*(5-a);
```

$$a := 1$$

$$-13 + 5\,b$$

Para borrar la asignación:

```
> unassign('a');
expand((a+5)^2);
```

$$a^2 + 10\,a + 25$$

Se puede usar la sintaxis a:=’a’; como alternativa a unassign(‘a’);

```
> y:=4;
```

$$y := 4$$

```
> exp(y);
```

$$\mathbf{e}^4$$

La orden **restart;** borra el contenido de la memoria de Maple.

Con determinadas órdenes se puede conseguir que el valor que damos a una variable no quede asignado para siempre, sino sólo en lo que afecta a dicha orden. Es el caso de **simplify**. Por ejemplo, si queremos saber, de forma simplicada, cuánto vale el polinomio $3x^2+2x+1$ en el punto $x=(a-1)^2$, podríamos hacer:

```
> 3*x^2+2*x+1;
x:=(a-1)^2:
simplify(3*x^2+2*x+1);
```

$$3\,x^2+2\,x+1$$

$$3\,a^4-12\,a^3+20\,a^2-16\,a+6$$

El problema es que así x se ha quedado con el valor $(a-1)^2$, pues :

```
> 5*x+6;
```

$$5\,(a-1)^2+6$$

Sin embargo si hacemos:

```
> restart;
simplify(3*x^2+2*x+1,{x=(a-1)^2});
```

$$6+3\,a^4-12\,a^3+20\,a^2-16\,a$$

```
> 5*x+6;
```

$$5\,x+6$$

Otra orden que actúa de forma similar es **eval**.

```
> eval(y^3+2,y=1);
eval((y+2)/(y-2),y=sqrt(z));
```

$$3$$

$$\frac{\sqrt{z}+2}{\sqrt{z}-2}$$

```
> y;
```

$$y$$

Sucesiones, sumas y funciones.

Una sucesión es una función donde la variable es un número natural y, en muchos aspectos, Maple las trata de igual forma. Maple tiene predefinidas las funciones elementales; para usarlas solo hay que conocer la sintaxis apropada.

Ejemplos de funciones:

```
> restart;
> log(x);ln(x);#las dos expresiones son el logaritmo
neperiano
```

$$\ln(x)$$

$$\ln(x)$$

```
> arctan(z);
```

$$\arctan(z)$$

```
> sin(y);
```

$$\sin(y)$$

```
> exp(a^2+1);
```

$$\mathbf{e}^{(a^2+1)}$$

Podemos calcular valores particulares:

```
> exp(2);arctan(1);
```

$$\mathbf{e}^2$$

$$\frac{1}{4}\pi$$

```
> sin(Pi/4);log(exp(5));
```

$$\frac{1}{2}\sqrt{2}$$

$$5$$

Analicemos unos ejemplos de simplificación:

```
> log(exp(x));
```

$$\ln(\mathbf{e}^x)$$

```
> simplify(%);
```

$$\ln(\mathbf{e}^x)$$

El resultado no es x porque las funciones elementales, se extienden a variables compleja y ya sabemos que sus propiedades en el cuerpo de los números complejos son diferentes de las que tienen en el de los números reales. Vimos en teoría que la función exponencial compleja no es inyectiva ... Así:

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

$$I\,\pi$$

(es el complejo i = I). Digamos al programa que la variable es real:

```
> simplify(log(exp(x)), assume=real);
```

$$x$$

También puede hacerse:

```
> assume(x,real);
simplify(log(exp(x)));
```

$$x\sim$$

El resultado es el mismo. El significado de la tilde que acompaña a x es que Maple nos avisa de que sobre la variable x hay hecha una suposición y se mantendrá hasta que "desasignemos" dicha variable.

Maple conoce algunas relaciones entre funciones trigonométricas. Para simplificar, existen dos funciones específicas: **simplify** (expresión, **trig**) , y , **combine** (expresión, **trig**).

```
> restart;
> c:=sin(a)*cos(b)+cos(a)*sin(b);
```

$$c := \sin(a)\cos(b) + \cos(a)\sin(b)$$

```
> simplify(c,trig);
combine(c,trig);
```

$$\sin(a)\cos(b) + \cos(a)\sin(b)$$

$$\sin(a+b)$$

```
> d:=sin(u)^6;
simplify(d,trig);
combine(d,trig);
```

$$d := \sin(u)^6$$

$$1 - 3\cos(u)^2 + 3\cos(u)^4 - \cos(u)^6$$

$$\frac{5}{16} - \frac{1}{32}\cos(6u) + \frac{3}{16}\cos(4u) - \frac{15}{32}\cos(2u)$$

DEFINICIÓN DE FUNCIONES.-

La forma más sencilla de definir una función es la siguiente:

```
> f:=x->x^3+1;
```

$$f := x \to x^3 + 1$$

Hemos definido una función con el nombre **f** que, a cualquier cosa, le asigna su cubo más uno:

```
> f(2);f(a);f(b^(1/3));f(gato);
```

$$9$$

$$a^3 + 1$$

$$b + 1$$

$$gato^3 + 1$$

Podemos usar cualquier letra o conjunto de letras y números para designar una función. Si usamos alguna denominación que interfiere con alguno ya predefinido Maple nos avisa:

```
> sin:=x->x+2;
Error, attempting to assign to `sin` which is protected
```

También podemos definir funciones de más variables:

```
> suma:=(x,y)->x+y;
```

$$suma := (x, y) \rightarrow x + y$$

```
> suma(4,11);suma(suma,suma);
```

$$15$$

$$2\ suma$$

Podemos definir funciones por trozos usando la orden **piecewise.**

```
> g:=x->piecewise(x<0,sin(x),x<1,exp(x+1),7);
```

$$g := x \rightarrow \text{piecewise}(x < 0, \sin(x), x < 1, \mathbf{e}^{(x+1)}, 7)$$

```
> g(-1);g(1/2);g(3);
```

$$-\sin(1)$$

$$\mathbf{e}^{(3/2)}$$

$$7$$

```
> g(a);
```

$$\begin{cases} \sin(a) & a < 0 \\ \mathbf{e}^{(a+1)} & a < 1 \\ 7 & otherwise \end{cases}$$

Otra forma de definir una función es mediante un *procedimiento* (procedure en inglés o subrutina en lenguaje de programación). Por ejemplo, vamos a definir de otra manera la función suma.

```
> restart;
> f:=proc(x,y)
x+y;
> end proc;
```

$$f := \mathbf{proc}\,(x, y)\ x + y\ \mathbf{end\ proc}$$

Primero indicamos que queremos la letra **f** para indicar un procedimiento que va a afectar a dos variables. Después indicamos el procedimiento, en este caso la suma de las dos variables. Con **end proc** terminamos el procedimiento.

```
> f(2,5);f(silla,mesa);
```

$$7$$

$$silla + mesa$$

Sucesiones y sumas.

Sucesiones, sumas.

La sucesión 1, x, $\frac{x^2}{2}$, $\frac{x^3}{6}$, $\frac{x^4}{24}$, $\frac{x^5}{120}$ se obtiene escribiendo:

```
> seq(x^n/n!,n=0..5);
```

$$1, x, \frac{1}{2}x^2, \frac{1}{6}x^3, \frac{1}{24}x^4, \frac{1}{120}x^5$$

Y la suma de sus términos, se obtiene:

```
> sum(x^n/n!,n=0..5);
```

$$1 + x + \frac{1}{2}x^2 + \frac{1}{6}x^3 + \frac{1}{24}x^4 + \frac{1}{120}x^5$$

Análogamente $\frac{1}{2^2} + \frac{1}{2^4} + \frac{1}{2^6} + \ldots + \frac{1}{2^{20}}$ se obtiene mediante la orden:

```
> sum(1/2^(2*n),n=1..10);
```

$$\frac{349525}{1048576}$$

```
> evalf(%);
```

$$.3333330154$$

Las sucesiones se tratan como funciones. Estamos acostumbrados a usar la variable n para sucesiones y x para funciones. Para Maple las dos variables van a ser reales (o incluso complejas); por eso, en determinadas situaciones habrá que indicarlo. Por ejemplo:

```
> restart;
> f:=n->n^2+1;
```

$$f := n \to n^2 + 1$$

```
> frac(f(n));#frac indica la parte fraccionaria de un
número
```

$$\mathrm{frac}(n^2 + 1)$$

```
> assume(n,integer);frac(f(n));
```

$$0$$

Los procedimientos son IMPRESCINDIBLES para definir sucesiones que no vengan dadas por una expresión funcional. Como ejemplo hagamos un procedimiento para crear la sucesión **g(n) = n!** .

```
> g:=proc(n::posint)#estamos indicando que la variable sea
un número natural
g(1):=1;
g(n):=n*g(n-1);
> end proc;
```

$$g := \mathbf{proc}(n::posint)\ g(1) := 1;\ g(n) := n \times g(n-1)\ \mathbf{end\ proc}$$

```
> g(5);g(50);factorial(50);
```

120

30414093201713378043612608166064768844377641568960512000000000000

30414093201713378043612608166064768844377641568960512000000000000

Creamos la sucesión de Fibonacci, en la que sus dos primeros términos son 1 y cada uno es la suma de los dos anteriores: $f_n=f_{n-1}+f_{n-2}$, para $n \geq 2$.

```
> restart;
> fibo:=proc(n::posint)
fibo(1):=1; fibo(2):=1;
fibo(n):=fibo(n-2)+fibo(n-1);
end proc;
```

fibo := **proc**(*n*::*posint*)

fibo(1) := 1; fibo(2) := 1; fibo(*n*) := fibo(*n* – 2) + fibo(*n* – 1)

end proc

```
> fibo(7); fibo(100);
```

13

354224848179261915075

```
> seq(fibo(i), i=1..7);
```

1, 1, 2, 3, 5, 8, 13

Conjuntos y listas.-

Un conjunto es una colección de objetos (números, variables, funciones o lo que sea). Para definir un conjunto deben indicarse sus elementos entre llaves y separados por comas. El nombre de un conjunto es el que se quiera:

```
> restart;
> conjunto1:={3,0.5,x,sin(y)};
```

conjunto1 := {3, sin(*y*), .5, *x*}

```
> conjunto2:={0.5,asdf,exp(3),12};
```

conjunto2 := {12, .5, *asdf*, $\mathbf{e}^3$}

```
> conjunto3:={x,x,y,y,y,x};
```

conjunto3 := {*x*, *y*}

Con los conjuntos, podemos hacer operaciones:

```
> conjunto1 union conjunto2;
conjunto1 intersect conjunto2;
```

{3, 12, sin(*y*), .5, *x*, *asdf*, $\mathbf{e}^3$}

{.5}

Observa que los elementos de un conjunto no guardan ninguna ordenación. Por eso, se manejan **listas**, que son conjuntos ordenados.

```
> lista1:=[3,0.5,x,sin(y)];
```

$$lista1 := [3, .5, x, \sin(y)]$$

```
> lista3:=[x,x,y,y,y,x];
```

$$lista3 := [x, x, y, y, y, x]$$

Maple sabe que es una lista por la estructura **[.,.,.,.]** .
Para extraer los elementos de una lista:

```
> lista1[4];
```

$$\sin(y)$$

```
> lista3[2..4];
```

$$[x, y, y]$$

A su vez, los elementos de una lista también pueden ser listas

```
> L:=[1,3,[5,2],[3,4,[2,7]]];
```

$$L := [1, 3, [5, 2], [3, 4, [2, 7]]]$$

```
> L[3];
```

$$[5, 2]$$

```
> L[3,1];
```

$$5$$

```
> L[4,3,2];
```

$$7$$

Podemos operar con los elementos de la lista uno a uno:

```
> lista3/lista3;
lista3 + 2*lista3;
```

$$1$$

$$[3\,x, 3\,x, 3\,y, 3\,y, 3\,y, 3\,x]$$

Programación.-

Maple dispone de los clásicos bucles **for, while, do,** asi como de **if, else** etc...

Los procedimientos (procedures) son programas construidos por el usuario con un nombre asignado de forma que funcionen como una orden de Maple.
La estructura, engeneral, será del tipo siguiente:

```
Nombre = proc(x1::t1,x2::t2,...xn::tn)
             local v1,v2,....vr;  #comentario 1
             global g1,g2,...gs;  #comentario 2
             options o1,o2,..ot;  #comentario 3
             description frase breve o cadena de caracteres
                       que describe el procedimiento ;
             aquí aparecerá la sucesión de órdenes o comandos
             y sentencias de Maple que constituyen el cuerpo
             del procedimiento
          end:
```

lo que corresponde a

Nombre del procedimiento = **proc** (argumentos o variables especificando el tipo si es preciso)

sucesión de nombres de las variables locales;
sucesión de nombres de variables globales;
opciones diversas;
descripción del procedimiento;
cuerpo del procedimiento
comando **end** (seguido de dos puntos para que no produzca una salida que seria todo el listado)

Definamos una función que llamaremos **numdiv**, que nos dé el número de divisores de un número natural:

```
> restart;
> numdiv:=proc(n::nonnegint)
local d,k;
d:=0;
for k from 1 to n do
if floor(n/k)=n/k then
d:=d+1;
end if;
end do;
print(d);
end proc:

> numdiv(20);
```


Variables locales y globales.

Veremos un ejemplo para explicar la diferencia entre variables globales y locales. Construiremos un procedimiento, que llamaremos P1, para encontrar el primer número entero cuyo factorial sea mayor que 10^{100} en el que incluso no figurará ningún argumento o variable de entrada. Primero definiremos la variable x asignándole el valor 2 en el modo interactivo ordinario de los comandos de Maple, con lo que, como sabemos, mantendrá ese valor mientras no la “desasignemos” expresamente. Esto es lo que llamaremos una variable global. A continuación escribiremos el citado procedimiento, incluyendo en el una variable con nombre x pero declarándola variable local. A lo largo del programa será asignada con diferentes valores y dentro del programa eso es lo que valdrá. Eso es lo que significa que dentro del procedimiento la variable x ha sido local. La última asignación válida dentro del programa la daremos como salida, pero una vez ejecutado el programa, tras la salida del mismo, el valor de x seguira siendo 2 hasta que sea expresamente “desasignado”.

```
> x:=2;
> P1:=proc()
    local x;
    x:=10: #se inicializa x
    while x!<10^100 do x:=x+1 od: #bucle principal
    x;  #se da como salida el ultimo valor de x
  end:
> P1();
> x;
```

EJERCICIO 1.-

Define una función **div**, que nos devuelva el conjunto de divisores de un número natural.

Paquetes (Packages)

El uso de algunas herramientas, requiere cargar ciertos paquetes, por ejemplo:

- LinearAlgebra para el álgebra lineal
- DEtool para las ecuaciones diferenciales ordinarias
- stats para estadística
- plots para algunos gráficos
- …

La forma de cargar un paquete es: **with(nombre del paquete)** y después punto y coma o dos puntos si no se quiere ver el contenido del paquete.

Por ejemplo carga el paquete para el álgebra lineal y escribe punto y coma al finalizar. Haz lo mismo escribiendo dos puntos.

Gráfica de funciones.-

Consideremos la siguiente expresión como una función de x:

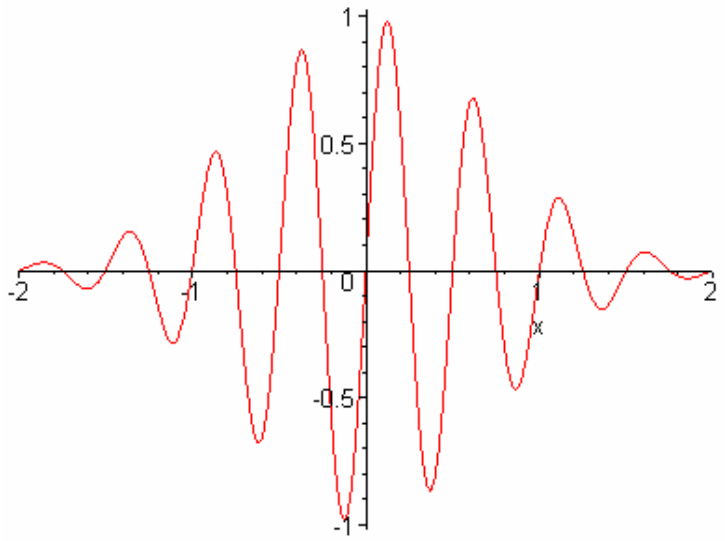
```
> F:=exp(-x^2)*sin(4*Pi*x);
```

$$F := \mathbf{e}^{(-x^2)} \sin(4\,\pi\,x)$$

Hacemos su gráfica en el intervalo [-2 .. 2]

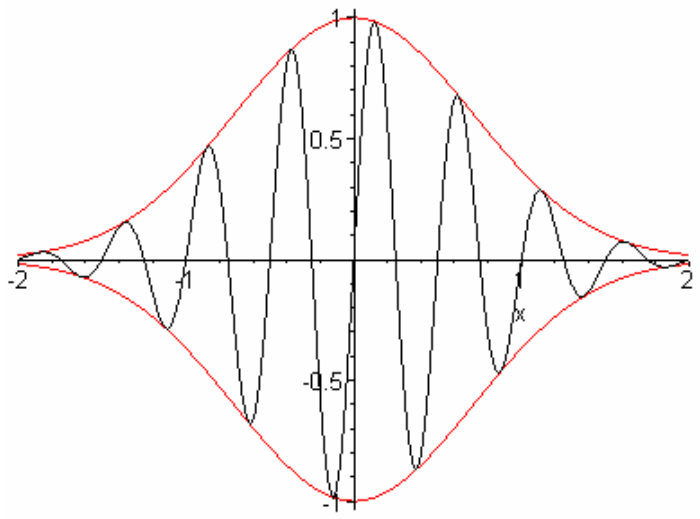
```
> plot(F, x=-2..2);
```

>



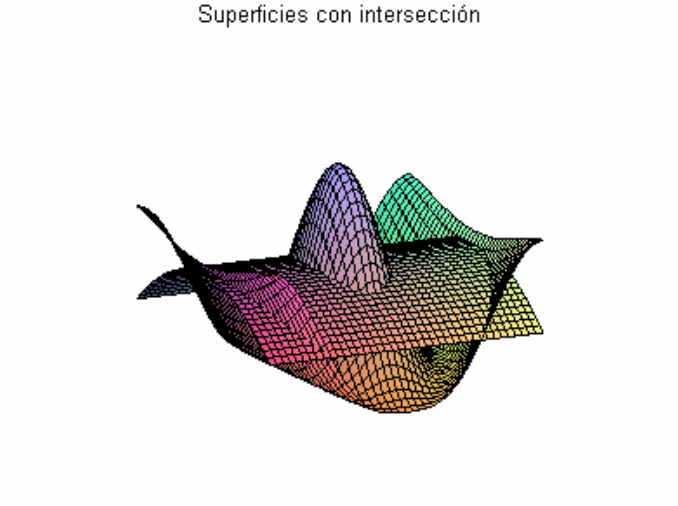
Hacemos ahora el gráfico de F junto con sus envolturas.

```
> plot([F, exp(-x^2), -exp(-x^2)], x=-2..2,
color=[black,red,red]);
```



Hacemos el gráfico de dos superficies que intersecan. Observa que usamos el comando *plot3d* en el caso de gráfico en 3 dimensiones y plot en caso de 2 dimensiones:

```
> plot3d({cos(sqrt(x^2+3*y^2))/(1+y^2/8), 2/15-
(2*x^2+x^2)/50},
x=-3..3, y=-3..3, grid=[41,41], orientation=[-26,71],
title="Superficies con intersección");
```

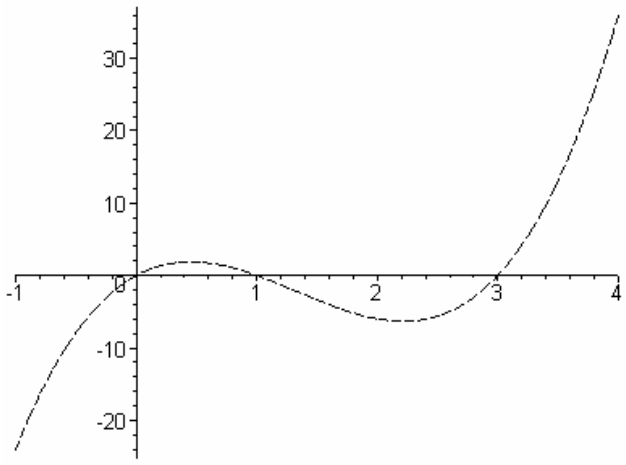


Gráfica de funciones escalares.-
Debe usarse el comando plot, en el cual indicamos la función a representar, el intervalo que se va a dibujar y el estilo con el que deseamos que se trace la línea sobre el gráfico (en color, en línea de trazos, etc...).

```
> g:=x->3*x^3-12*x^2+9*x;
```

$$g := x \to 3x^3 - 12x^2 + 9x$$

```
> plot(g,-1..4,color=black,linestyle=3);
```

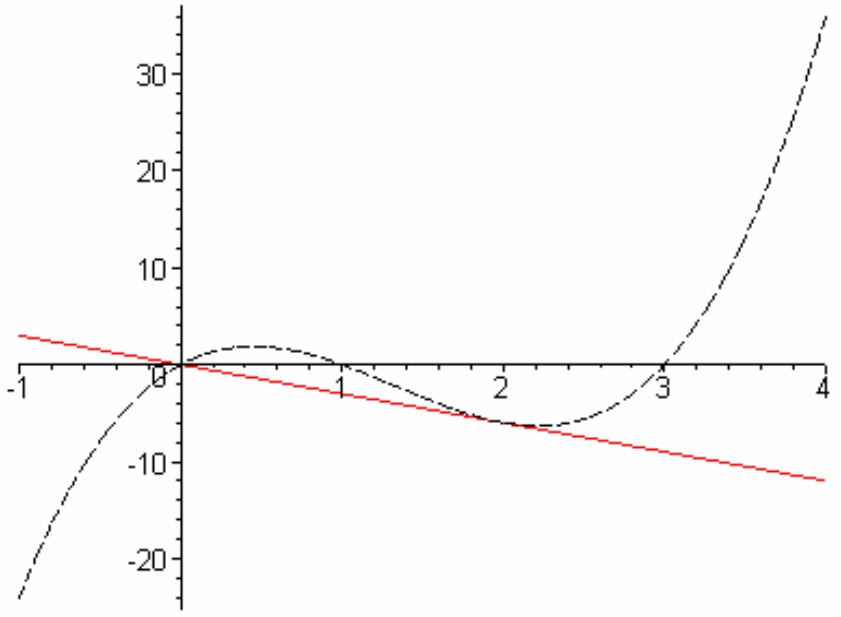


Para representar la recta tangente a la curva g por un punto x_0, recordemos su ecuación: $y(x)=g(x_0)+g'(x_0)(x-x_0)$.
Usamos el comando **D** para derivar y calcular la tangente en el punto $x_0=2$:

```
> y:=x->g(2)+D(g)(2)*(x-2);
```

$$y := x \to g(2) + D(g)(2)\,(x-2)$$

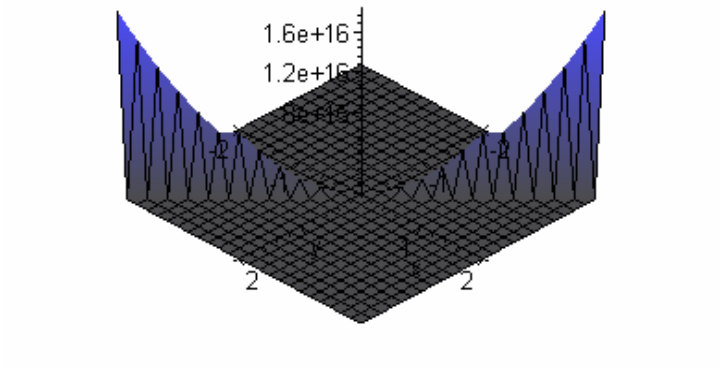
```
> plot([g,y],-1..4,color=[black,red],linestyle=[3,1]);
```



Gráficas de campos escalares

Para hacer el gráfico del campo escalar: $f(x,y)=(x^2+y^2)/(x+y)$ en la región $|x|\leq 2$, $|y|\leq 2$, usamos el comando **plot3d** con la opción axes=normal para ver los ejes:

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



Escribe ?plot para más información.

Ecuaciones

Las ecuaciones pueden ser resueltas mediante el comando **solve**.
Por ejemplo:

```
> solve(cos(x)=1/sqrt(2));
```

$$\frac{1}{4}\pi$$

```
> evalf(%);
```

$$.7853981635$$

```
> solve(x^4+x+1,x);
```

$$\text{RootOf}(\_Z^4 + \_Z + 1, \mathit{index} = 1), \text{RootOf}(\_Z^4 + \_Z + 1, \mathit{index} = 2),$$
$$\text{RootOf}(\_Z^4 + \_Z + 1, \mathit{index} = 3), \text{RootOf}(\_Z^4 + \_Z + 1, \mathit{index} = 4)$$

```
> evalf({%});
```

$$\{ .7271360845 + .9340992895\ I, -.7271360845 + .4300142883\ I,$$
$$-.7271360845 - .4300142883\ I, .7271360845 - .9340992895\ I \}$$

También podemos resolver sistemas de ecuaciones :

```
> eqns := {u+v+w=1, 3*u+v=3, u-2*v-w=0};
```

$$\mathit{eqns} := \{ u + v + w = 1, 3\,u + v = 3, u - 2\,v - w = 0 \}$$

```
> sols := solve( eqns );
```

$$\mathit{sols} := \{ u = \frac{4}{5}, w = \frac{-2}{5}, v = \frac{3}{5} \}$$

Diferenciación e Integración.

Para derivar usamos el comando **diff** y para integrar, usamos el comando **int**.

Algunos ejemplos:

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

$$\cos(x)$$

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

$$0$$

```
> diff(sin(x),x$3);
```

$$-\cos(x)$$

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

$$\sin(\cos(x)) - x\cos(\cos(x))\sin(x)$$

O también

```
> f:=x^2+sin(3*x);
```

$$f := x^2 + \sin(3x)$$

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

$$2x + 3\cos(3x)$$

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

$$2 - 9\sin(3x)$$

```
> diff(exp(x)*cos(y),y);
```

$$-\mathbf{e}^x \sin(y)$$

(derivada parcial respecto de y)

Integrar:

```
> int(f,x);
```

$$\frac{1}{3}x^3 - \frac{1}{3}\cos(3x)$$

```
> int(f,x=0..Pi);
```

$$\frac{1}{3}\pi^3 + \frac{2}{3}$$

Ejercicios para practicar

EJERCICIO 2.- Sean tres puntos del plano de coordenadas A=(-1,1/2), B=(1/2, -1) y C=(3/2,1).

a) Representar dichos puntos en un sistema de ejes coordenados.
b) Dibujar el triángulo delimitado por ellos.
c) Colorear la región del plano delimitada por dicho triángulo.
d) Añadir letras a los vértices.
e) Poner título al dibujo.

EJERCICIO 3.- Una isla se encuentra a 1 kilómetro de la costa y en dicha costa existe un pueblo 10 km más abajo. Una compañía de transportes se propone construir un puerto. Sea *d* la distancia del pueblo al puerto y sea $L = \sqrt{101 - 20d + d^2}$ la distancia del puerto a la isla.

La velocidad de un camión es de 70 km/h. y la de un barco de 20 km/h. Expresar el tiempo total necesario para transportar mercancías del pueblo a la isla en función de la distancia del puerto al pueblo. Evaluar dicha función para las distancias d=0,1, …, 10 km. Construir una tabla para dichos valores y representar gráficamente la función.

EJERCICIO 4.- Serie Geométrica.

(a) Para cada n=0...9 determinar la suma de los n primeros términos de la serie:

$$1 + 1/3 + 1/3^2 + ... + 1/3^n + ...$$

llamada serie geométrica de razón r = 1/3. Represente gráficamente dichas sumas y sugiera la convergencia de dicha suma al seguir añadiendo más términos.

(b) Obtener una expresión simbólica que calcule la suma de los n primeros términos de dicha serie y determine su límite cuando n tiende a infinito.

(c) Determinar la suma de la serie

(d) Proceda como en el apartado b) para probar que la serie geométrica converge para todo r tal que $-1 < r < 1$ y calcule su suma.