PRACTICA 0. INSTRUCCIONES DE MANEJO. PRIMEROS EJEMPLOS.

1 PREGUNTAS BÁSICAS.

1.1 ¿DONDE SE OBTIENE EL PROGRAMA?
Maxima es un potente programa de software libre para la manipulación de expresiones numéricas y simbólicas (es muy semejante, en cuanto a su entorno y en cuanto a su uso, de otros programas de cálculo simbólico -como por ejemplo Mathematica). Es decir, Maxima se puede usar como calculadora, pero además nos permite trabajar con expresiones en las que aparecen matrices, variables o parámetros (como x+y o cos(x+3)). En particular podremos emplear el programa para resolver las operaciones más habituales de la asignatura, como operar con vectores o matrices, resolver sistemas de ecuaciones lineales, calcular límites, derivadas, integrales, desarrollos en serie; resolver ecuaciones diferenciales; o representar las gráficas de funciones de 1 y 2 variables.

El programa se puede descargar desde la página web http://maxima.sourceforge.net/es/

Su instalación no presenta ningún problema (basta con ejecutar el fichero .exe). La versión con la que se han elaborado este manual es la versión de wxMaxima 12.04.0

1.2 ¿CÓMO FUNCIONA?
El entorno de ejecución de Maxima puede resultar duro ya que consiste en la clásica línea para ejecutar comandos. Afortunadamente existen entornos más amigables de ejecución y nosotros vamos a trabajar con wxMaxima ya que incluye menús desplegables con la mayoría de los comandos y cuadros de diálogo para introducir los datos necesarios en cada caso, que hace que sea más sencillo trabajar con Maxima.

Una vez instalado y lanzado el programa wxMaxima, para empezar a trabajar con él hay que pulsar la tecla ENTER y aparece el símbolo: [-->

Ya podemos escribir a continuación la operación que queramos: Por ejemplo, si queremos calcular 3+5 escribiremos dicha operación a continuacion de lo anterior, le damos a MAYUSCULA+ENTER (o también a la tecla INTRO del teclado numérico) y obtendremos

```
--> 3+5;
```

Aunque por defecto el programa nos pondrá ";" al final del cada operación, es aconsejable que lo pongamos nosotros (se justifica más adelante).
El programa también nos pone por defecto (%i1) y (%o1) para indicarnos que es la primera entrada (input 1) que hemos introducido y la primera salida (output 1) que hemos obtenido.
También es posible acudir a cualquiera de los resultados anteriores que hayamos obtenido y escribir encima del mismo (por ejemplo, si nos hemos equivocado a la hora de plantear una operación); o copiar el resultado con el cursor y trasladarlo más abajo (es decir, las mismas operaciones que solemos hacer bajo windows, por ejemplo con word, o programas similares).
Si en lugar de ejecutar la operación que queramos hacer con MAYUSCULA+ENTER (o con la tecla INTRO del teclado numérico) ponemos ";" y le damos directamente a ENTER, veremos que obtenemos una nueva línea debajo de la anterior, en la que podremos incluir otras operaciones. Cuando le demos entonces a INTRO, veremos que nos calcula todas las operaciones que hayamos incluido en dicha entrada:

```
--> 3+6;
    4-2;
    4*(-8);
```

1.3 ¿CÓMO OBTENEMOS AYUDAS?
El programa Maxima dispone de una ayuda en castellano (si se ha instalado la versión en este idioma) a la que se accede a través del menú superior o pulsando la tecla F1. En la pantalla que aparece se pueden consultar la información de tres formas distintas:
1. Contenido: Nos permite navegar por el manual. la información aparece agrupada por materias y en cada materia, casi siempre después de una breve introducción, aparecen los comandos relacionados con esa materia en orden alfabético.
2. Índice: Es un listado de todos los comandos disponibles. Seleccionando uno nos aparece su descripción a la derecha.
3. Buscar: Debe introducir una palabra relacionada con lo que desea buscar y obtendrá las materias en las que la palabra proporcionada aparece.

También se accede a la ayuda escribiendo "??" (para búsqueda aproximada), seguido del término a buscar (y sin ";" al final, pero pulsando SHIFT+ENTER).
Por ejemplo, si queremos saber lo que hay relacionado con log, haremos

```
--> ??log
```

En el mismo menú superior tenemos algunos comandos que nos pueden ser útiles, como:
- describe(expr), que nos da ayuda sobre expr
- Ejemplo(expr), que nos da un ejemplo de expr
- A propósito("expr"), que nos da comandos relacionados con expr

1.4 OTROS MANUALES

A través de la web se pueden encontrar gran variedad de manuales (aunque los mismos, y a diferencia de lo que ocurre con el programa, que es libre, son de diferentes autores, que tendremos que reseñar si se van a utilizar por nuestra parte). Yo os puedo recomendar los siguientes:
http://maxima.sourceforge.net/docs/manual/es/maxima.html
http://www.ugr.es/~dpto_am/docencia/Apuntes/Practicas_con_Maxima.pdf
http://www.unp.edu.pe/pers/ripanaque/download/manual.pdf
http://calculoparaingenieros.files.wordpress.com/2012/09/manualmaxima2.pdf

2 MANEJO DE WXMAXIMA.

2.1 INTRODUCCIÓN DE DATOS.

Tenemos la posibilidad de escribir varios comandos dentro de una celda. Por ejemplo, podemos escribir (acabar cada línea con ";" y darle a ENTER):

```
--> 3!;
    %/3;
    %O1*10;
    100/3;
    100.0/3.0;
    cos(%pi);
```

Ahora DAR a INTRO del teclado numérico y observar los resultados. Como vemos para cada entrada aparece una salida en una línea distinta y numerada.

El símbolo "%" tiene varios usos en la ordenes anteriores que vamos a aclarar:
En la segunda línea "%" se utiliza para hacer referencia a la última salida generada por Maxima. En "%o1" se utiliza para hacer referencia a la salida número 1 output 1. Por último el " %pi" en la ´ultima línea se utiliza para hacer referencia a la constante pi. Otras constantes que Maxima reconoce son el número e (%e) y la unidad imaginaria i (%i).

Cada entrada que le damos al programa debe terminar con ";" o con "$". Si termina con ";", Maxima devuelve el resultado, pero si no se desea que aparezca ningún resultado, se debe terminar la línea con $. Este símbolo es útil cuando se quieren hacer varias operaciones intermedias y no se desea que aparezcan los resultados.

Obsérvese que los resultados de 100/3 y en 100.0/3.0 son distintos, aunque aparentemente la operación es la misma. Esto es porque Maxima intenta mantener la precisión y no evalúa expresiones como 100/3 o sqrt(2) (la raíz cuadrada es sqrt) a no ser que se indique. En la entrada %i5 se utiliza coma flotante y, por eso, Maxima evalúa el resultado. Sin embargo, observar lo que ocurre al hacer

```
--> sqrt(2*%e);
```

Para obtener la aproximación numérica de una operación utilizamos el comando float, de manera que

```
--> float(sqrt(2*%e));
```

Podemos escribir comentarios dentro de una determinada sentencia (por ejemplo, para indicar lo que estamos haciendo) para lo que escribiremos dicho comentario entre "/*" y "*/", a continuación le damos a ENTER y debajo escribimos la operación a efectuar:

```
--> /*vamos a calcular el valor aproximado de la raiz de 2e*/
    float(sqrt(2*%e));
```

Maxima puede trabajar con números enteros con cualquier número de dígitos. Sin embargo, cuando la representación es demasiado larga por defecto elimina la mayor parte de los dígitos intermedios. Por ejemplo, si evaluamos 123^123, tendremos

```
--> 123^123;
```

2.2 OPERACIONES ARITMÉTICAS ELEMENTALES.
Los operadores aritméticos son: suma "+", resta "-", producto "*", cociente "/" y elevado "^" o "**". Como siempre, para combinar estos operadores han de tenerse en cuenta los criterios de prioridad en matemáticas: En un primer nivel de prioridad está la potencia; en un segundo nivel están el producto y el cociente; finalmente aparecen la suma y la resta. Si dos operadores tienen la misma prioridad se evalúa primero el que figura a la izquierda y después el de la derecha.
Para cambiar el orden de ejecución de las operaciones pueden utilizarse paréntesis.

Veamos algunos ejemplos: Teclear lo que viene a continuación y observar los resultados (después de cada línea dar a INTRO):

```
--> pez*pez+4*pez+pez;
```

```
--> 2.3;
    /*el punto está reservado para los decimales*/
```

```
--> 3y;
    /*si tecleamos 3y el programa no lo entiende*/
    /*lo correcto es poner 3*y*/
```

EJERCICIO 1:
Ejecutar las siguientes operaciones y observar el orden en que se realizan las operaciones:

```
1) 2*4+3     5) 6/(3-2)   9) 2*(3^2)   13) (b/c + a)^d - e
2) 2*(4+3)   6) (6/3)-2   10) (2*3)^2 14) b/(c+ a^d-e)
3) (2*4)+3   7) 6/3 - 2   11) 2*3^2    15) (b/c + a)^(d-e)
4) 6/3-2     8) (2*3)^2   12) b/(c+a)^(d-e)   16) (3-8)*10
```

2.3 VALORES NUMÉRICOS
Maxima devuelve los resultados de operaciones con números reales (o complejos) de forma exacta y sin aproximaciones decimales.
Veamos ejemplos:

```
--> 2/3+2;
```

```
--> %*5;
```

```
--> 2/3+4/5-3/7*8;
```

Pero si trabajamos con valores decimales el resultado que obtenemos será del mismo tipo:

```
--> sqrt(5.0);
```

```
--> exp(6);
```

```
--> exp(6.0);
```

Para obtener el valor decimal de un número real podemos usar:
- el comando float (ya lo hemos visto con anterioridad).
- añadir una coma "," seguida de numer.
- desde el menú de wxMaxima, seleccionar Numérico->A real (float).

```
--> %pi*2, numer;
```

Maxima redondea, por defecto, a los primeros 16 dígitos. Se puede cambiar la precisión desde Numérico->Establecer precisión. Para representar un número real en este formato se usa el comando bfloat o bien desde Numérico->A real grande (bigfloat).

EJERCICIO 2: Calcular
1) (2/3 - 3/5)*5/2 y lo mismo pero con 30 cifras significativas.
2) 2^100 y lo mismo pero con 10 cifras significativas.
3) Raíz cuadrada de 2 con 47 cifras significativas.
4) Raíz cúbica de 35 con 400 cifras significativas.

2.4 CONSTANTES Y FUNCIONES

En esta última sección de esta práctica introductoria vamos a ver como se definen, asignan o borran valores a variables o etiquetas y cómo se define y evalua una función.

2.4.1 Etiquetas

Una etiqueta se define mediante el símbolo ":" seguido del valor que queremos darle. Por ejemplo, si queremos que la etiqueta "amapola" tome el valor 6, haremos

```
--> amapola:6;
```

de forma que si calculamos 3*amapola-15 obtendremos

```
--> 3*amapola-15;
```

De igual forma podemos definir una fórmula y asignar valores a una variable. Veamos el siguiente ejemplo (observar lo que se hace):

```
--> v:2$ w:3$
    formula:v*2/w-v*5;
```

2.4.2 Constantes

Maxima tiene constantes matemáticas predeterminadas. Por ejemplo el número e y por lo tanto conviene no usar dichas constantes como nombres de etiquetas o variables para evitar errores. Dichas constantes se diferencian porque llevan el símbolo % delante de la constante y sin espacio (%e). Otros ejemplos son: %i (número i), %pi (número p), %phi (razón aurea), etc.
Tampoco conviene utilizar como nombres de etiquetas aquellas que definen algún comando en Maxima, por ejemplo, integrate, diff, sum, ...

2.4.3 Funciones

Para definir cualquier función, la sintaxis es: nombre de la función y argumentos, por ejemplo, f(x) o h(x;y) seguido del símbolo ":=" y la definición de la función.
Ejemplos de función de una y varias variables:

```
--> f(x):=x^2+sin(x)/x;
```

```
--> h(x,y):=2*%e^(x*y)-3/x^2;
```

Se evalúa una función sustituyendo los argumentos por sus valores correspondientes:
Por ejemplo, si queremos evaluar la anterior función f(x) en el punto x=2, y la función
h(x,y) en el punto (2,4), haremos

```
--> f(2);
    h(2,4);
```

Maxima tiene funciones predefinidas. Algunas de las más comunes son: sqrt (raíz cuadrada), sin (seno), cos (coseno), tan (tangente), exp (exponencial), log (logaritmo en base e), abs (valor absoluto).

Podemos ver (y borrar) las funciones que hemos definido y su definición a través del menú superior, en
Maxima->Mostrar funciones (borrar funciones)
y
Maxima->Mostrar definiciones.

También podemos definir funciones a trozos con
if condición then expresión 1 else expresión 2.
Por ejemplo si hacemos

```
--> g(x):=if x<=0 then x^3-x^2+3 else exp(x);
    [g(-2),g(0),g(4)];
```

habremos definido la función g(x)=x^3-x^2+3, si x<=0,
mientras que si x>0, se tiene que g(x)=exp(x);
también le hemos pedido que evalúe dicha función en x=-2, 0 y 4.

EJERCICIO 3: Definir las funciones f(x)=x^3-x^2+3 y g(x)=x^3-x^2+k y calcular f(-2), f(1/2), g(-2). Si hacemos k=-1, calcular g(f(x)) y g(f(0)).

EJERCICIO 4: Definir la función f(x,y)=x^3-y^2-1 y evaluarla en los puntos (-2,1) y (0,3).

3 UN RÁPIDO REPASO POR wxMAXIMA.

A continuación vamos a efectuar un rápido repaso por las operaciones que podemos efectuar en wxMaxima. En esta primera aproximación, solo se trata de que el alumno comprenda lo que wxMaxima puede realizar, aunque inicialmente no comprenda como se escriben determinadas sentencias.
En las prácticas siguientes de la asignatura, estudiaremos con más profundidad cada uno de los apartados siguientes. De momento, sólo se trata de que se vaya entendiendo lo que se va a realizar en los ejercicios siguientes:

3.1 SOBRE ALGEBRA LINEAL

3.1.1 Ejercicios con vectores

Podemos definir los vectores: v=(2,-3) y w=(1,2):

```
(%i) v:[2,-3];
     w:[1,2];
```

Si queremos escoger la componente de un vector; por ejemplo la segunda componente del vector v anterior:

```
(%i) v[2];
```

Si queremos realizar la suma: v+w

```
(%i) v+w;
```

po el Producto por escalares: -3·v

```
(%i) -3*v;
```

También podemos calcular el Producto escalar de dos vectores: v·w

```
(%i) v.w;
```

o calcular la Norma (o módulo) de un vector: ||v||

```
(%i) sqrt(v.v);
```

3.1.2 Ejercicios con matrices

Podemos definir las matrices siguientes:

$$A=\begin{pmatrix}1 & 2\\ 3 & 4\end{pmatrix},\quad B=\begin{pmatrix}0 & 1\\ -2 & 1\end{pmatrix},\quad C=\begin{pmatrix}1 & 2 & 3\\ 4 & 5 & 6\\ 7 & 8 & 9\end{pmatrix}$$

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

Elegimos el coeficiente {2,3} de C:

```
--> C[2][3];
```

Si queremos sumar dos matrices: A+B

```
--> A+B;
```

o realizar el producto por escalares: 3·A

```
--> 3*A;
```

o el producto de dos matrices: A·B (OJO: no es A*B)
Comprobar la diferencia tecleando lo siguiente:

```
--> A.B;
    A*B;
```

Se puede multiplicar una matriz por un vector vector: A·v

```
--> A.v;
```

o hallar la matriz traspuesta: A^t

```
--> transpose(A);
```

o la inversa: A^-1 (OJO: no es A^(-1)); ver la diferencia:

```
--> invert(A);
    A^-1;
```

Si queremos hallar el determinante: |A|

```
--> determinant(A);
```

o calcular su rango: rg(C)

```
--> rank(C);
```

Más adelante veremos la importancia de calcular el
Núcleo: Ker(C)

```
--> nullspace(C);
```

```
--> args(%);
```

y de la Imagen: Im(C)

```
--> columnspace(C);
    args(%);
```

3.1.3 Ejercicios con sistemas de ecuaciones lineales

SCD: x+y=2,2x+3y=2

```
--> linsolve([x+y=2,2*x+3*y=2],[x,y]);
```

SI: x+y=2,2x+2y=2

```
--> linsolve([x+y=2,2*x+2*y=2],[x,y]);
```

SCI (solución paramétrica): x+y=2,2x+2y=4

```
--> linsolve([x+y=2,2*x+2*y=4],[x,y]);
```

SCI (solución implícita): x+y=2,2x+2y=4

```
--> linsolve_params:false$
    linsolve([x+y=2,2*x+2*y=4],[x,y]);
```

3.1.4 Ejercicios de diagonalización

Para hallar el polinomio característico: p_A(x)

```
--> charpoly(A,x);
```

```
--> expand(%);
```

y los Autovalores: lambda_i

```
--> eigenvalues(A);
```

```
--> solve(charpoly(A,x),x);
```

o los Autovectores: v_i

```
--> eigenvectors(A);
```

3.2 SOBRE CÁLCULO

3.2.1 Ejercicios sobre Cálculo Diferencial

Podemos calcular límites finitos: lim_x->0 sen x/x

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

o límites infinitos: lim_x->-inf (2x-1)^3/(x^2+1)

```
--> limit((2*x-1)^3/(x^2+1),x,-inf);
```

Se puede obtener la suma de Series numéricas: sum 1/n^2

```
--> simpsum;
    sum(1/n^2, n, 1, inf);
    simpsum:true;
    sum(1/n^2, n, 1, inf);
    reset(simpsum)$
```

Para hallar la derivada de una función (de una o varias variables). Por ejemplo, para hallar la derivada parcial 3 veces respecto de x de la función sin 2xy:

```
--> diff(sin(2*x*y),x,3);
```

o la matriz Hessiana: H(sin(x+y))

```
--> hessian(sin(x+y),[x,y]);
```

o un polinomio de Taylor: (sin x)/x, n=10, x0=0

```
--> taylor(sin(x)/x,x,0,10);
```

```
--> ratdisrep(%);
```

```
--> powerdisp;
    powerdisp:true$
    ratdisrep(taylor(sin(x)/x,x,0,10));
    powerdisp:false$
    ratdisrep(taylor(sin(x)/x,x,0,10));
```

3.2.2 Ejercicios sobre Cálculo Integral

Podemos hallar una Primitiva: Por ejemplo de f(x)= x/(x^3-2x^2+1):

```
--> integrate(x/(x^3-2*x^2+1),x);
```

o una integral definida: Por ejemplo, para calcular en el intervalo [0,pi] la integral de (cos(x))^2:

```
--> integrate(cos(x)^2,x,0,%pi);
```

También podemos hallar integrales impropias: Para hallar en el intervalo [-inf,inf] la integral de f(x)= e^(-x^2/2):

```
--> integrate(exp(-x^2/2),x,-inf,inf);
```

```
--> rootscontract(%);
```

3.2.3 Ejercicios sobre Cálculo Vectorial

Gradiente: grad x·y^2·sin(z)

```
--> load(vect)$
```

```
--> express(grad(x*y^2*sin(z)));
```

```
--> ev(%,diff);
```

Divergencia: div (x,z·y^2,sin(x+z))

```
--> express(div([x,z*y^2,sin(x+z)]));
```

```
--> ev(%,diff);
```

Rotacional: rot (x^2,y^3,z)

```
--> express(curl([x^2,y^3,z]));
```

```
--> ev(%,diff);
```

Laplaciano: lap x·y^2

```
--> express(laplacian(x*y^2));
```

```
--> ev(%,diff);
```

3.3 SOBRE ECUACIONES DIFERENCIALES
3.3.1 Ejercicios sobre EDO de 1er orden

De variables separadas: y'=sen x/sqrt(y)

```
--> ode2('diff(y,x)=sin(x)/sqrt(y),y,x);
```

EDO Lineales: y'=y/x+1, y(1)=3

```
--> ode2('diff(y,x)=y/x+1,y,x);
```

```
--> ic1(%,x=1,y=3);
```

Homogéneas: y'=(x+y)/(2x+y)

```
--> ode2('diff(y,x)=(x+y)/(2*x+y),y,x);
```

Factor integrante: y'=(4x^3-y)/(2x-x^4/y)

```
--> ode2('diff(y,x)=(4*x^3-y)/(2*x-x^4/y),y,x);
```

3.4 SOBRE REPRESENTACIONES GRÁFICAS
3.4.1 Representaciones en 2D

```
--> plot2d(sin(2*x),[x,-2*%pi,2*%pi]);
```

```
--> plot2d([x^2,sqrt(2*x)],[x,-2,2]);
```

3.4.2 Representaciones de curvas y superficies en 3D

```
--> plot3d(cos(x*y),[x,-3,3],[y,-3,3]);
```

```
--> plot3d([cos(u)*cos(v),sin(u)*cos(v),sin(v)],
    [u,0,2* %pi],[v,- %pi/2, %pi/2]);
```

habremos definido la función g(x)=x^3-x^2+3, si x<=0,
mientras que si x>0, se tiene que g(x)=exp(x);
también le hemos pedido que evalúe dicha función en x=-2, 0 y 4.

4 GRÁFICOS EN EL PLANO CON plot2d.

4.1 COORDENADAS CARTESIANAS.

El comando que se utiliza para representar la gráfica de una función de una variable real es plot2d que actúa, como mínimo, con dos parámetros: la función (o lista de funciones a representar), y el intervalo de valores para la variable x. Al comando plot2d se puede acceder también a través del menú Gráficos->Gráficos 2D

```
--> plot2d(sin(2*x),[x,-2*%pi,2*%pi]);
```

```
--> plot2d([x^2,sqrt(2*x)],[x,-2,2]);
```

Cuando pulsamos en el menú Gráficos->Gráficos 2D , aparece una ventana de diálogo con varios campos que podemos completar o modificar:
a) Expresión(es). La función o funciones que queramos dibujar. Por defecto, wxMaxima rellena este espacio con % para referirse a la salida anterior.
b) Variable x. Aquí establecemos el intervalo de la variable x donde queramos representar la función.
c) Variable y. Ídem para acotar el recorrido de los valores de la imagen.
d) Graduaciones. Nos permite regular el número de puntos en los que el programa evalúa una función para su representación en polares. Veremos ejemplos en la sección siguiente.
e) Formato. Maxima realiza por defecto la gráfica con un programa auxiliar. Si seleccionamos "en línea", dicho programa auxiliar es wxMaxima y obtendremos la gráfica en una ventana alineada con la salida correspondiente. Hay dos opciones más y ambas abren una ventana externa para dibujar la gráfica requerida: "gnuplot" es la opción por defecto que utiliza el programa Gnuplot para realizar la representación; también está disponible la opción "openmath" que utiliza el programa XMaxima. Prueba las diferentes opciones y decide cuál te gusta más.
f) Opciones. Aquí podemos seleccionar algunas opciones para que, por ejemplo, dibuje los ejes de coordenadas ("set zeroaxis;"); dibuje los ejes de coordenadas, de forma que cada unidad en el eje Y sea igual que el eje X ("set size ratio 1; set zeroaxis;"); dibuje una cuadrícula ("set grid;") o dibuje una gráfica en coordenadas polares ("set polar; set zeroaxis;"). Esta última opción la comentamos más adelante.
g) Gráfico al archivo. Guarda el gráfico en un archivo con formato Postscript.
Evidentemente, estas no son todas las posibles opciones. La cantidad de posibilidades que tiene Gnuplot es inmensa.
Observación. El prefijo “wx” añadido a plot2d o a cualquiera del resto de las órdenes que veremos en esta práctica (plot3d, draw2d, draw3d) hace que wxMaxima pase automáticamente a mostrar los gráficos en la misma ventana y no en una ventana separada. Es lo mismo que seleccionar en línea. Por ejemplo,

```
--> wxplot2d(sin(2*x),[x,-2*%pi,2*%pi]);
```

```
--> plot2d(x/(x^2-4),[x,-6,6],[y,-6,6],
    [gnuplot_preamble, "set zeroaxis;"])$
```

```
--> plot2d(x/(x^2-4),[x,-6,6],[y,-6,6],
    [gnuplot_preamble, "set size ratio 1; set zeroaxis;"])$
```

```
--> plot2d(x/(x^2-4),[x,-6,6],[y,-6,6],
    [gnuplot_preamble, "set grid;"])$
```

4.2 COORDENADAS POLARES.
Al representar una curva en coordenadas polares estamos escribiendo la longitud del vector como una función que depende del ángulo. En otras palabras, para cada ángulo fijo decimos cuál es el módulo del vector. El ejemplo más sencillo de función que a cualquiera se nos viene a la cabeza son las funciones constantes: La función f :[0,2pi]->R, f()=1 tiene como imagen aquellos vectores que tienen módulo 1 y argumento entre 0 y 2pi. Para ello, tenemos que seleccionar "set polar; set zeroaxis;" en el campo Opciones de Gráficos 2D:

```
--> plot2d([1], [ph,0,2*%pi],
    [plot_format, gnuplot],
    [gnuplot_preamble, "set polar; set zeroaxis;"],[x,-1,1])$
```

y si queremos que la gráfica sea proporcionada, hacemos

```
--> plot2d([1], [ph,0,2*%pi],
    [plot_format, gnuplot],
    [gnuplot_preamble, "set polar;set size ratio 1; set zeroaxis;"],
    [x,-1,1])$
```

Otra gráfica viene dada por

```
--> plot2d(ph,[ph,0,4*%pi],
    [gnuplot_preamble, "set polar; set zeroaxis;"],[x,-15,15])$
```

Observamos que la hélice resultante no es nada “suave”. Para conseguir el efecto visual de una línea curva como es esta hélice, añadimos el parámetro nticks. Por defecto, para dibujar una gráfica en paramétricas el programa evalúa en 10 puntos. Para aumentar este número de puntos, aumentamos dicho parámetro, por ejemplo nticks=30, o bien, podemos regularlo desde el campo Graduaciones dentro de de Gráficos 2D.

```
--> plot2d(ph,[ph,0,4*%pi],
    [gnuplot_preamble, "set polar; set zeroaxis;"], [nticks,30],
    [x,-15,15])$
```

Si representamos la función
r(theta)=exp(cos(theta))-2cos(4theta)+sin(theta/12)^5
obtenemos algo parecido a una mariposa:

```
--> r(ph):=exp(cos(ph))-2*cos(4*ph)+sin(ph/12)^5$
    plot2d(r(ph),[ph,0,2*%pi],
    [gnuplot_preamble, "set polar; set zeroaxis;"],[x,-5,5])$
```

4.3 COORDENADAS PARAMÉTRICAS.
El programa wxMaxima nos permite también representar curvas en forma paramétrica, es decir, curvas definidas como (x(t),y(t)), donde el parámetro t varía en un determinado intervalo compacto [a,b].
Para ello, dentro del comando plot2d añadimos “parametric” de la forma siguiente: plot2d([parametric,x(t),y(t),[t,a,b]])
Para acceder a esta opción de la función plot2d podemos hacerlo a través del botón Especial que aparece en la parte superior derecha de la ventana de diálogo Gráficos 2D.
Para terminar, aquí tienes algunas curvas planas interesantes:
Astroide: Es la curva trazada por un punto fijo de un círculo de radio r que rueda sin deslizar dentro de otro círculo fijo de radio 4r.
Sus ecuaciones paramétricas son:

```
--> plot2d([parametric,cos(t)^3,sin(t)^3,[t,0,2*%pi],[nticks,50]]);
```

Cardioide: Es la curva trazada por un punto fijo de un círculo de radio r que rueda sin deslizar alrededor de otro círculo fijo del mismo radio.
Sus ecuaciones paramétricas
son:

```
--> plot2d([parametric,(1+cos(t))*cos(t),(1+cos(t))*sin(t),
    [t,0,2*%pi],[nticks,50]]);
```

Lemniscata de Bernoulli:Es el lugar geométrico de los puntos P del plano, cuyo producto de distancias a dos puntos fijos F1 y F2, llamados focos, verifica la igualdad abs(P-F1)*abs(P-F2)=1/4*abs(F1-F2)^2
En coordenadas cartesianas esta curva viene dada por la ecuación (x^2+y^2)^2=x^2-y^2. Aquí tienes sus ecuaciones paramétricas:

```
--> plot2d([parametric, (cos(t))/(1+sin(t)^2),
    (cos(t)*sin(t))/(1+sin(t)^2), [t,0,2*%pi],[nticks,70]]);
```

Espiral equiangular: También llamada espiral logarítmica. Es aquella espiral en la que el radio vector corta a la curva en un ángulo constante alfa. Sus ecuaciones paramétricas son:

```
--> %alpha:%pi/2-0.2$
    plot2d([parametric,exp(t*cot(%alpha))*cos(t),
    exp(t*cot(%alpha))*sin(t),[t,0,4*%pi],[nticks,90]]);
```

Cicloide: También conocida como tautocrona o braquistocrona. Es la curva que describiría un punto de una circunferencia que avanza girando sin deslizar. Sus ecuaciones paramétricas son:

```
--> plot2d([parametric,t-sin(t),1-cos(t),[t,0,6*%pi],[nticks,90]]);
```

```
--> plot3d([cos(u)*cos(v),sin(u)*cos(v),sin(v)],
    [u,0,2* %pi],[v,- %pi/2, %pi/2]);
```

Al comando plot3d se puede acceder a través del menú Gráficos->Gráficos 3D. Después de esto aparece una ventana con varios campos para rellenar:

a) Expresión. La función o funciones que vayamos a dibujar.
b) Variable. Hay dos campos para indicar el dominio de las dos variables.
c) Cuadrícula. Indica cuántas valores se toman de cada variable para representar la función. Cuanto mayor sea, más suave será la representación a costa de aumentar la cantidad de cálculos.
d) Formato. Igual que en plot2d, permite escoger qué programa se utiliza para representar la función. Se puede girar la gráfica en todos ellos salvo si escoges “en línea”.
e) Opciones. Las comentamos a continuación.
f) Gráfico al archivo. Permite elegir un fichero donde se guardará la gráfica.
Quizá la mejor manera de ver el efecto de las opciones es repetir el dibujo anterior. La primera de ellas es "set pm3d at b" que dibuja la superficie usando una malla y en la base añade curvas de nivel (como si estuviéramos mirando la gráfica desde arriba):

```
--> plot3d(cos(x*y), [x,-5,5], [y,-5,5], [plot_format,gnuplot],
    [gnuplot_preamble, "set pm3d at b"])$
```

La segunda hace varias cosas, "set pm3d at s" nos dibuja la superficie coloreada, "unset surf" elimina la malla y "unset colorbox" elimina la barra que teníamos en la derecha con la explicación de los colores y su relación con la altura (el valor de la función):

```
--> plot3d(cos(x*y), [x,-5,5], [y,-5,5], [plot_format,gnuplot],
    [gnuplot_preamble, "set pm3d at s; unset surf;
    unset colorbox"])$
```

La tercera, "set pm3d map", nos dibuja un mapa de curvas de nivel (gráfico de densidad) con alguna ayuda adicional dada por el color: Así, si queremos representar solamente el gráfico de densidad para esta superficie, haremos:

```
--> plot3d(cos(x*y), [x,-5,5], [y,-5,5],
    [gnuplot_preamble,"set pm3d map"]);
```

La cuarta, "set hidden3d", sólo muestra la parte de la superficie que sería visible desde nuestro punto de vista. En otras palabras, hace la superficie sólida y no transparente:

```
--> plot3d(cos(x*y), [x,-5,5], [y,-5,5], [plot_format,gnuplot],
    [gnuplot_preamble, "set hidden3d"])$
```

En el dibujo anterior (en el papel) es posible que no aprecie bien. A simple vista parece el mismo dibujo que teníamos dos salidas antes. Observa bien: hay una pequeña diferencia. El uso de pm3d hace que se coloree el dibujo, pero cuando decimos que no se muestra la parte no visible de la figura nos estamos refiriendo a la malla. Quizá es mejor dibujar la malla y el manto de colores por separado para que se vea la diferencia. Esta opción no viene disponible por defecto en wxMaxima. Ten en cuenta que las opciones que tiene Gnuplot son casi infinitas y sólo estamos comentando algunas.

```
--> plot3d(x^2-y^2, [x,-5,5], [y,-5,5], [plot_format,gnuplot],
    [gnuplot_preamble, "set pm3d at b; set hidden3d"])$
```

```
--> plot3d(x^2-y^2, [x,-5,5], [y,-5,5], [plot_format,gnuplot],
    [gnuplot_preamble, "set pm3d at b"])$
```

La quinta y la sexta opciones nos permiten dibujar en coordenadas esféricas o cilíndricas. Ya veremos ejemplos más adelante.
Si queremos representar solamente un gráfico de contorno (curvas de nivel), es posible mediante contour_plot, como vemos a continuación:

5 GRÁFICOS EN 3D.

Con Maxima se pueden representar funciones de dos variables de forma similar a como hemos representado funciones de una. La principal diferencia es que vamos a utilizar el comando plot3d en lugar de plot2d, pero igual que en el caso anterior, son obligatorios la función o funciones y el dominio que tiene que ser de la forma [a,b]x[c,d]:
plot3d(f(x,y),[x,a,b],[y,c,d]) gráfica de f(x,y) en [a,b]x[c,d]

```
--> plot3d(cos(x*y),[x,-3,3],[y,-3,3]);
```

La gráfica que acabas de obtener se puede girar sin más que pinchar con el ratón sobre ella y deslizar el cursor.
plot3d también permite trazar la gráfica de una superficie definida en forma paramétrica; sin embargo, no permite gráficas de curvas en R3: Para ello usaremos
plot3d([x(u,v),y(u,v),z(u,v)],[u,umin,umax],[v,vmin,vmax])
Así, si queremos representar la esfera unitaria definida paramétricamente mediante (u,v)->(sen u sen v,cos u sen v,cos v), con u entre 0 y 2pi, y v entre -pi/2 y pi/2:

```
--> plot3d([cos(u)*cos(v),sin(u)*cos(v),sin(v)],
    [u,0,2* %pi],[v,- %pi/2, %pi/2]);
```

Al comando plot3d se puede acceder a través del menú Gráficos->Gráficos 3D. Después de esto aparece una ventana con varios campos para rellenar:

a) Expresión. La función o funciones que vayamos a dibujar.
b) Variable. Hay dos campos para indicar el dominio de las dos variables.
c) Cuadrícula. Indica cuántas valores se toman de cada variable para representar la función. Cuanto mayor sea, más suave será la representación a costa de aumentar la cantidad de cálculos.
d) Formato. Igual que en plot2d, permite escoger qué programa se utiliza para representar la función. Se puede girar la gráfica en todos ellos salvo si escoges “en línea”.
e) Opciones. Las comentamos a continuación.
f) Gráfico al archivo. Permite elegir un fichero donde se guardará la gráfica.
Quizá la mejor manera de ver el efecto de las opciones es repetir el dibujo anterior. La primera de ellas es "set pm3d at b" que dibuja la superficie usando una malla y en la base añade curvas de nivel (como si estuviéramos mirando la gráfica desde arriba):

```
--> plot3d(cos(x*y), [x,-5,5], [y,-5,5], [plot_format,gnuplot],
    [gnuplot_preamble, "set pm3d at b"])$
```

La segunda hace varias cosas, "set pm3d at s" nos dibuja la superficie coloreada, "unset surf" elimina la malla y "unset colorbox" elimina la barra que teníamos en la derecha con la explicación de los colores y su relación con la altura (el valor de la función):

```
--> plot3d(cos(x*y), [x,-5,5], [y,-5,5], [plot_format,gnuplot],
    [gnuplot_preamble, "set pm3d at s; unset surf;
    unset colorbox"])$
```

La tercera, "set pm3d map", nos dibuja un mapa de curvas de nivel (gráfico de densidad) con alguna ayuda adicional dada por el color: Así, si queremos representar solamente el gráfico de densidad para esta superficie, haremos:

```
--> plot3d(cos(x*y), [x,-5,5], [y,-5,5],
    [gnuplot_preamble,"set pm3d map"]);
```

La cuarta, "set hidden3d", sólo muestra la parte de la superficie que sería visible desde nuestro punto de vista. En otras palabras, hace la superficie sólida y no transparente:

```
--> plot3d(cos(x*y), [x,-5,5], [y,-5,5], [plot_format,gnuplot],
    [gnuplot_preamble, "set hidden3d"])$
```

En el dibujo anterior (en el papel) es posible que no aprecie bien. A simple vista parece el mismo dibujo que teníamos dos salidas antes. Observa bien: hay una pequeña diferencia. El uso de pm3d hace que se coloree el dibujo, pero cuando decimos que no se muestra la parte no visible de la figura nos estamos refiriendo a la malla. Quizá es mejor dibujar la malla y el manto de colores por separado para que se vea la diferencia. Esta opción no viene disponible por defecto en wxMaxima. Ten en cuenta que las opciones que tiene Gnuplot son casi infinitas y sólo estamos comentando algunas.

```
--> plot3d(x^2-y^2, [x,-5,5], [y,-5,5], [plot_format,gnuplot],
    [gnuplot_preamble, "set pm3d at b; set hidden3d"])$
```

```
--> plot3d(x^2-y^2, [x,-5,5], [y,-5,5], [plot_format,gnuplot],
    [gnuplot_preamble, "set pm3d at b"])$
```

La quinta y la sexta opciones nos permiten dibujar en coordenadas esféricas o cilíndricas. Ya veremos ejemplos más adelante.
Si queremos representar solamente un gráfico de contorno (curvas de nivel), es posible mediante contour_plot, como vemos a continuación:

```
--> contour_plot(cos(x*y), [x,-5,5], [y,-5,5])$
```

6 GRÁFICOS CON draw.

El paquete draw se distribuye conjuntamente con Maxima y constituye una interfaz que comunica de manera muy eficiente Maxima con Gnuplot. Este paquete incorpora una considerable variedad de funciones y opciones que permiten obtener la representación de un amplio número de objetos gráficos bidimensionales y tridimensionales. En este caso nos vamos a centrar solamente en GRAFICOS TRIDIMENSIONALES, ya que con draw vamos a poder realizar gráficas que no podemos realizar con plot3g, como por ejemplo, representar curvas en R3. No obstante, invitamos al lector a que profundice en este paquete (a menudo mucho más sencillo e intuitivo que plot) para ver otras opciones. Toda la información al respecto se puede ver fácilmente en el manual que se encuentra en:
http://www.ugr.es/~dpto_am/docencia/Apuntes/Practicas_con_Maxima.pdf
Para poder utilizar el paquete draw es preciso cargarlo en la memoria, y para ello se utiliza la función load(draw).
Las funciones draw, draw2d y draw3d devuelven las salidas gráficas en una ventana de Gnuplot, aparte de la ventana de trabajo actual. No obstante el entorno gráfico wxMaxima permite utilizar las funciones wxdraw, wxdraw2d y wxdraw3d que sí devuelven las salidas en el cuaderno de trabajo actual. Debe tenerse presente que al utilizar la función wxdraw3d, el punto de vista de la gráfica obtenida no puede ser cambiado en tiempo real.
En esta práctica se van a mostrar los resultados mediante las función draw3d, pero todos los ejemplos mostrados pueden ser ejecutados, sin ningún problema, con la función wxdraw3d (lo mismo ocurre con draw y draw2d, como puede verse en el manual anteriormente citado).

Principales objetos gráficos tridimensionales incorporados en draw:
- points([x1,x2,...];[y1,y2,,,,],[z1,z2,...]) representa puntos en R3
- vector([p1,p2,p3],[v1,v2,v3]) nos da el vector (v1,v2,v3) con punto de aplicación (p1,p2,p3)
- explicit(f(x,y),x,xmin,xmax,y,ymin,ymax) representa a una función explícita f en el dominio dado.
- implicit(E(x,y,z),x,xmin,xmax,y,ymin,ymax,z,zmin,zmax) representa a una ecuación implícita E en el dominio dado
- parametric(fx,fy,fz,t,tmin,tmax) curva paramétrica tridimensional, cuyo parámetro t satisface tmin<t<tmax
- cylindrical(r(z,theta),z,zmin,zmax,theta,thetamin,thetamax), nos da una función cilíndrica r donde los parámetros varían en los intervalos que se indican
- spherical(r(fi;theta),fi,fimin,fimax,theta,thetamin,thetamax) nos da una función esférica r donde los parámetros varían en los intervalos que se indican
Veamos ejemplos de todos éstos comandos (y otros):

Aquí se muestra la gráca del vector cuyo punto de aplicación es el origen y cuya parte vectorial es (1,1,1): (Primero tenemos que llamar al paquete draw)

```
--> load(draw)$
```

```
--> draw3d(vector([0,0,0],[1,1,1]));
```

Esto muestra la gráfica de la función f(x,y)=sen x + sen(xy), en [0,2pi]x[0,2pi]:

```
--> draw3d(explicit(sin(x)+sin(x*y),x,0,2*%pi,y,0,2*%pi));
```

Aquí se muestra la gráfica de una superficie generada a partir de una ecuación implícita (se indica la ecuación y donde varían las 3 variables x, y, z):

```
--> draw3d(implicit((sqrt(x^2+y^2)-4)^2+z^2=4,x,-6,6,
    y,-6,6,z,-2,2) );
```

Esta es la gráfica de la curva dada por la parametrización t->(cos t,sen t,t/8), con 0<t<4pi (estas curvas en R3 no podemos representarlas mediante plot3d):

```
--> wxdraw3d(parametric(cos(t),sin(t),t/8,t,0,4*%pi));
```

Aquí se muestra la gráfica de la superficie definida por la parametrización (u,v)->((2+cos v)cos u,(2+cos v)sin u,sin v), con 0<u<2pi y 0<v<2pi:

```
--> draw3d(parametric_surface((2+cos(v))*cos(u),
    (2+cos(v))*sin(u),sin(v),u,0,2*%pi,v,0,2*%pi));
```

Esta es la gráfica de la superficie definida en coordenadas cilíndricas mediante (z,t)->cos z, con -2<z<2 y 0<t<2pi:

```
--> draw3d(cylindrical(cos(z),z,-2,2,t,0,2*%pi));
```

He aquí la gráfica de la superficie definida en coordenadas esféricas mediante (a,t)->1, con 0<a<2pi y 0<t<pi:

```
--> draw3d(spherical(1,a,0,2*%pi,t,0,%pi));
```

A continuación incluimos otros ejemplos donde aparecen determinadas opciones que podemos realizar:

Esto define una curva, algunos puntos sobre ésta y algunos vectores tangentes a la misma:

```
--> a(t):=[cos(t),sin(t),t/8]$
    define("a'"(t),diff(a(t),t))$
    t0:create_list(i*%pi/4,i,0,16)$
    T:create_list( vector(a(i),"a'"(i)),i,t0)$
    P:map(a,t0)$
```

```
--> objetos:([nticks=100,color=red,
    apply(parametric,append(a(t),[t,0,4*%pi]))
    color=blue,unit_vectors=true,T,
    color=green,point_type=7,points(P),
    user_preamble="set size ratio 1",
    title="Campo vectorial tangente",
    xyplane=0,axis_3d=false,
    xtics=false,ytics=false,ztics=false,
    xaxis=true,yaxis=true,zaxis=true,
    xlabel="x",ylabel="y",zlabel="z"])$
```

```
--> draw3d(objetos);
```

La superficie implícita definida anteriormente pero con otras opciones:

```
--> draw3d(x_voxel=17,y_voxel=17,z_voxel=15,
    palette=[12,5,27],enhanced3d=true,
    implicit((sqrt(x^2+y^2)-4)^2+z^2=4,x,-6,6,
    y,-6,6,z,-2,2));
```