PRÁCTICA 1:
Primeros pasos con wxMaxima

Maxima es un programa que realiza cálculos matemáticos de forma tanto numérica como simbólica, esto es, sabe tanto manipular números como calcular la derivada de una función. Sus capacidades cubren sobradamente las necesidades de un alumno de un curso de Cálculo en unos estudios de Ingeniería. Se encuentra disponible bajo licencia GNU GPL tanto el programa como los manuales del programa.

Hay muchos programas que cumplen en mayor o menor medida los requisitos que se necesitan para enseñar y aprender Cálculo. Sólo por mencionar algunos, y sin ningún orden particular, casi todos conocemos *Mathematica* (©Wolfram Research) o *Maple* (©Maplesoft). También hay una larga lista de programas englobados en el mundo del software libre que se pueden adaptar a este trabajo.

Siempre hay que intentar escoger la herramienta que mejor se adapte al problema que se presenta y, en nuestro caso, *Maxima* cumple con creces las necesidades de un curso de Cálculo. Es evidente que *Mathematica* o *Maple* también pero creemos que el uso de programas de software libre permite al alumno y al profesor estudiar cómo está hecho, ayudar en su mejora y, si fuera necesario y posible, adaptarlo a sus propias necesidades.

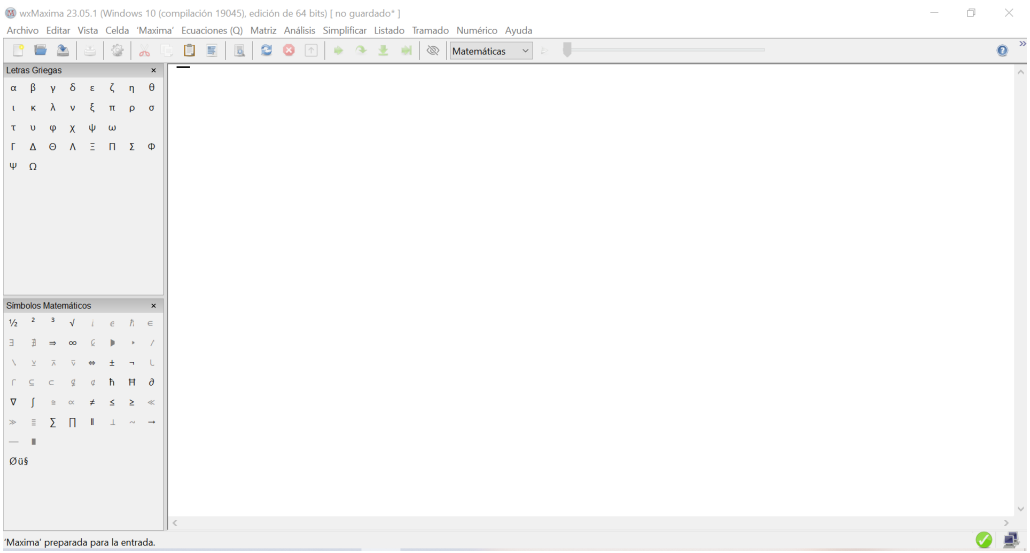
No es nuestra intención hacer una historia de *Maxima*, ni explicar cómo se puede conseguir o instalar, tampoco aquí encontrarás ayuda ni preguntas frecuentes ni nada parecido. Cualquiera de estas informaciones se encuentra respondida de manera detallada en la página web del programa:

https://maxima.sourceforge.io/

En esta página puedes descargarte el programa y encontrar abundante documentación sobre cómo instalarlo.

Vamos a comenzar familiarizándonos con *Maxima* y con el entorno de trabajo *wxMaxima*. Cuando iniciamos el programa se nos presenta una ventana como la de la Figura. En la parte superior tienes el menú con las opciones usuales (abrir, cerrar, guardar) y otras relacionadas con las posibilidades más "matemáticas" de *Maxima*. En segundo lugar aparecen algunos iconos que sirven de atajo a algunas operaciones y la ventana de trabajo.

Los paneles de comandos que aparecen en la parte izquierda, los abrimos (o cerramos) yendo en el menú a Vista —> Sidebars.



Operaciones básicas

+	suma
*	producto
/	división
^ o **	potencia
sqrt( )	raíz cuadrada

Constantes

Además de las funciones usuales, *Maxima* también conoce el valor de algunas de las constantes típicas.

%pi	el número π
%e	el número e
%i	la unidad imaginaria
%phi	la razón aúrea, $\frac{1+\sqrt{5}}{2}$

```
(%i1) 3^37;
(%o1) 450283905890997363
(%i2) %pi;
(%o2) π
(%i3) pi;
(%o3) pi
```

Cálculo simbólico

Cuando hablamos de que *Maxima* es un programa de cálculo simbólico, nos referimos a que no necesitamos trabajar con valores concretos. Fíjate en el siguiente ejemplo:

```
(%i4) x/2+3*x/5;
(%o4) 11 x
      ----
       10
```

Como habíamos comentado, nos interesa sobre todo el cálculo simbólico. Pero imaginemos que queremos saber una aproximación decimal de alguna operación, por ejemplo $3\sqrt{2}+25$. Tenemos tres formas fundamentales para hacerlo:

float(*número*)	Expresión decimal de *número*
número,numer	Expresión decimal de *número*
bfloat(*número*)	Expresión decimal larga de *número*

```
(%i5) 3*sqrt(2)+25;
(%o5) 3 √2+25
(%i6) float(3*sqrt(2)+25);
(%o6) 29.24264068711928
(%i7) 3*sqrt(2)+25,numer;
(%o7) 29.24264068711928
(%i8) bfloat(3^37);
(%o8) 4.502839058909974b17
```

La última expresión indica que lo que hay antes de la "b", hay que multiplicarlo por 10 elevado al número que hay después (en este caso,1). Se puede cambiar el nº de cifras decimales en **Numérico—>Establecer precisión** (por defecto son 16 cifras decimales).

También podemos poner el programa en modo numérico. Para ello en el menú **Numérico—> Conmutar salida numérica**. Hay que acordarse de volver a cambiarlo si queremos seguir con el cálculo simbólico.

Funciones usuales

Además de las operaciones elementales que hemos visto, *Maxima* tiene definidas la mayor parte de las funciones elementales. Los nombres de estas funciones suelen ser su abreviatura en inglés, que algunas veces difiere bastante de su nombre en castellano.

`sqrt(x)`	raíz cuadrada de x
`exp(x)`	exponencial de x
`log(x)`	logaritmo neperiano de x
`sin(x), cos(x), tan(x)`	seno, coseno y tangente *en radianes*
`csc(x), sec(x), cot(x)`	cosecante, secante y cotangente *en radianes*
`asin(x), acos(x), atan(x)`	arcoseno, arcocoseno y arcotangente
`sinh(x), cosh(x), tanh(x)`	seno, coseno y tangente hiperbólicos
`asinh(x), acosh(x), atanh(x)`	arcoseno, arcocoseno y arcotangente hiperbólicos

```
(%i9) sin(%pi/4);
(%o9) 1/√2
```

Por defecto, las funciones trigonométricas están expresadas en radianes.

```
(%i10) log(20);
(%o10) log(20)
```

y si lo que nos interesa es su expresión decimal

```
(%i11) log(20),numer;
(%o11) 2.995732273553991
```

Observación Mucho cuidado con utilizar `ln` para calcular logaritmos neperianos: ⚠

Insercción de texto

Podemos comentar resultados, explicaciones etc en Maxima. Para ello vamos a **Celda—> Nueva celda de texto** Nos inserta una celda con fondo verde-azulado donde podemos escribir. También podríamos elaborar un documento con secciones y subsecciones donde Maxima nos los numera automáticamente. Para una celda de sección, hay que ir a **Celda—> Nueva celda de sección**

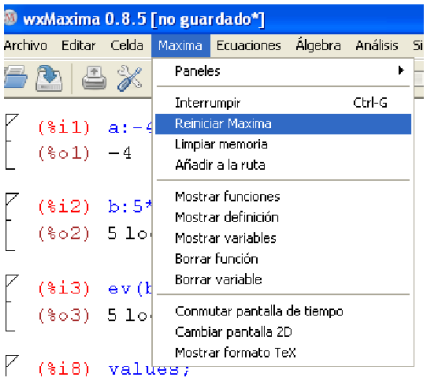
1

También se puede comentar con la secuencia /* comentario */

```
/* No operes 2*4 */;
```

Reinicio de Maxima

A medida que en una sesión de Maxima vamos definiendo variables, funciones, etc. no basta con borrar las celdas donde están definidas, pues continuan vigentes en memoria, pudiendo llegar a obtener resultados extraños debido a que, por ejemplo, a la variable x le habíamos dado un valor previo y no nos acordamos de vaciarla. Por eso, quizás sea conveniente hacer un reinicio de Maxima y se olvide de todo lo anterior. Para ello, vamos a **Maxima—> Reiniciar Maxima.** Luego conviene ir a **Celdas—> Evaluar todas las celdas.** También podemos limpiar memoria **Maxima—> Limpiar memoria** con parecidos resultados




```
(%i12) kill(all);
(%o0) done
```

Variables

El uso de variables es muy fácil y cómodo en *Maxima*. Uno de los motivos de esto es que no hay que declarar tipos previamente. Para asignar un valor a una variable utilizamos los dos puntos

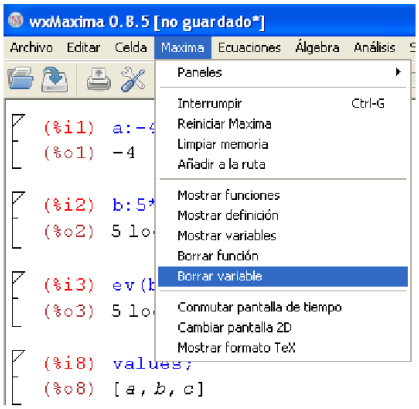
```
(%i1) a:2;
(%o1) 2
(%i2) a^2;
(%o2) 4
```

Evaluación de valores en expresiones

expr, [a_1=*valor1*, a_2=*valor2*···]	En *expr* sustituye a_1 por *valor1*, a_2 por *valor2*, ···

```
(%i3) b:5*log(x)-a^2;
(%o3) 5 log(x)−4
(%i4) b,x=3;
(%o4) 5 log(3)−4
```

Desde el menú de Maxima podemos borrar los valores de cualquier variable o incluso todas. Para ello, vamos a **Maxima** —> **Borrar variables** y en la ventana que nos sale escribimos los nombres de las variables a borrar, separadas por comas. Por defecto, las borra todas. También podes ver todas las variables que hay definidas, en el mismo menú en "Mostrar variables".



Simplificación de expresiones

`radcan(`*expr*`)`	simplifica expresiones con radicales
`ratsimp(`*expr*`)`	simplifica expresiones racionales
`fullratsimp(`*expr*`)`	simplifica expresiones racionales

Para simplificar expresiones racionales, `ratsimp` funciona bastante bien aunque hay veces que es necesario aplicarlo más de una vez. La orden `fullratsimp` simplifica algo mejor a costa de algo más de tiempo y proceso.

```
(%i5) (x+a)*(x-a);
(%o5) (x−2)(x+2)
(%i6) ratsimp((x+a)*(x-a));
(%o6) x^2−4
```

Listas

Maxima tiene una manera fácil de agrupar objetos, ya sean números, funciones, cadenas de texto, etc. y poder operar con ellos. Una lista se escribe agrupando entre corchetes los objetos que queramos separados por comas. Por ejemplo,

```
(%i7) v:[0,1,-3];
(%o7) [0,1,−3]
```

first, second, · · · , last	Primer, segundo,...., último elemento de una lista
lista[k]	k-ésimo elemento de la lista
sort	Ordena los elementos de una lista
length	longitud de la lista

```
(%i8) first(v);
(%o8) 0
(%i9) v[2];
(%o9) 1
```

También podemos construir una lista a partir de una fórmula:

makelist(*expr, var, n, m*)	Construye una lista variando la variable *var* desde *n* hasta *m* con la expresión *expr*

```
(%i10) makelist(k^3,k,1,10);
(%o10) [1,8,27,64,125,216,343,512,729,1000]
```

Vectores

Una lista, también podemos considerar que es un vector. En tal caso, podemos efectuar las operaciones habituales: suma, producto por un escalar y producto escalar.

```
(%i12) v1:[1,0,-1];
       v2:[-2,1,3];
(%o11) [1,0,−1]
(%o12) [−2,1,3]
(%i13) v1.v2;
(%o13) −5
```

NOTA: Para el producto escalar, debemos utilizar "." Si utilizamos " * " nos multiplica término a término (y no lo suma)

```
(%i14) v1*v2;
(%o14) [−2,0,−3]
```

Matrices

Para definir una matriz, lo hacemos con el comando *matrix()* cuyo argumento es una serie de listas, cada una de ellas representa una fila de la matriz.

matrix(*fila1, fila2,* · · ·)	Definir la matriz
rank(*matriz*)	Calcula el rango de la matriz
determinant(*matriz*)	Calcula el determinante de la matriz
invert(*matriz*)	Calcula la matriz inversa.

```
(%i16) A:matrix([1,2,3],[-1,0,3],[2,1,-1]);
       B:matrix([-1,1,1],[1,0,0],[-3,7,2]);
```

$$(\%o15)\begin{pmatrix}1 & 2 & 3\\ -1 & 0 & 3\\ 2 & 1 & -1\end{pmatrix}$$

$$(\%o16)\begin{pmatrix}-1 & 1 & 1\\ 1 & 0 & 0\\ -3 & 7 & 2\end{pmatrix}$$

Podemos efectuar todas las operaciones habituales sobre matrices: sumas, producto por escalares y producto (usando " . ")

```
(%i17) A.B;
```

$$(\%o17)\begin{pmatrix}-8 & 22 & 7\\ -8 & 20 & 5\\ 2 & -5 & 0\end{pmatrix}$$

El operador * multiplica los elementos de la matriz entrada a entrada.

```
(%i18) A*B;
```

$$(\%o18)\begin{pmatrix}-1 & 2 & 3\\ -1 & 0 & 0\\ -6 & 7 & -2\end{pmatrix}$$

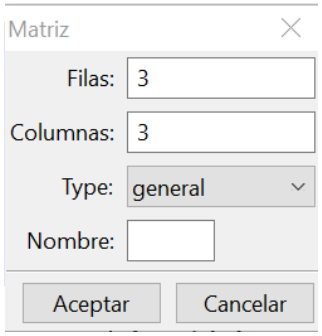
El comando ^^n eleva toda la matriz a *n*, esto es, multiplica la matriz consigo misma *n* veces. Por el contrario el comando ^n eleva cada entrada de la matriz a *n*.

```
(%i20) A^^2;
       A^2;
```

$$(\%o19)\begin{pmatrix}5 & 5 & 6\\ 5 & 1 & -6\\ -1 & 3 & 10\end{pmatrix}$$

$$(\%o20)\begin{pmatrix}1 & 4 & 9\\ 1 & 0 & 9\\ 4 & 1 & 1\end{pmatrix}$$

Existen gran cantidad de comandos para matrices, además de los expuesto anteriormente. Consulte la ayuda de Maxima si fuera necesario. Por supuesto, se puede definir una matriz desde el menú de Maxima, en **Matriz → Introducir Matriz**



Funciones

Para definir una función en *Maxima* se utiliza el operador `:=`. Se pueden definir funciones de una o varias variables, con valores escalares o vectoriales,

`funcion(var1,var2,..):=(expr1,expr2,...)`	definición de función
`define (func,expr)`	la función vale *expr*
`fundef(func)`	devuelve la definición de la función
`functions`	lista de funciones definidas por el usuario
`remfunction(func1,func2,...)`	borra las funciones

```
(%i21) kill(all)$
(%i1) f(x):=sin(x);
(%o1) f(x):=sin(x)
(%i2) f(%pi/4);
(%o2) 1/sqrt(2)
```

También se puede utilizar el comando `define` para definir una función. Por ejemplo, podemos utilizar la función g para definir una nueva función y, de hecho veremos que ésta es la manera correcta de hacerlo cuando la definición involucra funciones previamente definidas, derivadas de funciones, etc. El motivo es que la orden `define` evalúa los comandos que pongamos en la definición.

```
(%i3) define(g(x),cos(x));
(%o3) g(x):=cos(x)
```

Funciones definidas a trozos

Las funciones definidas a trozos plantean algunos problemas de difícil solución para *Maxima*. Esencialmente hay dos formas de definir y trabajar con funciones a trozos:

a) definir una función para cada trozo con lo que tendremos que ocuparnos nosotros de ir escogiendo de elegir la función adecuada, o
b) utilizar una estructura `if-then-else` para definirla.[4]

Cada uno de los métodos tiene sus ventajas e inconvenientes. El primero de ellos nos hace aumentar el número de funciones que definimos, usamos y tenemos que nombrar y recordar. Además de esto, cualquier cosa que queramos hacer, ya sea representar gráficamente o calcular una integral tenemos que plantearlo nosotros. *Maxima* no se encarga de esto. La principal limitación del segundo método es que las funciones definidas de esta manera no nos sirven para derivarlas o integrarlas, aunque sí podremos dibujar su gráfica. Por ejemplo, la función

$$f(x) = \begin{cases} x^2, & \text{si } x < 0 \\ x^3, & \text{en otro caso} \end{cases}$$

la podemos definir de la siguiente forma utilizando el segundo método

```
(%i4) f(x):= if x<0 then x^2 else x^3;
(%o4) f(x):=if x<0 then x^2 else x^3
```

Gráficos en el plano con plot2d

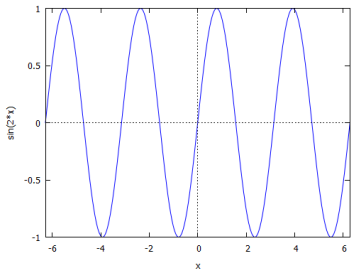
Coordenadas cartesianas

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 **Tramado → Tramado 2d** **plot2d**

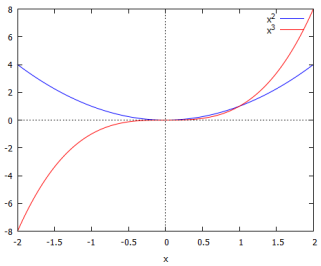
`plot2d(f(x),[x,a,b])`	gráfica de $f(x)$ en $[a,b]$
`plot2d([f1(x),f2(x),...],[x,a,b])`	gráfica de una lista de funciones en $[a,b]$

Por ejemplo:

```
(%i5) plot2d(sin(2*x),[x,-2*%pi,2*%pi])$
```




```
(%i6) plot2d([x^2,x^3],[x,-2,2])$
```

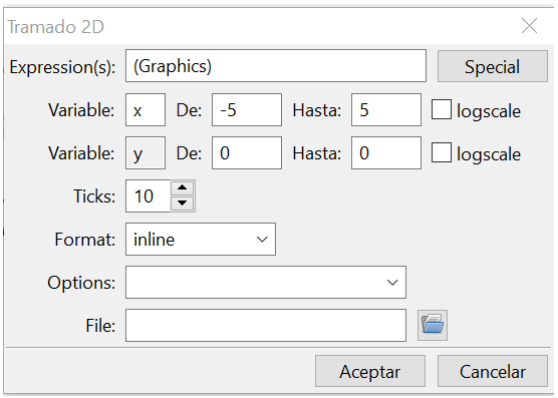


Observa en esta última salida cómo el programa asigna a cada gráfica un color distinto para diferenciarlas mejor y añade la correspondiente explicación de qué color representa a cada función.

Cuando pulsamos el botón **Tramado 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;"`).

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 este capítulo (`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`.

Gráficos con draw

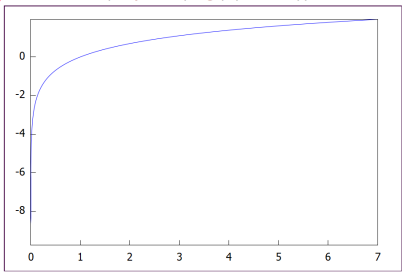
Con la orden draw2d podemos dibujar gráficos en 2 dimensiones con la curva en explícitas, pero también implícitas, polares. etc. Un gráfico está compuesto por varias opciones y el objeto gráfico que queremos dibujar.

`draw2d(opciones, objeto gráfico,...)` dibuja gráfico dos dimensional

Las opciones son numerosas y permiten controlar prácticamente cualquier aspecto imaginable. Aquí comentaremos algunas de ellas pero la ayuda del programa es insustituible. En segundo lugar aparece el objeto gráfico.
Estos pueden ser de varios tipos aunque los que más usaremos son quizás `explicit` y `implicit`.

explicit: Usaremos explicit($f(x),x,a,b$) para dibujar $f(x)$ en $[a, b]$.

```
(%i7) wxdraw2d(explicit(log(x),x,0,7));
```

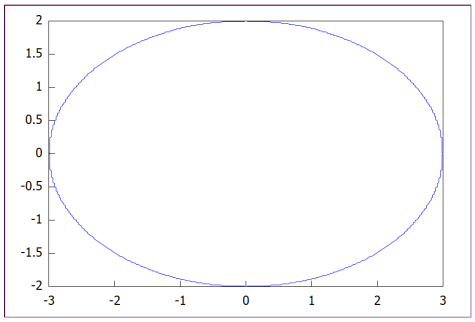


(%t7)

(%o7)

implicit: nos permite dibujar el lugar de los puntos que verifican una ecuación en el plano. Además de la ecuación debemos indicar los intervalos dónde pueden tomar valores las variables.

```
(%i8) wxdraw2d(implicit((x/3)^2+(y/2)^2=1,x,-3,3,y,-2,2))$
```



(%t8)

⚠ **Observación** El comando plot2d no permite dibujar curvas en implícitas y debemos usar en este caso el comando draw2d.

Resolución de ecuaciones

Maxima puede resolver los tipos más comunes de ecuaciones y sistemas de ecuaciones algebraicas de forma exacta. Por ejemplo, sabe encontrar las raíces de polinomios de grado bajo (2,3 y 4). En cuanto a polinomios de grado más alto o ecuaciones más complicadas, no siempre será posible encontrar la solución exacta. En este caso, podemos intentar encontrar una solución aproximada en lugar de la solución exacta.

La orden solve

La orden `solve` nos da todas las soluciones, ya sean reales o complejas de una ecuación o sistema de ecuaciones. **solve**

`solve(`*ecuación*`,`*incógnita*`)`	resuelve la ecuación
`solve([`*ecuaciones*`],[`*variables*`])`	resuelve el sistema
`multiplicities`	guarda la multiplicidad de las soluciones

```
(%i9) solve(x^2+3*x+1=0,x);
```

$$(\%o9)\quad \left[x=-\frac{\sqrt{5}+3}{2},x=\frac{\sqrt{5}-3}{2}\right]$$

La orden `solve` no sólo puede resolver ecuaciones algebraicas.

```
(%i10) solve(sin(x)*cos(x)=0,x);
solve: using arc-trig functions to get a solution.
Some solutions will be lost.
```

$$(\%o10)\quad \left[x=0,x=\frac{\pi}{2}\right]$$

¿Qué ocurre aquí? La expresión $\operatorname{sen}(x)\cos(x)$ vale cero cuando el seno o el coseno se anulen. Para calcular la solución de $\operatorname{sen}(x)=0$ aplicamos la función arcoseno a ambos lados de la ecuación. La función arcoseno vale cero en cero pero la función seno se anula en muchos más puntos. Nos estamos dejando todas esas soluciones y eso es lo que nos está avisando *Maxima*.

También podemos resolver sistemas de ecuaciones. Sólo tenemos que escribir la lista de ecuaciones y de incógnitas. Por ejemplo, para resolver el sistema

$$\left.\begin{aligned} x^2+y^2&=1\\ (x-2)^2+(y-1)^2&=4\end{aligned}\right\}$$

escribimos

```
(%i11) solve([x^2+y^2=1,(x-2)^2+(y-1)^2=4],[x,y]);
```

$$(\%o11)\quad \left[\left[x=\frac{4}{5},y=-\frac{3}{5}\right],[x=0,y=1]\right]$$

Sistemas de ecuaciones lineales

`linsolve([`*ecuaciones*`],[`*variables*`])`	resuelve el sistema

linsolve

En el caso particular de sistemas de ecuaciones lineales puede ser conveniente utilizar `linsolve` en lugar de `solve`. Ambas órdenes se utilizan de la misma forma, pero `linsolve` es más eficiente en estos casos.

```
(%i13) eq:[x+y+z+w=1,x-y+z-w=-2,x+y-w=0]$
       linsolve(eq,[x,y,z]);
```

$$(\%o13)\quad \left[x=\frac{4\,w-3}{2},y=-\frac{2\,w-3}{2},z=1-2\,w\right]$$

Algsys

`algsys([`*ecuaciones*`],[`*variables*`])`	resuelve la ecuación o ecuaciones
`realonly`	si vale `true`, `algsys` muestra sólo soluciones reales

algsys

La orden `algsys` resuelve ecuaciones o sistemas de ecuaciones algebraicas. La primera diferencia con la orden `solve` es pequeña: `algsys` siempre tiene como entrada listas, en otras palabras, tenemos que agrupar la ecuación o ecuaciones entre corchetes igual que las incógnitas.

La segunda diferencia es que `algsys` intenta resolver numéricamente la ecuación si no es capaz de encontrar la solución exacta.

```
(%i15) eq:x^6+x+1;
       solve(eq,x);
```

$$(\%o14)\quad x^6+x+1$$

$$(\%o15)\quad \left[0=x^6+x+1\right]$$

```
(%i17) algsys([eq],[x])$
       transpose(%);
```

$$(\%o17)\quad \begin{pmatrix} [x=-1.03838075445846\,\%i-0.1547351444968429] \\ [x=1.03838075445846\,\%i-0.1547351444968429] \\ [x=-0.3005069203095515\,\%i-0.7906671888144177] \\ [x=0.3005069203095515\,\%i-0.7906671888144177] \\ [x=0.9454023333112606-0.6118366937810087\,\%i] \\ [x=0.6118366937810087\,\%i+0.9454023333112606] \end{pmatrix}$$

El teorema de los ceros de Bolzano

Uno de los primeros resultados que aprendemos sobre funciones continuas es que si cambian de signo tienen que valer cero en algún momento.

El comando find_root encuentra una solución de una función (ecuación) continua que cambia de signo.

`find_root (f(x),x,a,b)` solución de f en $[a,b]$

```
(%i18) f(x):=exp(x)+log(x);
(%o18) f(x):=exp(x)+log(x)
```

Buscamos un par de puntos donde cambie de signo

```
(%i20) f(1),numer;
       f(%e^-3),numer;
(%o19) 2.718281828459045
(%o20) -1.948952728663784
```

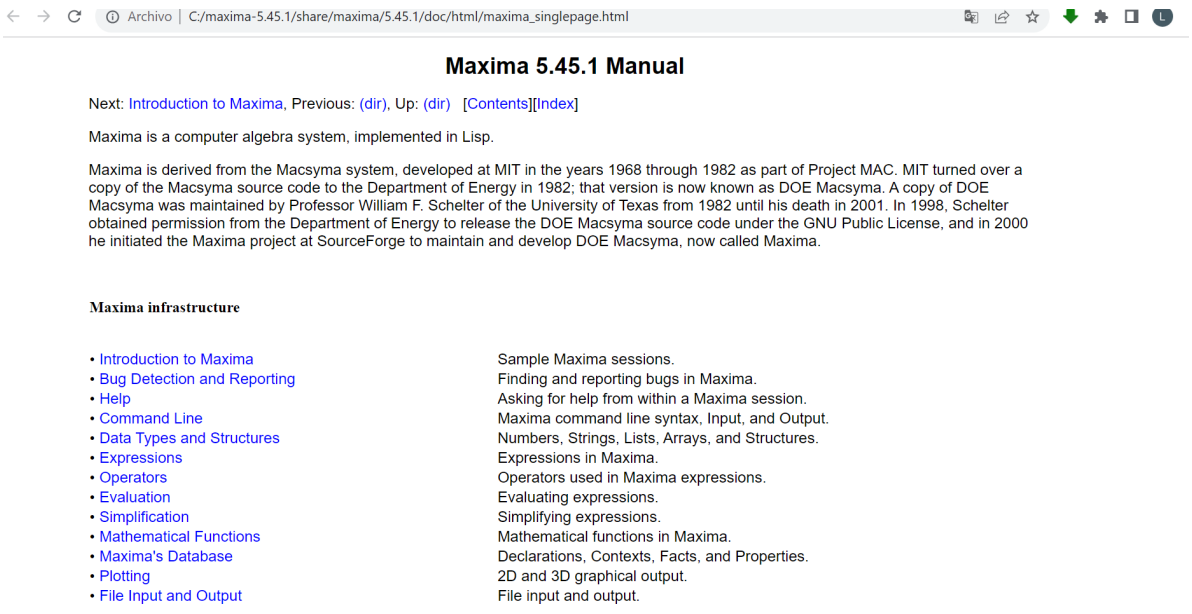
Vale, ya que tenemos dos puntos donde cambia de signo podemos utilizar `find_root`:

```
(%i21) find_root(f(x),%e^-3,1);
(%o21) 0.2698741375734492
```

⚠ **Observación** Este método encuentra *una* solución pero no nos dice cuántas soluciones hay.

La ayuda de *Maxima*

El entorno *wxMaxima* permite acceder a la amplia ayuda incluida con *Maxima* de una manera gráfica.



EJERCICIOS PROPUESTOS

1) Calcular $1 + \frac{1}{9} + 3^{2+4}$.

2) Calcular $\sqrt{4 + \sqrt{144}} + \sqrt[3]{27}$.

3) Calcular $\sqrt{1 + \sqrt{4}} + \sqrt{2}$.

4) Cacular $e^{\pi i}$.

5) Dar una expresión aproximada de $\sqrt[5]{\pi}$.

6) Calcular $\text{sen}\frac{\pi}{4} + \cos\frac{\pi}{2} + \ln e^4$.

7) Calcular $arctg1 + ar\cos(-1)$.

8) Asignar al símbolo *pepe* el valor de 100! y calcular $\frac{pepe}{95!}$.

9) Sustituir en la expresión $pepe + \cos^2 x + sen^2 x^2$ la variable x por 0.

10) Sustituir en la expresión $(x + y)^2 - x^3$ la variable x por 1 y la variable y por -1.

11) Simplificar la expresión $(x + y)(x - y) - x^2$.

12) Construir con *makelist* una lista con los 30 primeros números impares y de manera que vayan alternando de signo.

13) Dada la matriz

$$A = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

Calcular $(I + A)^3$ y la transpuesta de $(I + A)^3$, siendo I la matriz identidad.

14) Definir la función $e + x + x^{10}$ y calcular $\text{f}(\frac{1}{2})$

15) Construir las funciones compuestas $f \circ g$ en los casos siguientes:
a) $f(x) = sen(x)$; $g(x) = 1 - x^2$

b) $f(u) = \frac{u-1}{u+1}$; $g(u) = \frac{u+1}{1-u}$

c) $f(x) = \frac{1}{x}$; $g(x) = tg(x)$

16) Representar en una misma gráfica las funciones seno y coseno en el intervalo $[-2\pi, 2\pi]$, de tal forma que una sea en color rojo, la otra en azul y con grosores distintos.

17) Definir y representar gráficamente la función:

$$f(x) = \begin{cases} e^{x+1} & si \quad 0 \leq x < 1 \\ \ln(1+x^2) & si \quad x \geq 1 \end{cases}$$

18) Definir y representar gráficamente la siguiente función:

$$f(x) = \begin{cases} 1 & si \quad x < 0 \\ 1 - x^2 & si \quad 0 \leq x \leq 2 \\ -3 & si \quad x > 2 \end{cases}$$

19) Hallar las raices de las siguientes ecuaciones y el orden de multiplicidad:
a) $x^3 + 3x^2 + 3x + 1 = 0$

b) $x^4 + x^2 + 3x + 1 = 0$

20) Resolver el sistema de ecuaciones: $\begin{cases} x^2 - y^2 = 1 \\ x + y + z = 0 \\ y^2 - z^2 = 0 \end{cases}$

21) Hallar las raices de la ecuación: $e^x + 1 = tg(x)$