BREVE INTRODUCCIÓN A MAXIMA

1. Introducción

MAXIMA es un sistema para la manipulación de expresiones simbólicas y numéricas, incluyendo diferenciación, integración, expansión en series de Taylor, transformadas de Laplace, ecuaciones diferenciales ordinarias, sistemas de ecuaciones lineales, y vectores, matrices y tensores. MAXIMA produce resultados con alta precisión usando fracciones exactas y representaciones con aritmética de coma flotante arbitraria. Adicionalmente puede representar gráficamente funciones y datos en dos y tres dimensiones.

El código fuente de MAXIMA puede ser compilado sobre varios sistemas incluyendo Windows, Linux y MacOS X. El código fuente para todos los sistemas y los binarios precompilados para Windows y Linux están disponibles en el Administrador de archivos de SourceForge (`http://sourceforge.net/projects/maxima/files/`).

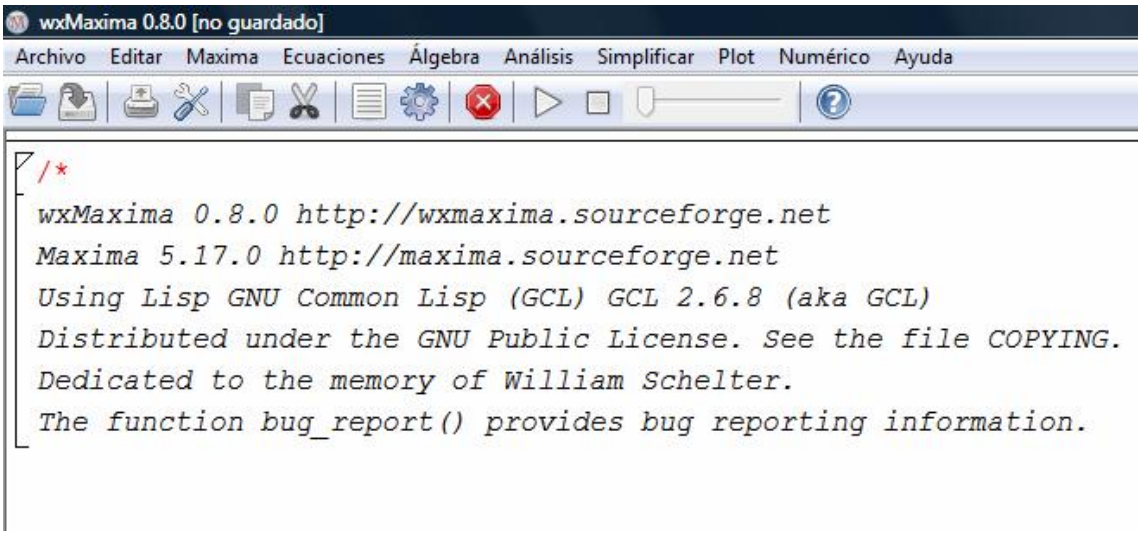
MAXIMA es un descendiente de Macsyma, un sistema de álgebra computacional desarrollado a finales de 1960 en el instituto tecnológico de Massachusetts (MIT). Está basado en el esfuerzo voluntario de una comunidad de usuarios activa, gracias a la naturaleza del *open source*. Macsyma fue revolucionario es sus días y muchos sistemas posteriores, tales como Maple y Mathematica, estuvieron inspirados en él.

La rama MAXIMA de Macsyma fue mantenida por William Schelter desde 1982 hasta su muerte en 2001. En 1998 él obtuvo permiso para liberar el código fuente bajo la licencia pública general (GPL) de GNU.

MAXIMA está en constante actualización, corrigiendo y mejorando el código y la documentación. Hay una lista de correo de MAXIMA para hacer sugerencias y consultar dudas (`http://maxima.sourceforge.net/maximalist.html`)

2. Organización del trabajo y uso como calculadora científica

Se inicia MAXIMA haciendo clic sobre el icono de acceso directo una vez instalado el programa. Al abrirse, MAXIMA despliega (no necesariamente en todas las versiones) alguna información importante acerca de la versión que se está usando y un prompt.



Si se teclea la operación a realizar (por ejemplo, $2+3$) y se pulsa la tecla *shift* y *enter* simultáneamente, aparecerá en pantalla el resultado

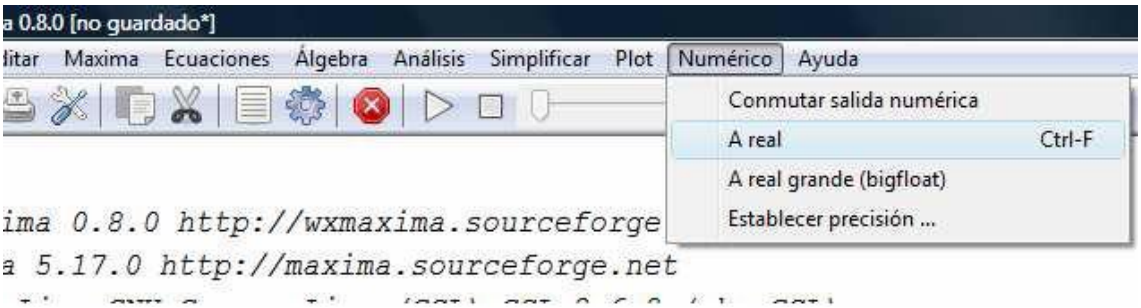
```
(%i1)  2+3;
(%o1)  5
```

Las entradas van numeradas por orden de actuación `(%in)` `(input)` y la salida correspondiente lleva la misma numeración `(%on)` `(output)`. Esta numeración facilita hacer referencia a resultados anteriores cuando se han realizado ya varias operaciones.

En el ejemplo anterior ($2+3=5$) se utiliza MAXIMA como una simple calculadora. Los símbolos de las operaciones básicas son los habituales para el cociente, producto y potencia:

```
(%i1)  2+3;
(%o1)  5
(%i2)  2*3;
(%o2)  6
(%i3)  2^3;
(%o3)  8
(%i4)  2/3;
(%o4)  2/3
```

Para obtener la aproximación decimal de un resultado se selecciona en la barra de herramientas superior Numérico A Real, tal y como se indica en la figura:



O bien se usa la orden `float(expr)`:

```
(%i5)  float(2/3);
(%o5)  0.66666666666667
```

Siempre que sea necesario se introducen paréntesis para que las operaciones se realicen en el orden correcto. Por ejemplo:

```
(%i8)  2+3^2;
(%o8)  11
(%i9)  (2+3)^2;
(%o9)  25
```

Para hacer referencia a un resultado anterior se utiliza el símbolo %in ó %on. Por ejemplo, en nuestro caso, %i8 es 11, de modo que:

```
(%i11)  %i8-4;
(%o11)  7
```

Para hacer referencia al resultado inmediatamente anterior basta escribir %, sin ningún número; en nuestro caso el último input ha sido (%i11), de modo que:

```
(%i12)  %^2;
(%o12)  49
```

También es posible operar con números complejos, teniendo en cuenta que la unidad imaginaria i habrá que introducirla como %i (no confundir con la numeración de las entradas):

```
(%i25)  (2*%i)*%i;
(%o25)  -2
```

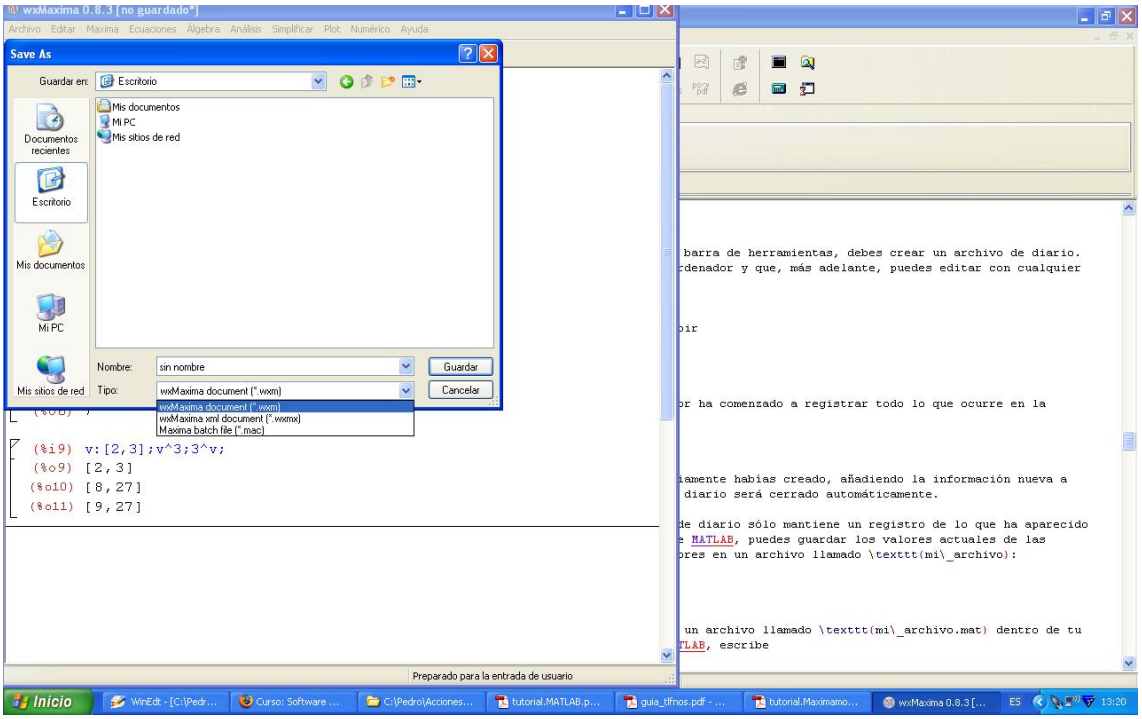
Estas son algunas de las funciones que están definidas en el programa:

sqrt(x)	raíz cuadrada		$n!$	factorial
exp(x)	exponencial(e^x)		abs(x)	valor absoluto
`sin(x)`	seno		`asin(x)`	arco seno
`cos(x)`	coseno		`acos(x)`	arco coseno
`tan(x)`	tangente		`atan(x)`	arco tangente
log (x)	logaritmo neperiano			

MAXIMA manipula constantes numéricas, por ejemplo, el número e se escribe `%e` y el número π se escribe `%pi`.

```
(%i18) log(%e);
(%o18) 1
(%i22) sin(%pi/2);
(%o22) 1
(%i32) atan(1);
(%o32) %pi/4
```

Si deseas guardar el trabajo que realizas en una sesión MAXIMA puedes hacerlo a través de la barra de herramientas. Es posible guardarlo con la extensión .wxm o .mac. En cualquier caso, los archivos contendrán una transcripción completa y literal de todos las órdenes que teclean en el ordenador y que no son borradas durante la sesión. En el caso de los archivos .vxm y .max, se pueden editar con cualquier editor de texto. También podrás ejecutarlos de nuevo con MAXIMA abriendo el archivo correspondiente con la barra de herramientas. Al salir de una sesión de MAXIMA o reiniciar el programa, los resultados obtenidos y valores de todas las variables se pierden.



Ejercicio 1.

a. En el directorio Mis documentos crea una carpeta con el nombre *Matematicas I*.

b. Abre una sesión en MAXIMA y ejecuta alguna operación. Guarda el archivo en la carpeta *Matematicas I* poniéndole el nombre y primer apellido de las personas que componen tu grupo de trabajo, por ejemplo *miguel macias antonio gutierrez*.

c. Cierra la sesión y vuelve a cargar el archivo. Ejecuta de nuevo las operaciones.

Ejercicio 2. Obtén el resultado exacto y decimal aproximado:

$$\frac{1+\sqrt{2}}{1-\sqrt{5}}, \quad \cos^2(\pi), \quad e^4\,34^7$$

Ejercicio 3. Calcula la forma binómica de

$$(3+2i)*(5-i)^4$$

.

3. Variables y funciones

Se puede asignar un valor a una variable. La sintaxis es:

`nombre: expresión matemática`

Por ejemplo, asignamos a x el valor $2*3$, a y el valor 2 y posteriormente calculamos xy:

```
(%i15) x:3*2; y:2;
(%o15) 6
(%o16) 2

(%i17) x; y; x*y;
(%o17) 6
(%o18) 2
(%o19) 12
```

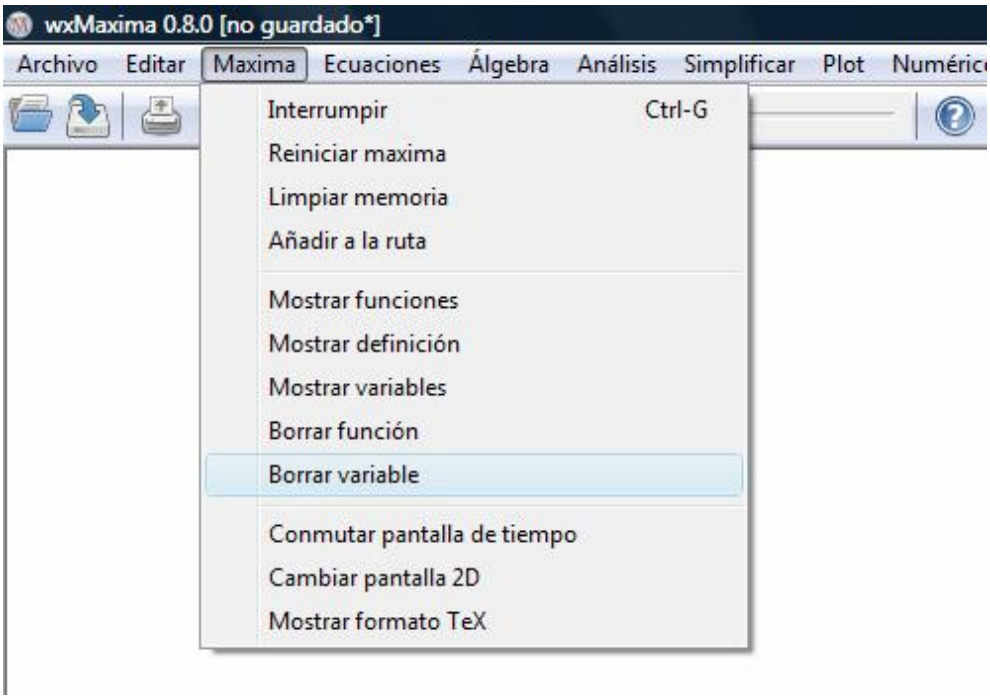
Podemos pedir a MAXIMA que desarrolle, por ejemplo, la expresión $(x+3y)^5$ (con la orden `expand(`$(x+3y)^5$`)`), y también podemos asignar el resultado obtenido a una variable (darle un nombre)

```
(%i13) h:expand((x+3*y)^5);
(%o13) 243 y^5 + 405 x y^4 + 270 x^2 y^3 + 90 x^3 y^2 + 15 x^4 y + x^5
```

Podemos referirnos a esa expresión con el nombre dado, de modo que, por ejemplo:

```
(%i14) h/(45+z);
                    243 y^5 + 405 x y^4 + 270 x^2 y^3 + 90 x^3 y^2 + 15 x^4 y + x^5
(%o14)  ------------------------------------------------------------------------------
                                          z + 45
```

Para borrar el valor asignado a la variable seleccionamos en la barra de herramientas:



si ahora escribimos el nombre de la variable el resultado es:

```
(%i16)  h;
(%o16)  h
```

Hay que notar que al seleccionar borrar variable aparece por defecto "`all`". Es necesario escribir el nombre de la variable que se desea borrar si no se quiere borrar todas.

Es importante notar:

- que el argumento de las funciones siempre se escribe entre paréntesis;
- por defecto las funciones trigonométricas operan en radianes

Para definir una función propia f de variable x se utiliza "`:=`". Por ejemplo:

```
(%i17)  f(x):=x^2;
(%o17)  f(x):=x^2
```

de modo que:

```
(%i18)  f(3);
(%o18)  9
```

Los nombres asignados a funciones se pueden borrar con la opción "borrar función"de forma análoga a como se borran variables.

Ejercicio 4. Asignando a $\frac{e^{1+i}-e^{-1-i}}{2i}$ y a $\frac{e^{1+i}+e^{-1-i}}{2}$ sendos nombres de variables, calcula

$$\left(\frac{e^{1+i}-e^{-1-i}}{2i}\right)^4+\left(\frac{e^{1+i}+e^{-1-i}}{2}\right)^4+2\left(\frac{e^{1+i}-e^{-1-i}}{2i}\right)^2\left(\frac{e^{1+i}+e^{-1-i}}{2}\right)^2.$$

Ejercicio 5. Define la función $g(x) := \text{sen}(x) * x^2 + 3 * x^7 - 78$ y calcula el valor decimal de $g(3)$.

4. Cálculo simbólico

Una de las mayores utilidades de MAXIMA es que permite realizar no sólo cálculo numérico ($3 + 62 - 1 = 64$), sino también cálculo simbólico ($3x–x + 2 = 2 + 2x$). Veamos algunos comandos para operar con expresiones algebraicas:

`expand(expresión)`	Desarrolla los productos y potencias que figuran en "expresión"
`factor(expresión)`	Factoriza "expresión"
`ratsimp(expresión)`	Simplificación racional de "expresión"

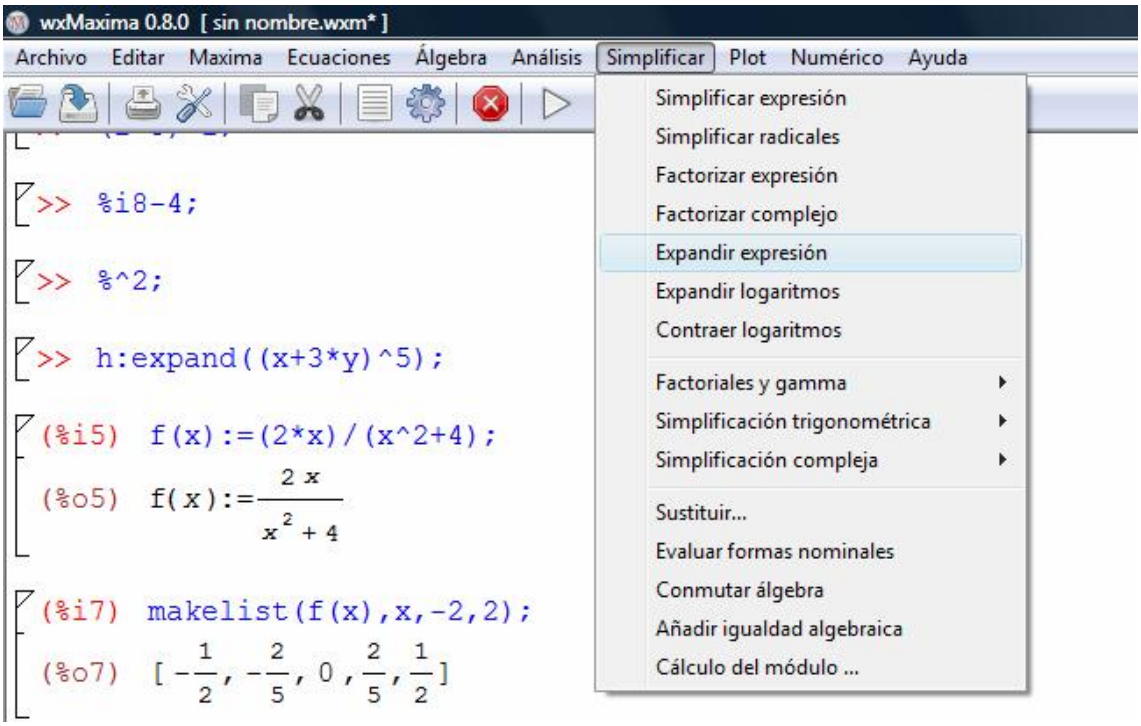
Ejemplos:

```
(%i1) expand((a+b)^3);
(%o1) b^3+3 a b^2+3 a^2 b+a^3

(%i2) factor(x^2-4*x+4);
(%o2) (x-2)^2

(%i3) ratsimp((1/(x+2))+(1/(x-2)));
(%o3) 2 x/(x^2-4)
```

Estas operaciones también pueden hacerse a partir de la barra de herramientas



MAXIMA permite calcular límites de funciones utilizando el comando

```
limit(expr, x, val, dir)
```

que calcula el límite de “expr” cuando la variable “x” tiende al valor “val” desde la dirección “dir” (`plus`: por la derecha, `minus`: por la izquierda). Por ejemplo:

```
(%i9)  limit(1/x, x, 0);
(%o9)  infinity

(%i10)  limit(1/x, x, 0, minus);
(%o10)  -∞

(%i11)  limit(1/x, x, 0, plus);
(%o11)  ∞
```

Si el límite no existe la respuesta es ”`und`”, es decir, indeterminado. Por ejemplo

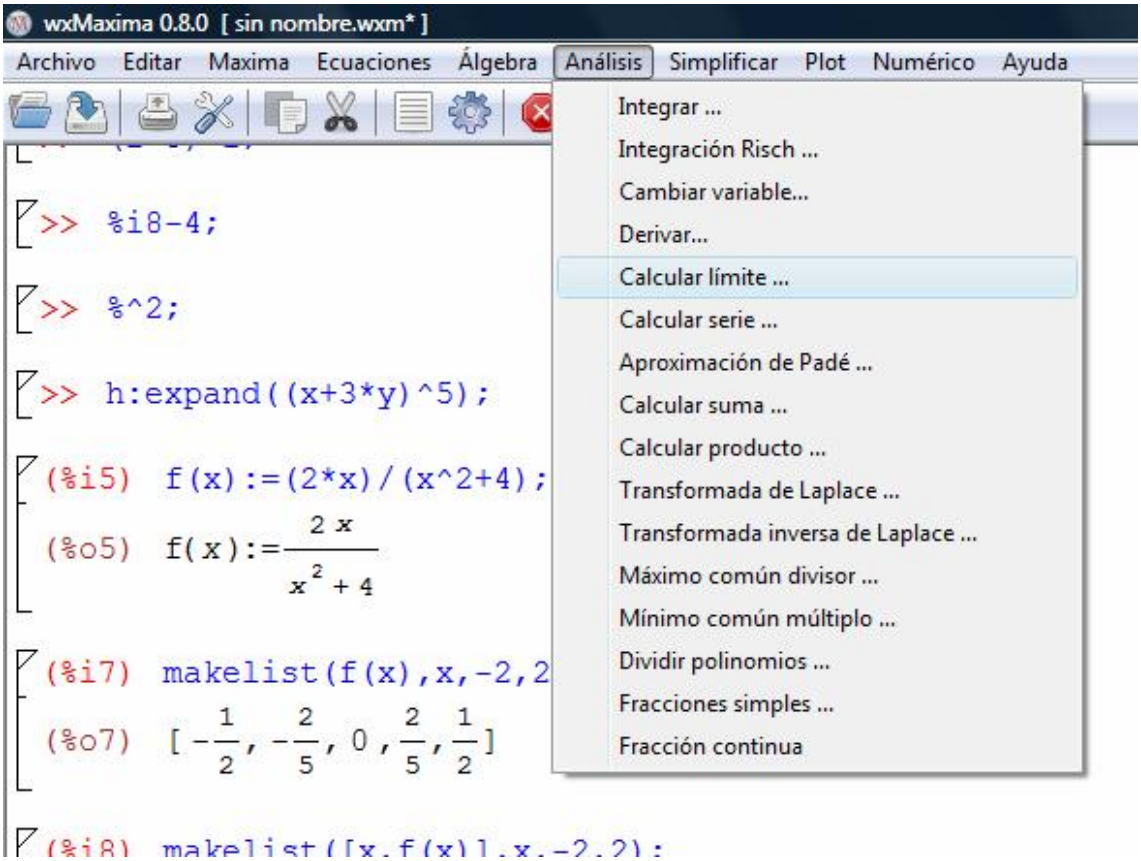
```
(%i12)  f(x):=x/abs(x);
(%o12)  f(x):=x/|x|

(%i13)  limit(f(x),x,0);
(%o13)  und

(%i14)  limit(f(x),x,0,plus);
(%o14)  1

(%i15)  limit(f(x),x,0,minus);
(%o15)  -1
```

También se puede calcular el límite utilizando la barra de herramientas



Ejercicio 6. Calcula el límite de la función $\frac{e^{1/x}}{x}$ cuando x tiende a 0.

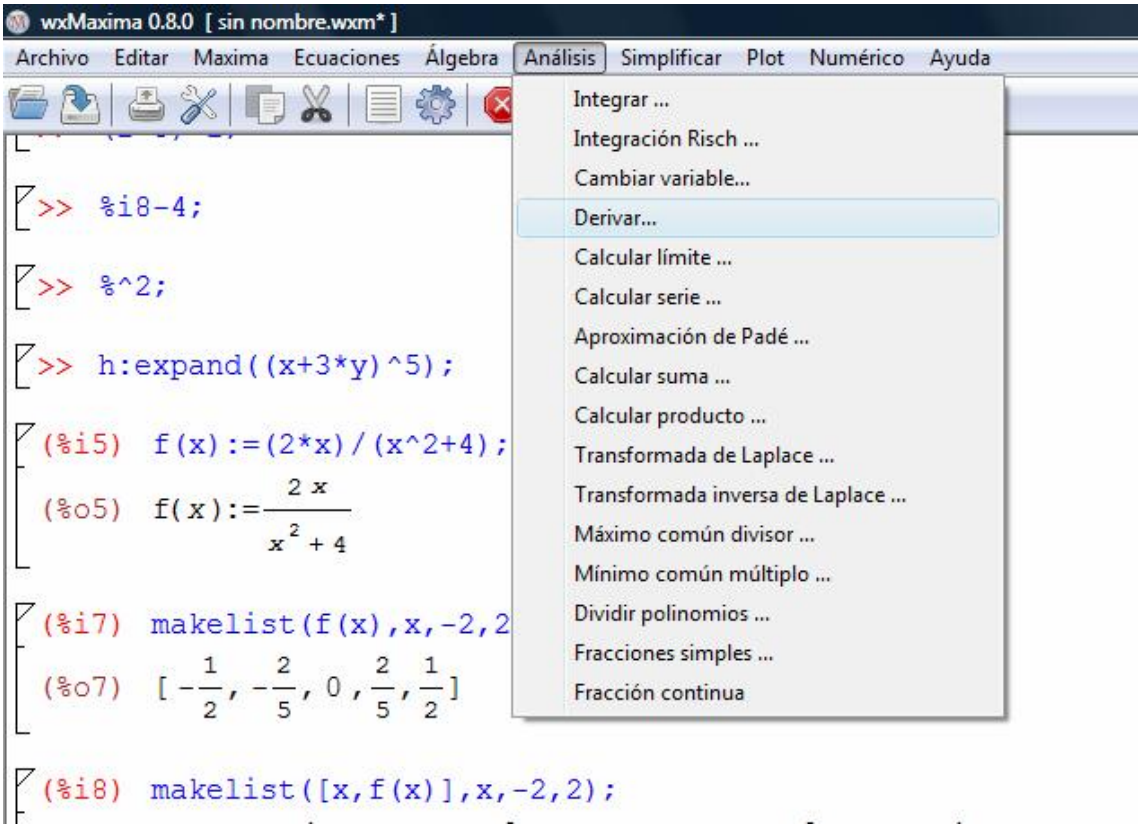
Para derivar funciones se utiliza el comando `diff(expr, x, n)` que deriva "expr" respecto a la variable x, hasta orden n.

```
(%i16) diff(sin(x),x,1);
(%o16) cos(x)

(%i17) diff(atan(x),x,1);
(%o17) 1/(x^2+1)

(%i18) diff(atan(x),x,2);
(%o18) -2*x/(x^2+1)^2
```

O directamente a partir de la barra de herramientas

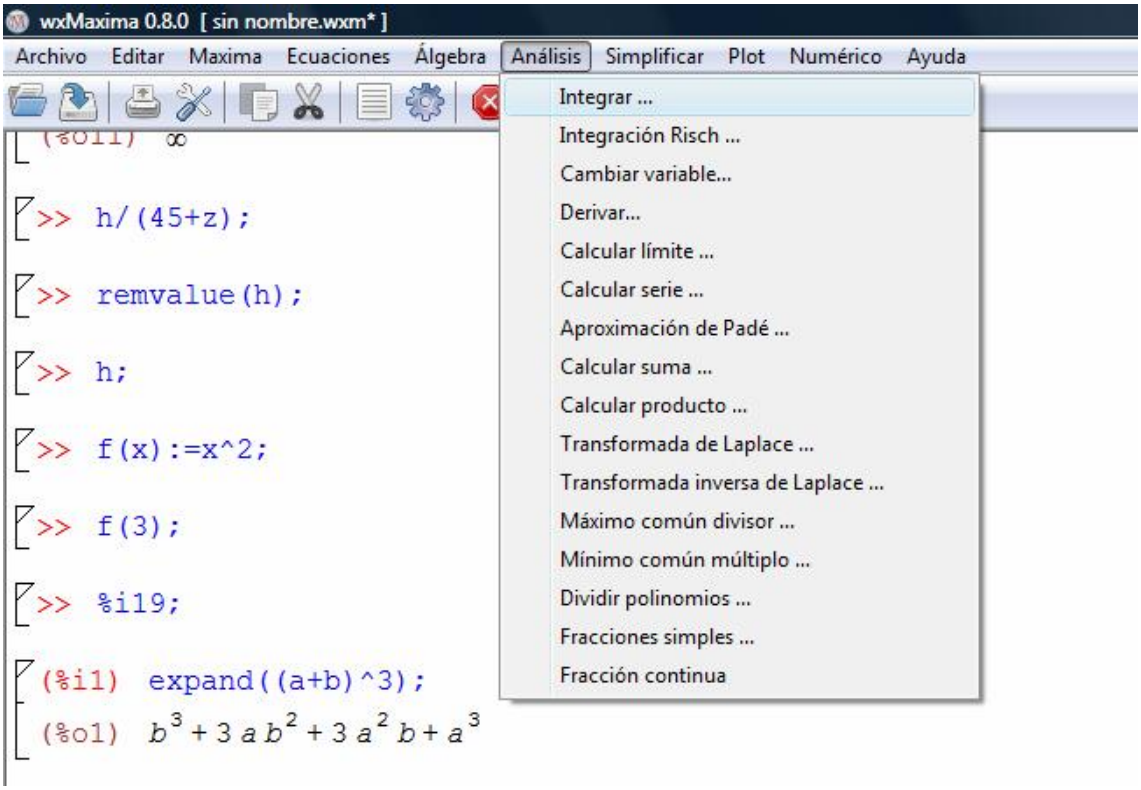


Para integrar se utiliza el comando `integrate(expr,x)` y si se trata de una integral definida se especifican los límites de integración `integrate(expr,x,a,b)`. Por ejemplo:

```
(%i14) integrate(sin(x)*cos(x), x);
                              cos(x)^2
(%o14)                      - --------
                                2

(%i17) integrate(sin(x)*cos(x), x, 0, %pi/2);
                                1
(%o17)                          -
                                2
```

O directamente a partir de la barra de herramientas:



Ejercicio 7. Calcula la función derivada y la integral de $e^{-x}\cos(x)$.

5. Vectores, matrices y funciones con matrices

Para manejar un vector, se introducen entre corchetes sus coordenadas separadas por comas. Las operaciones básicas (suma de vectores, resta de vectores y multiplicación de un vector por un escalar) se realizan elemento a elemento. El producto escalar se designa con un punto. Algunos ejemplos:

```
(%i27)  [2,3]+[1,0];
(%o27)  [3,3]

(%i28)  [3,2,1].[-1,0,5];
(%o28)  2

(%i29)  u:[a,b];
(%o29)  [a,b]

(%i30)  v:[c,d];
(%o30)  [c,d]

(%i31)  u-v;
(%o31)  [a-c,b-d]

(%i32)  u.v;
(%o32)  bd+ac
```

La operación "*elevar a*"se lleva a cabo también elemento a elemento si los operandos son un escalar y un vector o un vector y un escalar.

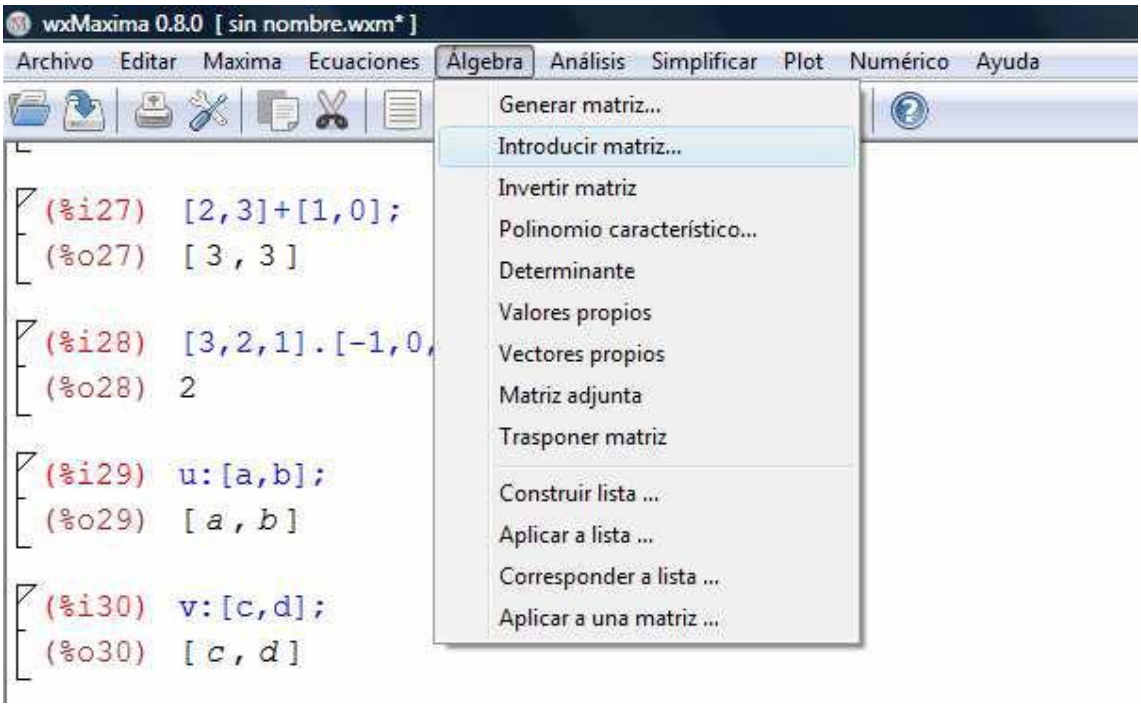
```
(%i9)  v:[2,3];v^3;3^v;
(%o9)  [2,3]
(%o10)  [8,27]
(%o11)  [9,27]
```

Las matrices se introducen con el comando `matrix()`, cuyo argumento son los elementos de cada fila entre corchetes y separados por comas. Por ejemplo:

```
(%i2)  matrix([1,2,3],[4,5,6]);
```

$$(\%o2)\quad \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

También se pueden introducir a partir de la barra de herramientas:



Para generar matrices diagonales podemos usar los comandos, `diagmatrix(n,x)` que devuelve la matriz diagonal de orden n y elemento x, o bien `ident(n)` devuelve la matriz identidad de orden n.

```
(%i49) diagmatrix(3,25);
                [ 25  0   0  ]
(%o49)          [  0  25  0  ]
                [  0  0   25 ]

(%i62) diagmatrix(4,k);
                [ k  0  0  0 ]
                [            ]
                [ 0  k  0  0 ]
(%o62)          [            ]
                [ 0  0  k  0 ]
                [            ]
                [ 0  0  0  k ]

(%i63) ident(4);
                [ 1  0  0  0 ]
                [            ]
                [ 0  1  0  0 ]
(%o63)          [            ]
                [ 0  0  1  0 ]
                [            ]
                [ 0  0  0  1 ]
```

Las operaciones básicas (suma de matrices, resta de matrices y multiplicación de una matriz por un escalar) se llevan a cabo elemento a elemento. Si $\lambda \in \mathbb{R}$ y A es una matrix, las operaciones λ^A y A^λ se calculan elemento a elemento. El producto matricial de dos matrices A y B se obtiene como $A.B$. Dada la matriz A, el producto $A.A$ se obtiene como $A\text{^^}2$ y la inversa, si existe, como $A\text{^^}(-1)$.

```
(%i20) x:matrix([17,3],[-8,11]);
(%o20) [ 17   3 ]
       [ -8  11 ]

(%i24) y:matrix([%pi,%e],[a,b]);
(%o24) [ %pi  %e ]
       [  a    b ]
```

Operaciones básicas:

```
(%i25) x+y;
(%o25) [ %pi + 17  %e + 3 ]
       [   a - 8     b + 11 ]

(%i26) x-y;
(%o26) [ 17 - %pi  3 - %e ]
       [  - a - 8   11 - b ]

(%i27) x*y;
(%o27) [ 17 %pi  3 %e ]
       [ - 8 a   11 b ]

(%i28) x^3;
(%o28) [ 4913   27   ]
       [ - 512  1331 ]

(%i31) 3^y;
(%o31) [ 3^%pi  3^%e ]
       [  3^a    3^b ]
```

Multiplicación de matrices:

```
(%i32) x.y;
(%o32) [ 3 a + 17 %pi   3 b + 17 %e ]
       [ 11 a - 8 %pi   11 b - 8 %e ]

(%i33) y.x;
(%o33) [ 17 %pi - 8 %e   3 %pi + 11 %e ]
       [   17 a - 8 b       11 b + 3 a ]
```

Una matriz elevada al exponente -1 con el operador de potencia no conmutativa equivale a la matriz inversa, si existe:

```
(%i34)  x^^(-1);
                [ 11/211   -3/211 ]
(%o34)          [                 ]
                [ 8/211    17/211 ]

(%i35)  x.x^^(-1);
                [ 1  0 ]
(%o35)          [      ]
                [ 0  1 ]
```

Para seleccionar elementos de una matriz

```
(%i46)  row(A,3);
(%o46)  [2  6  4  -2  0]
```

`row(A,i)` devuelve la fila i de la matriz A.

```
(%i47)  col(A,5);
                [ 0 ]
                [   ]
                [ 4 ]
(%o47)          [   ]
                [ 0 ]
                [   ]
                [ 4 ]
```

`col(A,i)` devuelve la columna i de la matriz A.

`submatrix(`i_1`, ..., `i_m`, A, `j_1`, ..., `j_n`)` devuelve una matriz en la que se han eliminado las filas i_1, ...,i_m de la matriz A y columnas j_1, ..., j_n.

```
(%i48)  submatrix(1,4,A,1,2,3);
                [ 6   4 ]
(%o48)          [       ]
                [ -2  0 ]
```

Otros comandos importantes:

`mat_trace(M)` calcula la traza de la matriz M.

`minor(M, i, j)` calcula el menor ij de la matriz M.

`rank(M)` calcula el rango de la matriz M.

`transpose(M)` calcula la matriz transpuesta de la matriz M.

`triangularize(M)` devuelve la forma triangular superior de la matriz M, obtenida por eliminación gaussiana.

```
(%i42) A:matrix([1,3,2,-1,0],[2,7,-5,6,4],[2,6,4,-2,0],[3,10,-3,5,4]);
```

$$(\%o42)\quad \begin{bmatrix} 1 & 3 & 2 & -1 & 0 \\ 2 & 7 & -5 & 6 & 4 \\ 2 & 6 & 4 & -2 & 0 \\ 3 & 10 & -3 & 5 & 4 \end{bmatrix}$$

```
(%i43) transpose(A);
```

$$(\%o43)\quad \begin{bmatrix} 1 & 2 & 2 & 3 \\ 3 & 7 & 6 & 10 \\ 2 & -5 & 4 & -3 \\ -1 & 6 & -2 & 5 \\ 0 & 4 & 0 & 4 \end{bmatrix}$$

```
(%i44) rank(A);
(%o44) 2

(%i45) triangularize(A);
```

$$(\%o45)\quad \begin{bmatrix} 1 & 3 & 2 & -1 & 0 \\ 0 & 1 & -9 & 8 & 4 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

`eigenvalues(M)` devuelve una lista con dos sublistas. La primera sublista la forman los valores propios de la matriz M y la segunda sus multiplicidades correspondientes.

`eigenvectors(M)` devuelve una lista de listas, la primera de las cuales es la salida de `eigenvalues` y las siguientes son los vectores propios de la matriz asociados a los valores propios correspondientes.

```
(%i56) A:matrix([1,0,0],[0,1,0],[0,0,4]);
(%o56) [1 0 0]
       [0 1 0]
       [0 0 4]

(%i57) eigenvalues(A);
(%o57) [[1,4],[2,1]]

(%i58) eigenvectors(A);
(%o58) [[[1,4],[2,1]],[1,0,0],[0,1,0],[0,0,1]]
```

Para generar una tabla de valores se utiliza el comando makelist:

`makelist(expr, i,` i_0`,` i_1`)` : devuelve una lista con los valores que va tomando "expr" para cada valor de i comprendido entre i_0 e i_1, a saltos de tamaño 1.

`makelist([expr1,expr2], i,` i_0`,` i_1`)`: devuelve una lista con parejas de los valores que toma "expr1" y "expr2".

Por ejemplo, dada una función $f(x)$

```
(%i5) f(x):=(2*x)/(x^2+4);
(%o5) f(x):=2 x/(x^2+4)
```

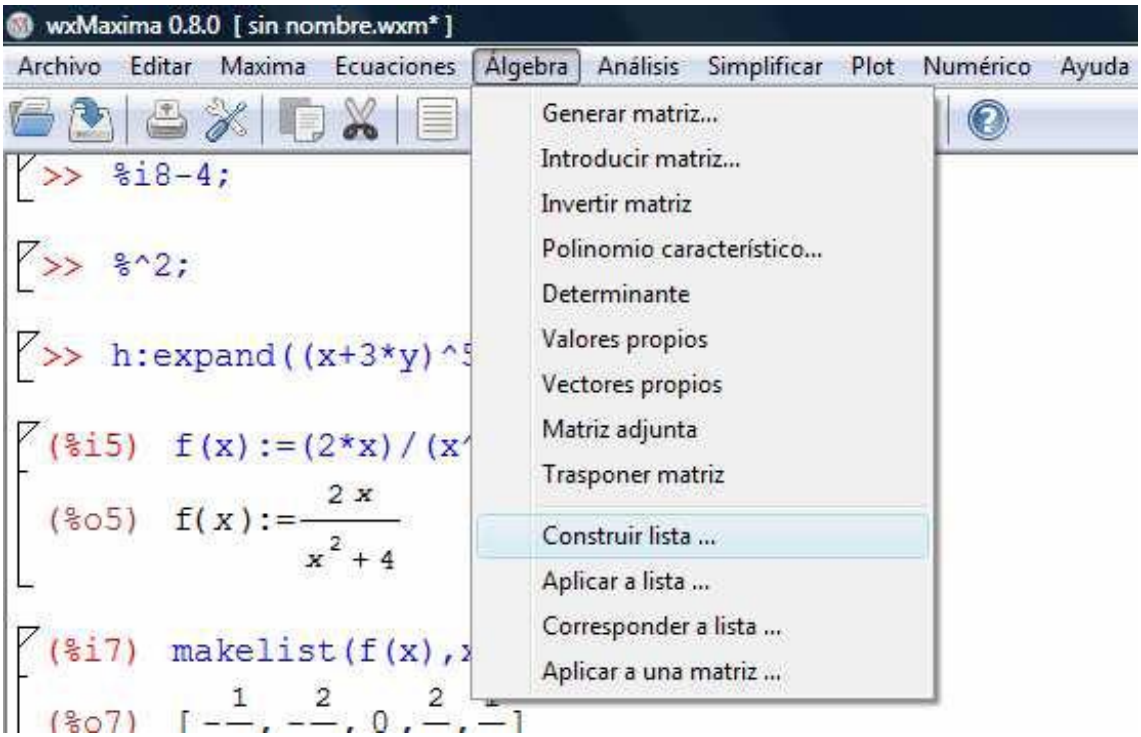
se puede calcular

```
(%i7) makelist(f(x),x,-2,2);
(%o7) [-1/2,-2/5,0,2/5,1/2]
```

También se puede calcular:

```
(%i8) makelist([x,f(x)],x,-2,2);
(%o8) [[-2,-1/2],[-1,-2/5],[0,0],[1,2/5],[2,1/2]]
```

Es posible introducir una lista a partir de la barra de herramientas



Aparece una ventana donde se introduce la expresión en función de una variable. Por ejemplo:

y el resultado es

```
(%i3) makelist(k^2+1, k, 1, 10);
(%o3) [2,5,10,17,26,37,50,65,82,101]
```

Si no se desea que la variable tome valores de 1 en 1 hay que escribir explícitamente la lista de valores que toma, como un argumento del comando `makelist`

```
(%i6)  makelist (k^2+1,k,[2,4,6]);
(%o6)  [5,17,37]
```

o introducir previamente la lista de valores que tomará y hacer referencia a ella con el nombre que se le haya dado. Por ejemplo

```
(%i10)  salto:makelist(k/2,k,3,6);
(%o10)  [1.5,2,2.5,3]

(%i11)  makelist(k^2+1,k,salto);
(%o11)  [3.25,5,7.25,10]
```

Ejercicio 8.

a. Crear un vector con los números naturales del 1 al 10.
b. Calcular la raíz cuadrada de cada uno de esos números.
c. Crear un vector con los números naturales del 1 al 1000.
d. Obtener un vector con el cociente de dividir cada número natural entre 1 y 1000 entre 7.

Ejercicio 9. Calcula la inversa de la matriz

$$A = \begin{pmatrix} 4 & 1 & 2 \\ 1 & 5 & 0 \\ 2 & 0 & 7 \end{pmatrix}$$

y el determinante de al matriz B obtenida de A tomando sus filas $1, 3$ y columnas $1, 2$.

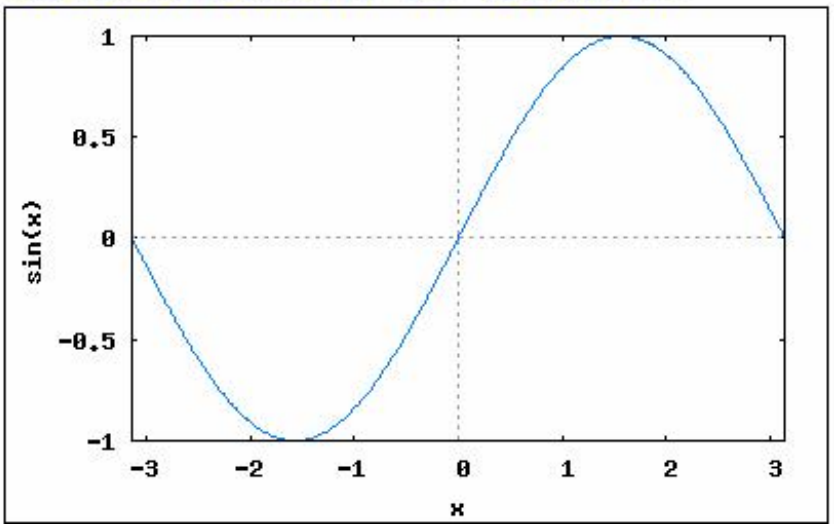
6. Representación gráfica de funciones

Para representar gráficamente una función se utiliza el comando `wxplot2d`:

```
wxplot2d(expr,[x,xmin,xmax])
```

donde "expr" es la función que se va a representar y $[x, min, max]$ es el intervalo de x en el que se dibuja la gráfica de la función. Por ejemplo:

```
(%i20) wxplot2d([sin(x)], [x,-%pi,%pi])$
```



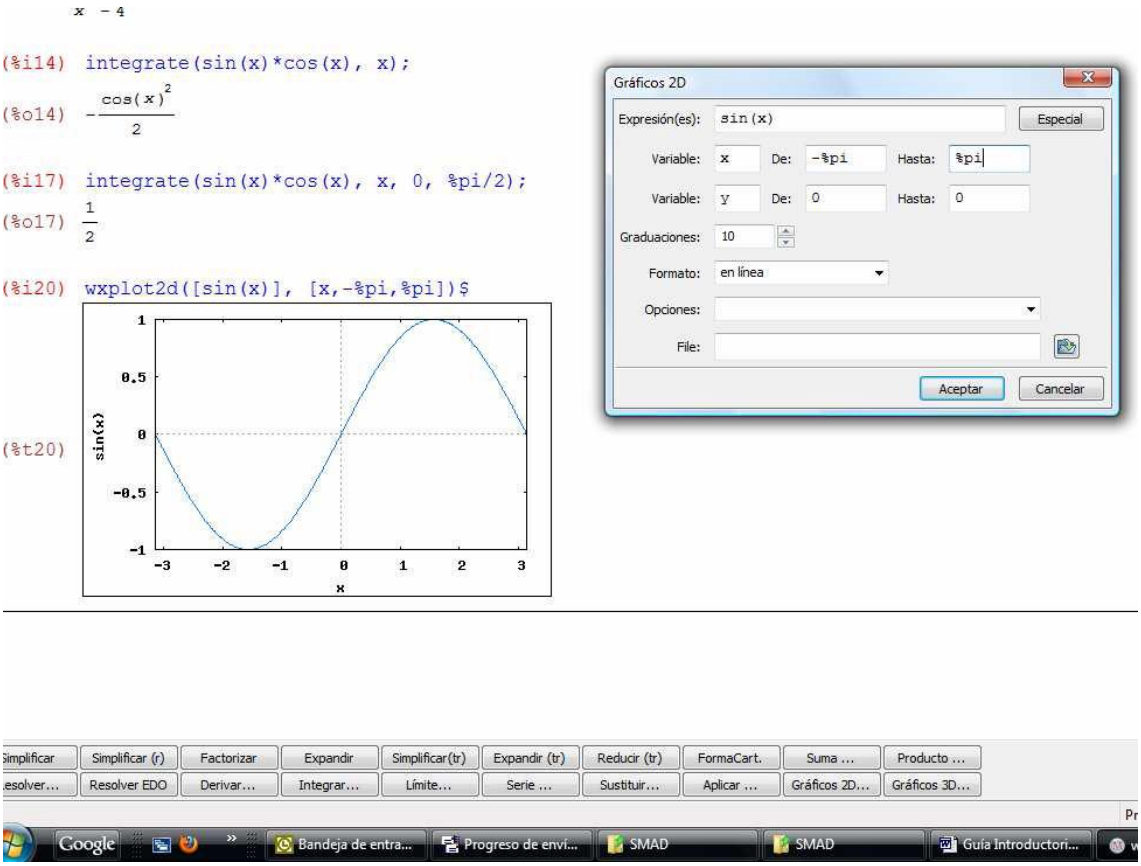

```
(%t20)
```

Si se escribe

```
plot2d(expr,[x,xmin,xmax])
```

la gráfica se muestra en una nueva pantalla en formato gnuplot.

También se puede hacer gráficas a partir de la barra de herramientas en la parte inferior de la pantalla. En la opción de formato se elige "en línea" si se desea que la gráfica aparezca en la misma ventana de trabajo.

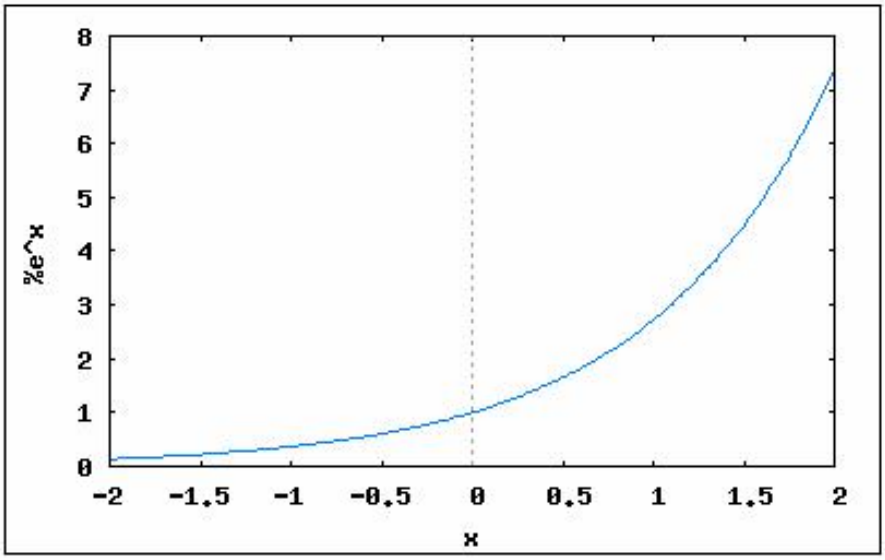


Si se ha definido una función previamente se puede utilizar el nombre que se le dio a la función en el comando `wxplot2d`.

El comando `plot` admite algunas opciones para especificar, por ejemplo, el rango de valores tanto del eje x como del eje y, el punto que se elige como origen de coordenadas, la escala de los ejes, etc. En el siguiente ejemplo se muestra la gráfica de la función exponencial con el eje y en escala lineal en la primera y en escala logarítmica en la segunda,

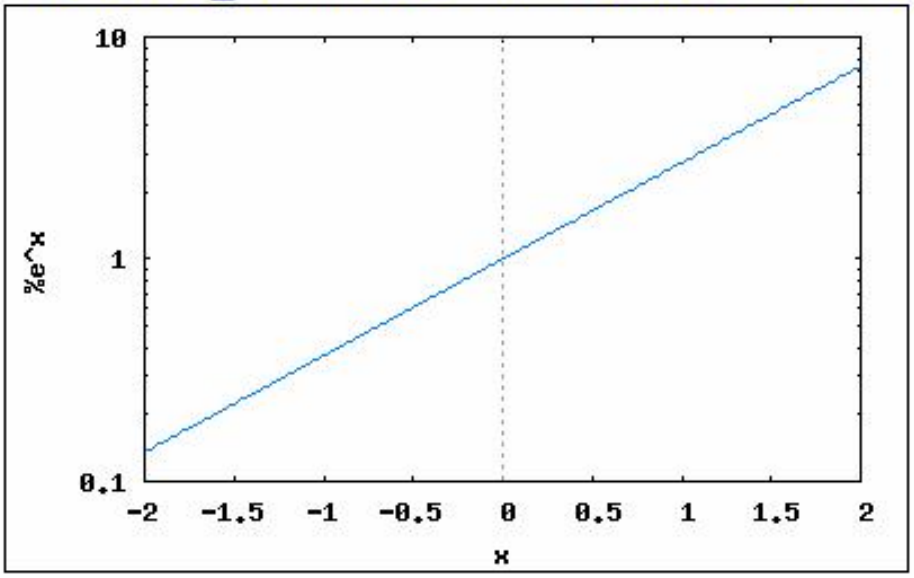
```
(%i29) wxplot2d([exp(x)], [x,-2,2])$
```

(%t29)




```
(%i30) wxplot2d([exp(x)], [x,-2,2],
  [gnuplot_preamble, "set logscale y;"]);
```

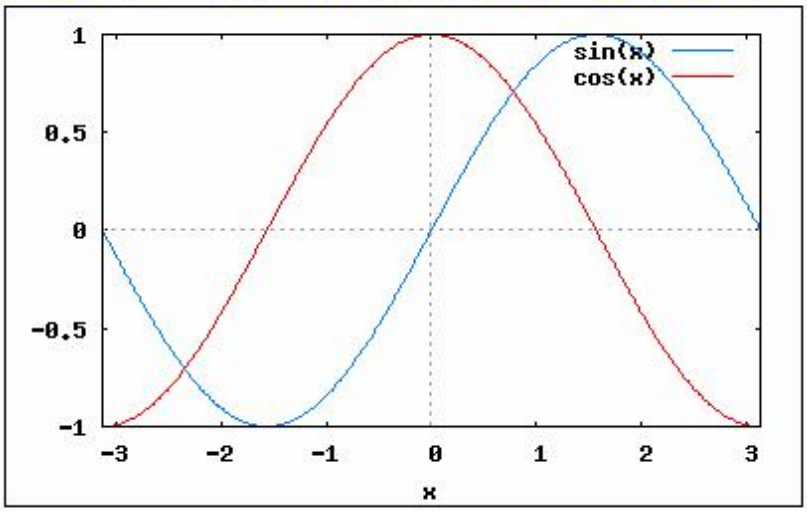
(%t30)



(%o30)

Se pueden también dibujar varias funciones en el mismo gráfico. Por ejemplo:

```
(%i12) wxplot2d([sin(x),cos(x)],[x,-%pi,%pi]);
```

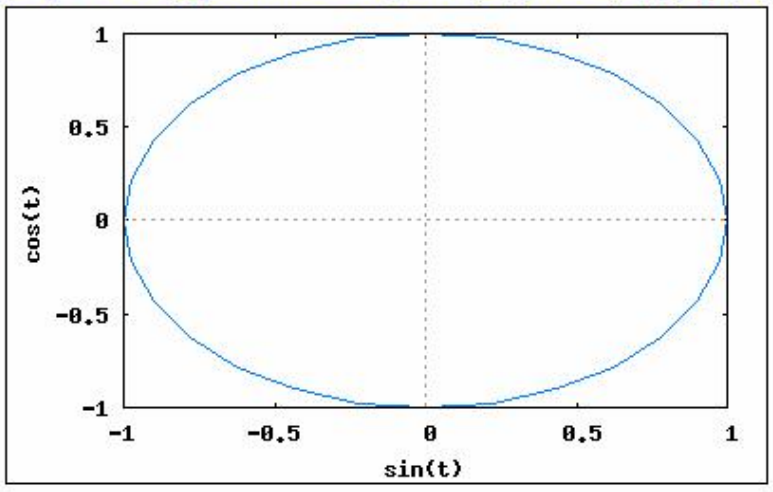



```
(%t12)

(%o12)
```

Para representar la gráfica de una función dada en coordenadas paramétricas se introduce la clave `parametric`, seguida de dos expresiones, x_{expr} , y_{expr}, que dependen de una única variable cuyo rango se especifica de la forma $[var, min, max]$. El gráfico mostrará los pares $[x_{expr}, y_{expr}]$ según var varía de min a max. Por ejemplo:

```
(%i13) wxplot2d([parametric,sin(t),cos(t)],[t,0,2*%pi]);
```



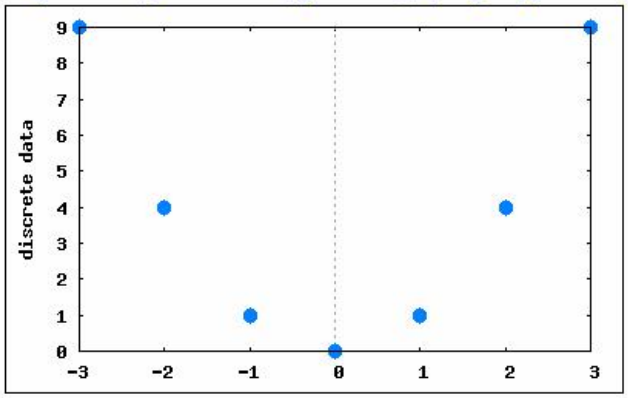

```
(%t13)

(%o13)
```

Para representar una lista de puntos

```
(%i17) puntos:makelist([x,x^2],x,-3,3);
(%o17) [[-3,9],[-2,4],[-1,1],[0,0],[1,1],[2,4],[3,9]]

(%i18) wxplot2d([discrete,puntos],[style,points]);
```



(%t18)

(%o18)

Ejercicio 10. Dibuja la gráfica de la función $y = \sin(x)$ entre los puntos 0 y π. Añade la gráfica del polinomio $y = x - x^3/6 + x^5/120$.

7. Estructuras de programación

MAXIMA dispone de una serie de operadores lógicos básicos que son fundamentales para la programación. Veamos algunos ejemplos:

Operación	Símbolo	Tipo
menor que	<	operador relacional infijo
menor o igual que	<=	operador relacional infijo
igualdad (sintáctica)	=	operador relacional infijo
negación de =	#	operador relacional infijo
igualdad (por valor)	equal	operador relacional infijo
negación de equal	notequal	operador relacional infijo
mayor o igual que	>=	operador relacional infijo
mayor que	>	operador relacional infijo
y	and	operador lógico infijo
o	or	operador lógico infijo
no	not	operador lógico prefijo

Usemos el operador `if` para comprobar si dos variables tienen el mismo valor o no:

```
(%i1) a:3;
(%o1) 3

(%i2) b:2;
(%o2) 2

(%i3) if a=b then c:1 else c:0;
(%o3) 0
```

Con la sentencia anterior el resultado será 1 si son iguales y 0 si son distintas.

Usemos el operador `for` para repetir una instrucción un determinado número de veces:

`for` i:i_0 `step` k `thru` i_1 `do` *cuerpo*

Con los comandos anteriores, el índice i recorre los valores desde i_0 hasta i_1 a pasos de tamaño k. En cada paso se ejecuta la sentencia que aparezca en *cuerpo*. Por ejemplo:

```
(%i4) a:[2,3,1,4,2];
(%o4) [2,3,1,4,2]

(%i5) for i:2 thru 5 do display(a[i]-a[i-1]);
3+-2=1
1+-3=-2
4+-1=3
2+-4=-2
(%o5) done
```

`for` i:i_0 `step` k `while` *condición* `do` *cuerpo*

Con los comandos anteriores, mientras que se verifique la *condición*, el índice i recorre los valores desde i_0 a pasos de tamaño k y en cada paso se ejecuta la la sentencia que aparezca en *cuerpo*. Por ejemplo:

```
(%i6) a:[2,3,1,4,2,8,9];
(%o6) [2,3,1,4,2,8,9]

(%i7) for i:1 thru 5 while (a[i]<=3 and i<=5) do display(a[i]);
a_1=2
a_2=3
a_3=1
(%o7) done
```

`for` i:i_0 `step` k `unless` *condición* i_1 `do` *cuerpo*

Con los comandos anteriores, hasta que se verifique la *condición*, el índice i recorre los valores desde i_0 a pasos de tamaño k y en cada paso se ejecuta la sentencia que aparezca en *cuerpo*. Por ejemplo:

```
(%i8) a:[2,3,1,4,2,8,9];
(%o8) [2,3,1,4,2,8,9]

(%i9) for i:1 unless (a[i]>=7 or i>4) do display(a[i]);
a_1 = 2
a_2 = 3
a_3 = 1
a_4 = 4
(%o9) done
```

Ejercicio 11. Crea una función que reciba un vector y devuelva la suma de sus componentes, usando un bucle para ello. Comprueba el resultado.

Ejercicio 12. (Para valientes) Crea una función que reciba un vector de números naturales y devuelva uno formado por aquellos que sean primos.