Maxima para bachillerato

Por: Manuel Jesús Quidiello

Maxima es un software libre GNU para cálculo matemático simbólico, esto es, no es una calculadora que opera con números, sino un entorno gratuito que realiza cálculo matemático con variables, constantes, parámetros, gráficas…

Existen versiones para todos los sistemas operativos actuales y una versión portable para el uso en cualquier tipo de computadora.
En este manual repasaremos los utilidades más usadas y relevantes para un alumno de bachillerato y primer curso de universidad.

1 Primeros pasos

Existen unas primeras consideraciones que hay que tener en cuenta para poder usar correctamente el software Maxima y que sin tenerlas en cuenta nos pueden acarrear problemas serios.

Para realizar cualquier tipo de operación en Maxima se utilizan comandos propios que se irán exponiendo en este manual (para avanzados se utilizará el manual original que se encuentra en su página web).

Es importante tener en cuenta que al terminar una instrucción, debemos cerrarla con un punto y coma para que tenga efecto.

```
solve([x^2-1], [x]);
```

En la versión para Windows nos encontraremos con un interface gráfico lo suficientemente potente como para que se nos haga la utilización de fórmulas, una tarea mucho más sencilla que la memorización de instrucciones.

Para que el comando se ejecute debemos de pulsar a la vez la tecla de Mayúsculas y la de Enter.

Para usar un resultado de realizado con anterioridad en posteriores cálculos, puede asignar dicho resultado a una **variable** o referirse a él por medio de la etiqueta asociada (%i* o %o*). Si tan solo pusiéramos %, haría mención al último comando introducido.

Adicionalmente puede usar % para referirse al último resultado obtenido.

```
(%i2) f: expand((x+2)^2);
(%o2) x^2+4 x+4

(%i4) diff (%o2,x);
(%o4) 2 x+4

(%i5) diff (f,x);
(%o5) 2 x+4
```

En este ejemplo expandimos (realizamos la identidad notable) y luego la derivamos respecto de x:

- Sin tener que volver a copiarla en el primer caso, haciendo referencia a la salida %o4
- Haciendo uso de la variable f a la que habíamos asociado la función.

Cuando no queramos que se evalúe la expresión por cualquier razón (ser un paso intermedio), tan solo debemos de comenzarla con una comilla simple (‘) y no se evaluará.

```
(%i1) expand((x+3)^3);
(%o1) x^3+9 x^2+27 x+27
```

```
(%i2) 'expand((x+3)^3);
(%o2) expand((x+3)^3)
```

```
(%i4) 'integrate (x^2, x, 1, 5);
(%o4) ∫_1^5 x^2 dx
```

SEGUNDO COMANDO

Podemos expresar como tiene que salir la operación que hemos introducido, esto es:

- Con mayor precisión numérica:

```
(%i1) sin (1/2), float;
(%o1) 0.4794255386042
```

- También podemos indicar que factorice una expresión:

```
(%i5) x^3 - 1, factor;
(%o5) (x-1)(x^2+x+1)
```

- Que expanda una expresión mediante expresiones trigonométricas:

```
(%i6) cos(4 * x) / sin(x)^4, trigexpand;
                  sin(x)^4-6 cos(x)^2 sin(x)^2+cos(x)^4
(%o6)             ------------------------------------
                                 sin(x)^4
```

1 Operadores

Los operadores pueden unarios, binarios, n-arios, aritméticos….

- Operador de **asignación:**
  - Para variables se usa el símbolo: :

```
(%i7) a:3;
(%o7) 3

(%i8) 5+a;
(%o8) 8
```

  - De funciones se usa el símbolo: :=

```
(%i2) F(x,y):=x^2+y;
                    2
(%o2) F(x,y):=x  +y

(%i3) F(2,3);
(%o3) 7
```

- Operadores **aritméticos básicos**:

Suma +	Resta -	Producto *	División /	Potenciación ^

- Producto no conmutativo (matricial): ·
- Operadores **relacionales**:

Menor <	Menor o igual <=	Mayor >	Mayor o igual >=	Negación #

- Operadores **no básicos**:

Valor absoluto: abs()
Valor absoluto de un número complejo: cabs()
Menor entero menor o igual: ceiling ()
Mayor entero menor o igual: floor()
Raíz cuadrada entera: isqt()
Factorial: !
Raíz cuadrada: sqrt()
Logaritmo neperiano: log()
Exponencial: exp ()
Resto entero de dividir a entre b: mod(a,b)
Número aleatorio entre 0 y x-1: random(x)

- **Trigonométricas:**

Seno: sin()	Arco seno: asin()
Coseno: cos()	Arco coseno: acos()
Tangente: tan()	Arco tangente: atan()
Cosecante: csc()	Arco cosecante: acsc()
Secante: sec()	Arco secante: asec()
Cotangente: cot()	Arco cotangente: acot()

- **Hiperbólicas:**

Seno hiperbólico: senh()	Arco seno hiperbólico: asinh()
Coseno hiperbólico: cosh()	Arco coseno hiperbólico: acosh()
Tangente hiperbólica: tanh()	Arco tangente hiperbólica: atanh()
Cosecante hiperbólica: csch()	Arco cosecante hiperbólica: acsch()
Secante hiperbólica: sech()	Arco secante hiperbólica: asceh()
Cotangente hiperbólica: coth()	Arco cotangente hiperbólica:acoth()

Expandir expresiones trigonométricas:

trigexpand(expresión);

```
(%i14) trigexpand(cos(y+x));
(%o14) cos(x)cos(y)-sin(x)sin(y)
```

2 Gráficos

La representación gráfica en Maxima puede hacerse mediante comandos directos o utilizando el interfaz gráfico de Windows.

Para una buena representación gráfica habría que utilizar el formato gnuplot en Windows, seleccionable desde la barra de comandos del interfaz gráfico.

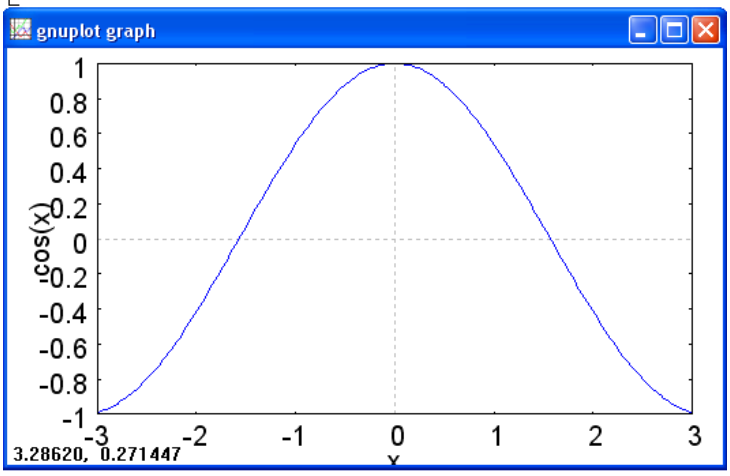
- **Por comandos: Gráficos paramétricos:** plot2d([función, [parametric,]])

plot2d ([función1, [parametric, función2x, función2y, [t, intervalo1, intervalo2], [nticks, 80]]], [x, a, b])$

nticks, 80: indica el número de puntos que debe calcular para describir la función.

- **Gráficos en en plano**: Plot2d(*función, [x,a,b]*)

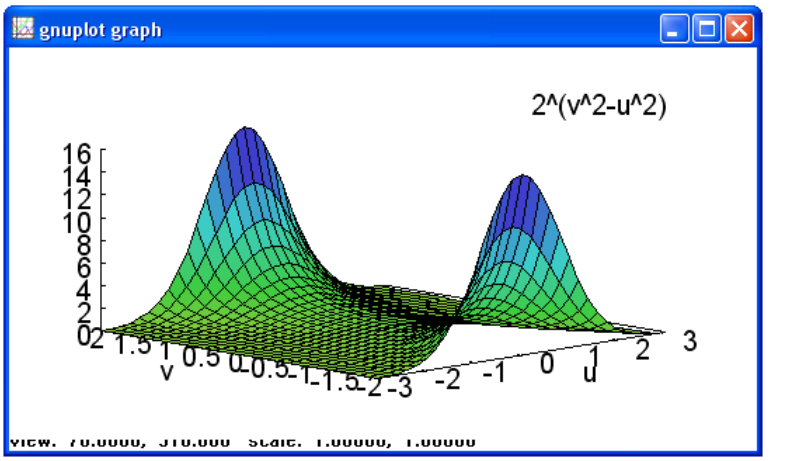
```
(%i35) plot2d (cos(x), [x, -3, 3]);
(%o35)
```



- **Gráficos en el espacio:**

Plot3d (*función,[x,a,b],[y,c,d]*)

```
(%i34) plot3d (2^(-u^2 + v^2), [u, -3, 3], [v, -2, 2]);
```



3 Polinomios

Dividir un polinomio entre otro, indicando la variable principal y dándonos el cociente:

quotient (dividendo,divisor);

```
(%i10) quotient (x^3-2*x^2+1,x-4);
(%o10) x^2+2 x+8
```

Dividir un polinomio entre otro, indicando la variable principal y dándonos el resto:

remainder (dividendo,divisor);

```
(%i11) remainder (x^3-2*x^2+1,x-4);
(%o11) 33
```

Dividir un polinomio entre otro, indicando la variable principal y dándonos el cociente y el resto de dicha división:

divide (dividendo, divisor, variable principal);

```
(%i2) divide (x^3+2*x^2-x-5, 2*x-3, x);
                                4 x^2 + 14 x + 17   11
(%o2)                          [----------------, --]
                                        8          8
```

Factoriza un polinomio o un número;

factor(*polinomio o factor*);

```
(%i3) factor(x^2-5*x+6);
(%o3) (x-3)(x-2)

(%i4) factor(5475);
                                  2
(%o4)                          3 5  73
```

Expandir un polinomio multiplicando los factores introducidos:

expand (*expresión*);

```
(%i6) expand((x-3)*(x+4));
                                2
(%o6)                          x  + x - 12
```

Cálculo del máximo común divisor entre varios polinomios, devolviendo también el cociente de cada polinomio entre dicho MCM en el caso de usar ezgcd en vez de gcd:

gcd (*p 1, p 2, p 3, ...)* ezgcd (*p 1, p 2, p 3, ...)*

```
(%i9) gcd(x^2-1,x+1,x^2+3*x+2);
(%o9) x+1
(%i8) ezgcd(x^2-1,x+1,x^2+3*x+2);
(%o8) [x+1,x-1,1,x+2]
```

4 Constantes

Las constantes son asignaciones predefinidas por el software y que no pueden utilizar como variables y su valor está introducido ya en el programa para ser utilizadas más cómodamente:

Número e: %e
Número imaginario: %i
Número Pi: %pi
Número aúreo: %phi
Infinito real positivo: inf
Infinito real negativo: minf
Infinito complejo: infinity

5 Matrices y vectores

En el conjunto de las matrices el operador producto es el símbolo . (como el punto normal), ya que el símbolo del producto normal *, se considera para productos conmutativos.

Para definir una matriz debemos utilizar el comando *matrix ();* :

```
(%i1) [2,3,5].[2,2,1];
(%o1) 15
```

```
(%i7) a:matrix([2,3],[1,5]);
(%o7)  [ 2  3 ]
       [ 1  5 ]
```

```
(%i8) b:matrix([1,0],[-1,2]);
(%o8)  [  1  0 ]
       [ -1  2 ]
```

```
(%i9) a.b;
(%o9)  [ -1   6 ]
       [ -4  10 ]
```

adjoint (M): devuelve el adjunto de la matriz introducida:

```
(%i7) a:matrix([2,3],[1,5]);
(%o7)  [ 2  3 ]
       [ 1  5 ]
```

```
(%i15) adjoint(a);
(%o15)  [  5  -3 ]
        [ -1   2 ]
```

determinant (M) : devuelve el determinante de la matriz seleccionada:

```
(%i7) a:matrix([2,3],[1,5]);
                        [ 2  3 ]
(%o7)                   [      ]
                        [ 1  5 ]

(%i16) determinant(a);
(%o16) 7
```

echelon(M) : devuelve la matriz escalonada M:

```
(%i1) M: matrix ([3, 7, aa, bb], [-1, 8, 5, 2], [9, 2, 11, 4]);
                        [ 3   7   aa   bb ]
                        [                 ]
(%o1)                   [ -1  8   5    2  ]
                        [                 ]
                        [ 9   2   11   4  ]

(%i2) echelon (M);
                        [ 1  -8  -5      -2      ]
                        [                        ]
                        [          28       11   ]
(%o2)                   [ 0   1   --       --   ]
                        [          37       37   ]
                        [                        ]
                        [ 0   0   1   37 bb - 119 ]
                        [             ---------- ]
                        [             37 aa - 313 ]
```

ident(M): devuelve la matriz identidad de orden n:

```
(%i3) ident(3);
                        [ 1  0  0 ]
                        [         ]
(%o3)                   [ 0  1  0 ]
                        [         ]
                        [ 0  0  1 ]
```

invert(M): devuelve la matriz inversa de M:

```
(%i5) M: matrix ([3, 7, 4], [-1, 8, 5], [9, 2, 11]);
                        [ 3   7   4  ]
                        [            ]
(%o5)                   [ -1  8   5  ]
                        [            ]
                        [ 9   2   11 ]

(%i6) invert(M);
                        [  13     23     1   ]
                        [  --   - ---    --  ]
                        [  55     110    110 ]
                        [                    ]
                        [  28     1      19  ]
(%o6)                   [ ---   - --   - --- ]
                        [ 165     110    330 ]
                        [                    ]
                        [  37     19     31  ]
                        [ - ---   --     --- ]
                        [ 165     110    330 ]
```

transpose(M): devuelve la matriz transpuesta de M:

```
(%i5) M: matrix ([3, 7, 4], [-1, 8, 5], [9, 2, 11]);
                [ 3   7   4  ]
                [            ]
(%o5)           [ -1  8   5  ]
                [            ]
                [ 9   2   11 ]

(%i11) transpose(M);
                [ 3   -1  9  ]
                [            ]
(%o11)          [ 7   8   2  ]
                [            ]
                [ 4   5   11 ]
```

triangularize(M): devuelve la matriz triangular inferior de la matriz M:

```
(%i12) triangularize(M);
                [ 3   7    4   ]
                [              ]
(%o12)          [ 0   31   19  ]
                [              ]
                [ 0   0    330 ]
```

6 Funciones especiales

solve(*ecuación, variable)* solve *([eqn 1, ..., eqn n], [x 1, ..., x n]):* resuelve cualquier ecuación respeto a la variable indicada.

solve (x^2 - 3*x + 5 = 15, x);

linsolve([ecuación1,ecuación2],[variable1,variable2]): resuelve ecuaciones lineales de varias ecuaciones y varias incógnitas.

linsolve ([2*x - 5*y = -3, 4*x - a*y = 13], [x, y]);

limit (expr, x, val): Calcula el límite de la expresión cuando la variable se aproxima al valor.

limit (x/(x+1), x, 4)

diff (función,variable): Calcula la derivada de la función respecto a la variable indicada.

diff (x^2+2x+5 , x);

integrate (función,variable): calcula la integral indefinida de la función respecto a la variable indicada.

integrate(x^2+3,x);

integrate (función,variable, x1,x2): calcula la integral definida de la función respecto a la variable indicada entre los límites de integración x1 y x2.

integrate(x^2+3,x,3,4);

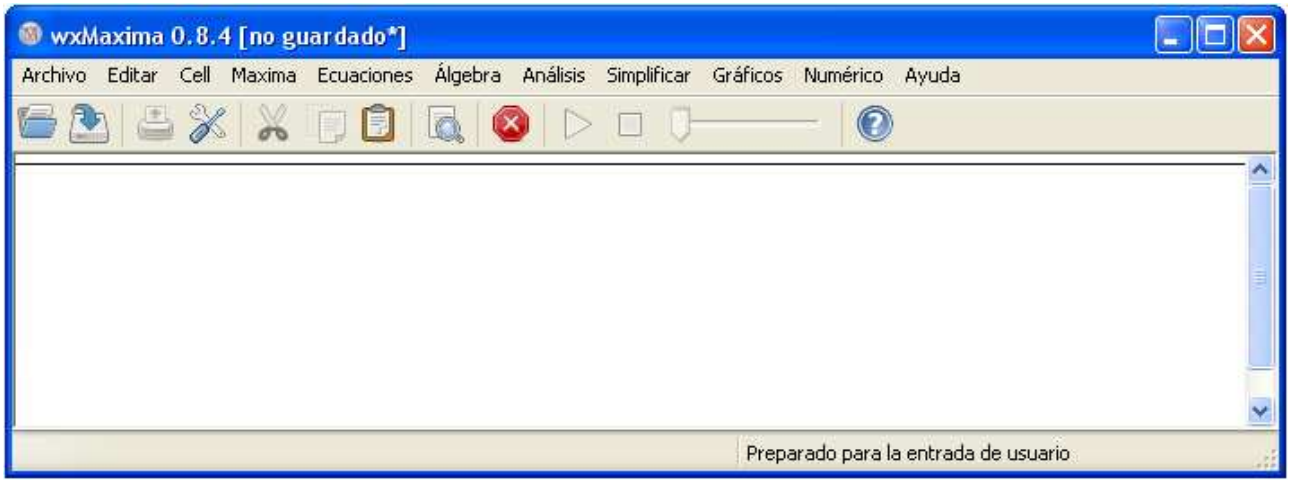
sum(expresión,variable,numero1,numero2): genera una serie con la expresión donde la variable va desde numero 1 a numero2:

sum (a[k], k, 1, 3);

7 La interfaz gráfica

Se denomina interfaz gráfica al entorno gráfico del sistema para comunicarse con el usuario. Maxima tiene un interfaz que facilita muchas de las operaciones matemáticas básicas y que nos ahorra de conocer el comando y su estructura para poder usarlo. Pasaremos a repasar de forma superficial el entorno y sus comandos más usados:

La siguiente imagen es el interfaz gráfico para el sistema Windows. Si se utiliza el software portable aparecerá además paneles por defecto en la parte inferior de la ventana.



El la parte central se irán introduciendo tanto las entradas a ejecutar como las salidas una vez procesadas.

Las entradas comienzan con el símbolo %i mientras que las salidas comienzan por %o:

```
(%i13) 3+5;
(%o13) 8
```

Podemos activar los paneles del interfaz en la opción cell para tener visualizados la mayoría de las opciones básicas a utilizar.

Podemos usar el resto de opciones de la barra de herramientas para no tener que recordar los comandos directos, tan solo con rellenar de forma sencilla la venta que aparezca en cada caso.