Maxima

Software Libre Científico

María José Ginzo Villamayor, Alejandro Saavedra Nieves e Paula Saavedra Nieves

Departamento de Estatística, Análise Matemática e Optimización
Universidade de Santiago de Compostela

31/03/2022

Esquema da presentación

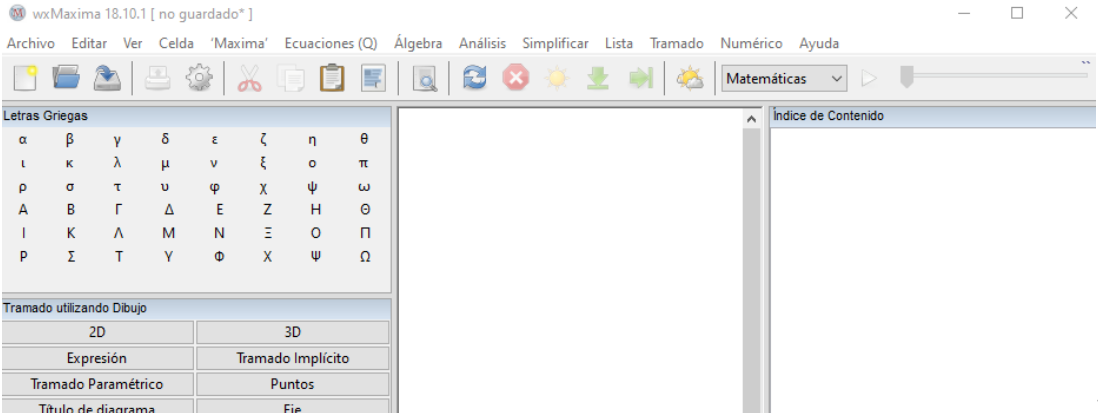
1. **Primeiros pasos con** `Maxima`
2. **Funcións básicas de** `Maxima`
3. **Representacións gráficas**
4. **Límites e derivadas**
5. **Matrices**
6. **Programación en** `Maxima`

Que é Maxima?

- Maxima é un sistema para a manipulación de expresións simbólicas e numéricas, incluíndo diferenciación, integración, expansión en series de Taylor, transformadas de Laplace, EDOs, sistemas de ecuacións lineais, vectores, matrices e tensores.
- Produce resultados de alta precisión.
- Grafica funcións e datos en dúas e tres dimensións.
- O código fonte pode ser compilado en varios sistemas operativos incluíndo Windows, Linux e MacOS X.
- Maxima é un descendente de Macsyma, o lendario sistema de álxebra computacional desenvolto a finais de 1960 no Instituto Tecnolóxico de Massachusetts (MIT).
- Dinámico: melloras no código e na documentación.
- Pode descargarse desde a súa páxina web: https://maxima.sourceforge.io/

Instalación

O programa é de libre distribución e pódese atopar na seguinte ligazón: https://maxima.sourceforge.io/
Descargamos o paquete adecuado para o noso sistema operativo, e instalámolo. Xa teriamos o noso lugar de traballo.

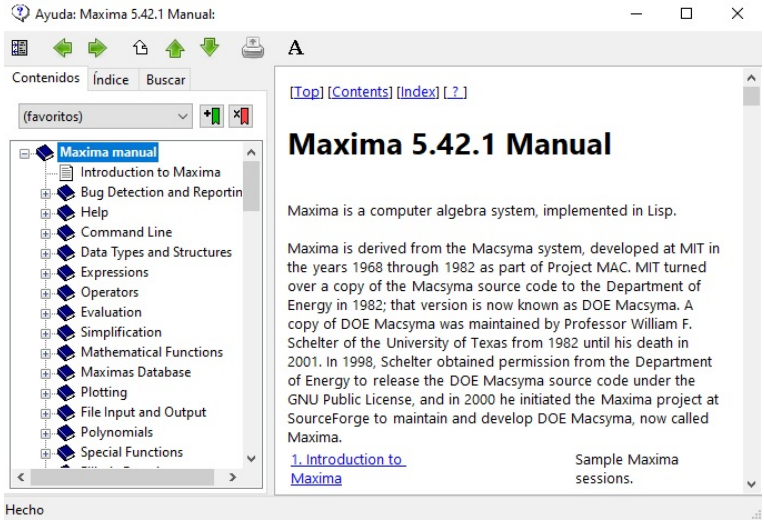


A axuda de Maxima

- A contorna wxMaxima permite acceder á ampla axuda incluída con Maxima dunha maneira gráfica.
- No mesmo menú temos algúns comandos que nos poden ser útiles:

describe(*expr*)	axuda sobre expr
example(*expr*)	exemplo de expr
apropos(‘‘expr’’)	comandos relacionados con expr
??expr	comandos que conteñen expr

A axuda de Maxima



A axuda de Maxima

Se coñecemos o nome do comando sobre o que estamos a buscar axuda, a orde `describe` dános unha explicación sobre a variable, comando ou o que sexa que preguntásemos.

```
(%i2) describe(dependencies);
      0: dependencies (Functions and Variables for Differentiation)
      1: dependencies $<1>$ (Functions and Variables for Differentiation)
Enter space-separated numbers, 'all' or 'none':
```

Ás veces nos equivocamos e non nos acordamos exactamente do nome do comando.

```
(%i2) describe(plot)
      No exact match found for topic ?plot?.
      Try ??? plot? (inexact match) instead.
(%o2) false
```

A axuda de Maxima

A solución témola escrita xusto na saída anterior: ?? busca en axuda comandos, variables, etc. que conteñan a cadea "plot".

```
(%i2) ??plot

0: Funciones y variables para plotdf
1: Introducción a plotdf
2: barsplot (Funciones y variables para gráficos estadísticos)
3: boxplot (Funciones y variables para gráficos estadísticos)
4: contour_plot (Funciones y variables para gráficos)
5: gnuplot_close (Funciones y variables para gráficos)
6: gnuplot_replot (Funciones y variables para gráficos)
7: gnuplot_reset (Funciones y variables para gráficos)
8: gnuplot_restart (Funciones y variables para gráficos)
9: gnuplot_start (Funciones y variables para gráficos)
10: plot2d (Funciones y variables para gráficos)
11: plot3d (Funciones y variables para gráficos)
12: plotdf (Funciones y variables para plotdf)
13: plot_options (Funciones y variables para gráficos)
14: scatterplot (Funciones y variables para gráficos estadísticos)
15: set_plot_option (Funciones y variables para gráficos)
Enter space-separated numbers, ?all? or ?none?:none;
```

A axuda de Maxima

- Se hai varias posibles eleccións, `Maxima` queda esperando ata que escribimos o número que corresponde ao ítem en que estamos interesados, ou `all` ou `none` se estamos interesados en todos ou en ningún, respectivamente.
- No menú de `wxMaxima` **Axuda ▶ Propósito para**. O seu propósito é similar a `??`, pero o resultado é levemente distinto, dános a lista de comandos nos que aparece a cadea `plot` sen incluír nada máis. Se xa temos unha idea do que estamos a buscar, moitas veces será suficiente con isto.

```
(%i2) apropos("plot");
(%o2)
[barsplot,boxplot,contour_plot,contour_plot-1,get_plot_option,gnuplot,gnuplot_4_0,gnuplot_cl
```

A axuda de Maxima

Coa orde example, obtemos un exemplo sobre como se utiliza unha orde que unha explicación "teórica".

```
(%i2) example(limit);
(%i2) limit(x*log(x),x,0,plus);
(%i2) limit((x+1)b(1/x),x,0);
(%i2) limit(%eˆx/x,x,inf);
(%i2) limit(sin(1/x),x,0);
```

A axuda completa de Maxima está dispoñible na súa páxina web: http://maxima.sourceforge.net/es/ en formatos PDF web. Existen moitas páxinas web que explican practicamente calquera detalle que se che poida ocorrer.

Operacións básicas

O sistema `wxMaxima` emprega a notación estándar para escribir as operacións matemáticas básicas: suma $+$, resta $-$, produto $*$, cociente $/$ e potencia ^. Tras introducir unha expresión, basta pulsar a tecla INTRO para que `Maxima` o execute.

```
(%i2) (2+3*(a^2)^3)/12;
```
(%o2) $\frac{3a^6+2}{12}$
```
(%i3) sqrt(12)+25^8;
```
(%o3) $2\sqrt{3}+152587890625$

É importante lembrar que en `Maxima` é necesario escribir sempre o operador da multiplicación ($*$). Pola contra dará un erro de sintaxe.

```
(%i2) 2x^3;
```
(%o2) incorrect syntax: x is not an infix operator2x^^
```
(%i2) 2*x^3;
```
(%o2) $2x^3$

Algunhas funcións matemáticas elementais

- Para introducir e^x escríbese $exp(x)$ ou %$ê$$x$
- Para introducir raíz de x escríbese $sqrt(x)$.
- Para introducir $|x|$ escríbese $abs(x)$.
- Para introducir $ln(x)$ escríbese $log(x)$.
- Para introducir outro tipo de logaritmo, decimal por exemplo, escríbese $log10(x) := log(x)/log(10)$ ou ben directamente $log(x)/log(10)$.

Operar con números complexos

Para traballar con números complexos hai que ter en conta que:

- A unidade imaxinaria escríbese %i
- Para a forma binómica ($a + bi$) hai que introducir `a+b*%i`.
- Para a forma exponencial ($re^{i\alpha}$) hai que introducir `r*%e^(%i*`α`)`.
- Para achar o módulo dun número complexo z: `cabs(z)`.
- Para achar o argumento dun número complexo z: `carg(z)`.
- Para achar a forma binómica de z; `rectform(z)`.
- Para achar a forma exponencial de z: `polarform(z)`.

Operar con números complexos

```
(%i5) 2-2*%i;
(%o5) 2 -2 %i
(%i5) cabs(2-2*%i);
(%o5) 2^(3/2)
(%i5) carg(2-2*%i);
(%o5) -%pi/4
(%i5) polarform(2-2*%i);
(%o5) 2^(3/2)%e^-(%i%pi/4)
(%i5) rectform(2^(3/2)*%e^(-(%i*%pi)/4));
(%o5) 2-2%i
```

Asignar nomes a datos ou expresións

```
(%i5) z:2-2*%i;
(%o5) 2 -2 %i
(%i5) cabs(z);
(%o5) 2^3/2
(%i5) carg(z);
(%o5) -%pi/4
(%i5) k:%e^%i;
(%o5) %e^%i %e
(%i5) rectform(k)
(%o5) %i sin(1) + cos(1)
```

Introducir comentarios

Comentarios

/∗ texto de comentarios ∗/

En `Maxima`, un comentario é calquera texto encerrado entre as marcas $/*$ e $*/$.

```
(%i5) /*Apartado (a)*/ f(x):=x^2;
```

(%o5) $f(x) := x^2$

Introducir e manexar funcións. Definición

Funcións

f():= funcion

Pódese definir a función cunha ou varias variables.

```
(%i5) f(x):=2*x;
```

(%o5) $f(x) := 2x$

```
(%i5) g(x,y):=2*x^2+5*y^4;
```

(%o5) $g(x,y) := 2x^2 + 5y^4$

É fundamental identificar as variables e usar **:=** para definila.

Introducir e manexar funcións. Definición

Para definir unha función a anacos utilízase:

If *condición* **then** *sentenza1* **else** *sentencia2*

A función $h(x) = \begin{cases} x & \text{si } x \leq 0 \\ \text{sen}(x) & \text{si } x > 0 \end{cases}$, defínese

```
(%i5) h(x):=if x<=0 then x else sin(x);
```

(%o5) $h(x) := \text{if } x \leq 0 \text{ then } x \text{ else } \sin(x)$

Resolver ecuacións

solve

O comando básico para resolver ecuacións de todo tipo é `solve`.

```
(%i2) solve(4*x^2+x=5,x);
```

(%o2) $[x = -\frac{5}{4}, x = 1]$

Resolver ecuacións

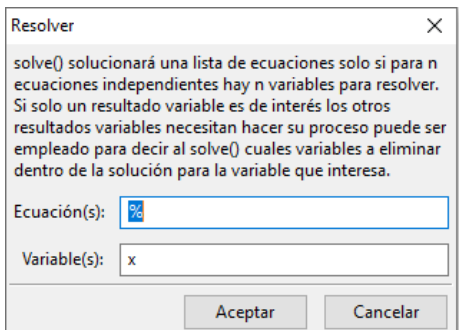
O comando `solve` pode resolver sistemas de ecuacións (mesmo non lineais). Para iso, hai que pasarlle unha lista coas ecuacións para resolver. As listas sempre se dan entre corchetes [ecuación1, ecuación2,...] e de novo hai que especificar a(s) variable(s) respecto da(s) que se quere resolver o sistema. As solucións, se existen, tamén se dan en forma de lista.

```
(%i2) eq1:2*x+y=5;
```
(%o2) $y + 2x = 5$
```
(%i2) eq2:2*x+5*y=10;
```
(%o2) $5y + 2x = 10$
```
(%i2) solve([eq1, eq2], [x,y]);
```
(%o2) $[[x = \frac{15}{8}, y = \frac{5}{4}]]$

Resolver ecuacións

Outra forma de resolver ecuacións ou sistemas é utilizar a opción de menú:

Ecuaciones ▶ Resolver



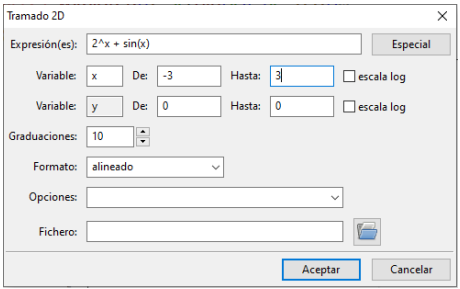
Funcións dunha variable. Gráficas en 2D

Pódese representar gráficamente unha función picamos no menú de

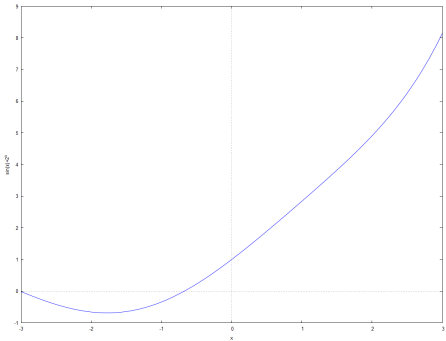
Tramado ▶ Tramado 2d....

Aparece unha pantalla para introducir o rango e o formato desexado e ao pulsar Aceptar aparece o gráfico.

Funcións dunha variable. Gráficas en 2D

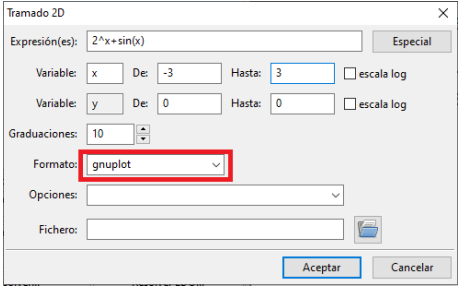


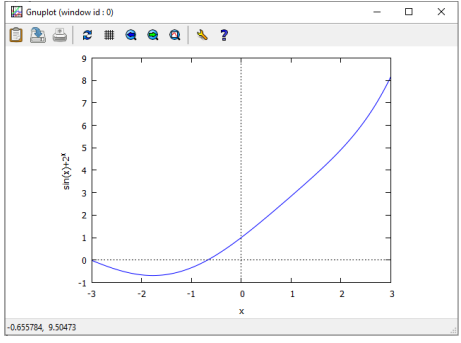

```
(%i2)
wxplot2d([2^x + sin(x)], [x,-3,3])$
(%o2)
```



Funcións dunha variable. Gráficas en 2D

Se queremos que o gráfico se abra noutra pestana, ten a vantaxe de que che indica as coordenadas do punto marcado polo cursor en cada momento, deberemos elixir o formato: **gnuplot**. Neste caso, é recomendable pechar a xanela gráfica ao terminar, para seguir traballando.



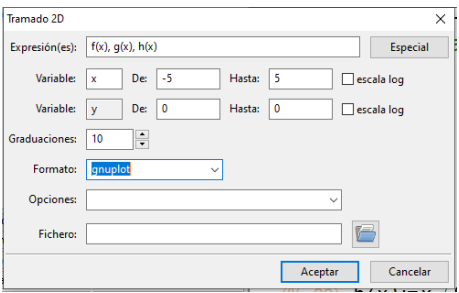


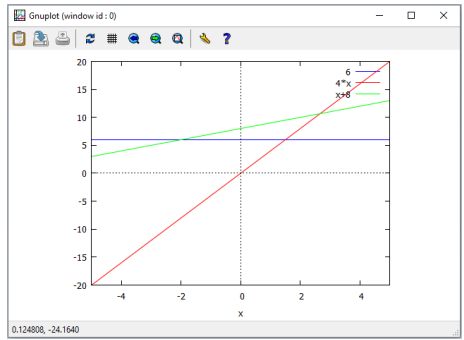
Representar varias funcións simultáneamente

Unha vez definidas as funcións, por exemplo, $f(x)$, $g(x)$, e $h(x)$, abrimos o menú de **Tramado ▶ Tramado 2d...** e introducímolas separadas por comas.
Ao pulsar Aceptar aparece a pantalla con todas as funcións pintadas.

Representar varias funcións simultáneamente

```
(%i2) f(x):=6;
```
(%o2) $f(x) := 6$
```
(%i2) g(x):=4*x;
```
(%o2) $g(x) := 4.x$
```
(%i2) h(x):=x+8;
```
(%o2) $h(x) := x + 8$



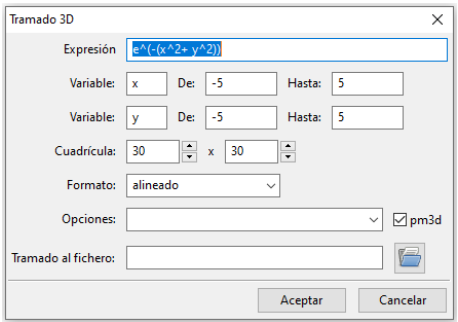



```
(%i2)
plot2d([f(x), g(x), h(x)], [x,-5,5],
[plot_format, gnuplot])$
```

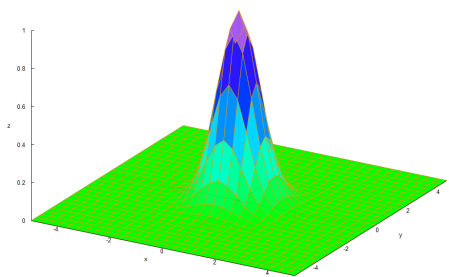
Funcións de dúas variables. Gráficas en 3D

A gráfica dunha función de dúas variables é unha superficie, que se pode representar co botón **Tramado ▶ Tramado 3d...**, introducindo os datos na xanela de diálogo, de modo análago a como se fai cunha función dunha variable.

Funcións de dúas variables. Gráficas en 3D



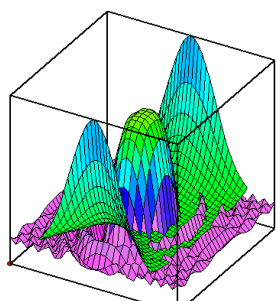

```
(%i2)
wxplot3d(%e^(-(x^2+ y^2)), [x,-5,5],
[y,-5,5])$
```



Funcións de dúas variables. Gráficas en 3D

Exemplo

```
(%i2)
plot3d ([[2^(-x^2 + y^2),[x,-2,2],[y,-2,2]], 4*sin(3*(x^2+y^2))/(x^2+y^2),
[x, -3, 3], [y, -2, 2]], [plot_format,xmaxima])$;
```



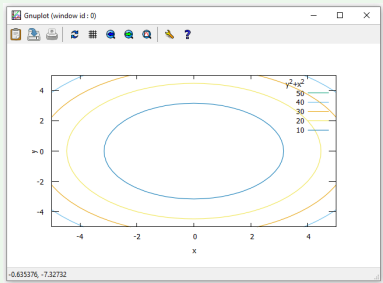
Funcións de dúas variables. Gráficas en 3D

Curvas de nivel

Tamén se poden representar as curvas de nivel dunha superficie utilizando a función `contour_plot`.

Exemplo

```
(%i2) contour_plot(x^2 + y^2, [x,-5,5], [y,-5,5]);
```



Límites

limit

O comando para achar límites, **limit**, é un dos máis sinxelos de usar. Para calcular $\lim_{x \to a} f(x)$ úsase $limit(f(x), x, a)$

```
(%i2) f(x):=(x+a)*(x^2+b*x+c);
```
(%o2) $f(x) := (x + a)(x^2 + b * x + c)$
```
(%i2) limit(f(x),x,1);
```
(%o2) $(a + 1)c + (a + 1)b + a + 1$
```
(%i2) limit(f(x),x,k);
```
(%o2) $k^3 + (b + a)k^2 + (c + ab)k + ac$

Límites

Maxima identifica límites finitos e infinitos. Tamén pode calcular límites laterais, coas directivas **minus** (límites pola esquerda) e **plus** (límites pola dereita).

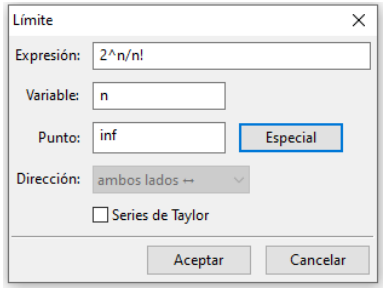
```
(%i2) limit(1/x,x,0,minus);
(%o2) -∞
(%i2) limit(1/x,x,0,plus);
(%o2) ∞
```

Límites

Outra forma, de calcular un límite sería

Análisis ▶ Encuentra límite...

escribir a expresión desexada, indicar a variable (x), o punto (x0) e a dirección (Esquerda, Dereita ou ambos os), e pulsamos aceptar.




```
(%i2) limit(2^n/n!, n, inf);
(%o2) 0
```

Derivadas

diff

A instrución para derivar respecto a unha variable é *diff* (función, variable).

```
(%i2) g(x,y):=sin(x*y);
(%o2) g(x,y) := sin(xy)
(%i2) diff(g(x,y),x);
(%o2) y cos(xy)
(%i2) diff(g(x,y),y);
(%o2) x cos(xy)
```

Derivadas

Pódense calcular derivadas segundas, terceiras, etc., sen máis que indicar a orde de derivación a continuación da variable.

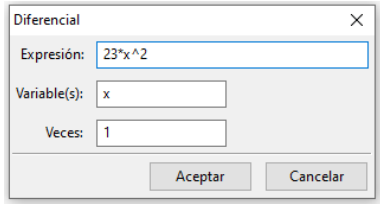
```
(%i2) diff(g(x,y),x,2);
```
$$(\%o2)\ -y^2\ sin(xy)$$
```
(%i2) diff(g(x,y),x,4);
```
$$(\%o2)\ y^4\ sin(xy)$$
```
(%i2) diff(diff(g(x,y),y,2), x, 2);
```
$$(\%o2)\ x^2\ y^2\ sin(xy) - 2sin(xy) - 4xy\ cos(xy)$$

Derivadas

Outra maneira, de calcular a derivada sería

Análisis ▶ Diferenciar...

escribir a expresión desexada, indicar a variable (x), a orde da derivada que queremos calcular e pulsamos aceptar.




```
(%i2) diff(23*x^2,x,1);
(%o2) 46x
```

Definir unha matriz

Pódense definir matrices de diferentes formas:

1. Utilizando as distintas opcións do menú **Álgebra ▶ Introducir matriz**; **Álgebra ▶ Generar matriz**; **Álgebra ▶ Generar matriz a partir de expresión**; ...
2. Declarando os seus elementos mediante listas, unha para cada fila, coa instrución `matrix([a11,a12,a13,...],[a21,a22,a23,...],...[an1,an2,...])`
3. Introducindo interactivamente baixo demanda os seus elementos coa instrución `entermatrix(NúmeroFilas, NúmeroColumnas)`
4. Mediante unha fórmula que define o elemento xenérico da matriz: `a[i,j]:=Fórmula de i y j$ genmatrix(a, NúmeroFilas, NúmeroColumnas)`

Definir unha matriz

Exemplo

```
(%i2) a:matrix([3,3,3],[1,3,5],[4,2,7]);
```

$$(\%o2)\quad \begin{bmatrix} 3 & 3 & 3 \\ 1 & 3 & 5 \\ 4 & 2 & 7 \end{bmatrix}$$

```
(%i2) genmatrix(b,2,2);
```

$$(\%o2)\quad \begin{pmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{pmatrix}$$

```
(%i2) b[i,j]:=i^2+j^2;
```

$$(\%o2)\quad b_{i,j} := i^2 + j^2$$

```
(%i2) genmatrix(b,2,2);
```

$$(\%o2)\quad \begin{pmatrix} 2 & 5 \\ 5 & 8 \end{pmatrix}$$

Definir unha matriz

Por outra banda, as matrices especiais, como as diagonais, simétricas, nulas ou a identidade, poden construírse utilizando o menú **Álgebra ▶ Introducir matriz**, ou ben comandos específicos:

- `diagmatrix(Número,Valor)`, que xera unha matriz diagonal de orde *Número* con elementos non nulos na diagonal, todos eles co mesmo *Valor*
- `ematrix(m,n,Z,i,j)`, que xera unha matriz $m \times n$ case nula na que todos os elementos nulos salvo o (i, j) cuxo valor é Z
- `zeromatrix(n,m)`, que xera a matriz nula de n filas e m columnas
- `ident(n)`, que xera a matriz identidade $n \times n$.

Definir unha matriz

Exemplo

```
(%i2) diagmatrix(4,5);
```

(%o2)
$$\begin{pmatrix} 5 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 \\ 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 5 \end{pmatrix}$$

```
(%i2) ematrix(3,3,5,3,2);
```

(%o2)
$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 5 & 0 \end{pmatrix}$$

```
(%i2) zeromatrix(2,2);
```

(%o2)
$$\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$$

```
(%i2) ident(5);
```

(%o2)
$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Recuperar elementos e submatrices

É posible asignar unha matriz a unha variable en `Maxima`, e logo extraer de forma independente filas(row), columnas(column) ou outro tipo de submatrices facendo uso dos comandos seguintes:

- `col(Matriz,NúmColumna)`, que recupera a columna cuxo número se indica.
- `row(Matriz,NúmFila)`,que recupera a fila cuxo número se indica.
- `Matriz[i, j]`, que recupera o elemento da fila i, columna j.
- `submatrix(i_1,i_2,...,i_p,Matriz,j_1,j_2,...,j_q)`, que elimina da *Matriz* as filas cuxos números son $i_1 \dots i_p$ e as columnas cuxos números son $j_1 \dots j_q$. Poden eliminarse unicamente filas ou columnas.
- `addrow(Matriz, lista_1,...,lista_p)`, que engade na base de *Matriz* as filas dadas polas listas (ou matrices) $lista_1, \dots, lista_p$. As lonxitudes deben ser concordantes.
- `addcol(Matriz,lista_1,...,lista_p)`, que engade á dereita de *Matriz* as filas dadas polas listas (ou matrices) $lista_1, \dots, lista_p$. As lonxitudes deben ser concordantes.

Recuperar elementos e submatrices

Exemplo

```
(%i2) c[i,j]:=i^2+j^2;
```

(%o2) $c[i,j] := i^2 + j^2$

```
(%i2) miMatriz:genmatrix(c,3,3);
```

(%o2)
$$\begin{pmatrix} 2 & 5 & 10 \\ 5 & 8 & 13 \\ 10 & 13 & 18 \end{pmatrix}$$

```
(%i2) miMatriz[2,1] /*recupera o elemento [2,1]*/;
```

(%o2) 5

```
(%i2) col(miMatriz,2) /*recupera a columna 2*/;
```

(%o2)
$$\begin{pmatrix} 5 \\ 8 \\ 13 \end{pmatrix}$$

```
(%i2) row(miMatriz,2) /*recupera a fila 2*/;
```

(%o2) $\begin{pmatrix} 5 & 8 & 13 \end{pmatrix}$

Recuperar elementos e submatrices

Exemplo

```
(%i2) submatrix(2,miMatriz,1) /*elimina a fila 2 e columna 1*/;
```

$$(\%o2)\quad \begin{pmatrix} 5 & 10 \\ 13 & 18 \end{pmatrix}$$

```
(%i2) addrow(miMatriz,[1,3,3]) /*engade unha nova fila*/;
```

$$(\%o2)\quad \begin{pmatrix} 2 & 5 & 10 \\ 5 & 8 & 13 \\ 10 & 13 & 18 \\ 1 & 3 & 3 \end{pmatrix}$$

```
(%i2) addcol(miMatriz,[2,3,3]) /*engade unha nova columna*/;
```

$$(\%o2)\quad \begin{pmatrix} 2 & 5 & 10 & 2 \\ 5 & 8 & 13 & 3 \\ 10 & 13 & 18 & 3 \end{pmatrix}$$

Operacións con matrices

Poden realizarse diferentes operacións con matrices usando os seguintes operadores:

- $+$ suma de dúas matrices,
- $-$ diferenza de dous matrices,
- $\cdot$ produto ordinario de dúas matrices,
- $*$ multiplicación de dous matrices, elemento a elemento, e tamén multiplicar por un número fixo todos os elementos,
- $/$ división de dúas matrices, elemento a elemento,
- ^^elevar unha matriz a unha potencia,
- ^elevar cada un dos elementos dunha matriz a unha potencia.

Operacións con matrices

- `traspose(NombreMatriz)`, que calcula a trasposta
- `adjoint(NombreMatriz)`, que calcula a adjunta
- `invert(NombreMatriz)`, que calcula a inversa utilizando o método dos adjuntos
- `determinant(NombreMatriz)`, que calcula o determinante dunha matriz
- `rank(NombreMatriz)`, que calcula o rango
- `charpoly(NombreMatriz)`, que calcula o polinomio característico
- `eigenvalues(NombreMatriz)`, que calcula os valores propios
- `eigenvectors(NombreMatriz)`, que calcula os vectores propios

Programación en Maxima

Maxima dispón dunha linguaxe de programación propio que permite definir novas funcionalidades.

Exemplo

- Definir a área dun triángulo

```
(%i2) area_tri(base, altura):= base*altura/2;
```

(%o2) $area\_tri(base, altura) := \frac{base\_altura}{2}$

- Definir a área dun cuadrado

```
(%i2) area_cua(lado):= lado^2;
```

(%o2) $area\_cua(lado) := lado^2$

Programación en Maxima: bucles

Para definir un bucle, o máis usual é utilizar a estrutura "for", cuxa sintaxe é

for *NombreVariable:valor inicial* **thru** *valor final de la variable* **do** *acción a realizar*

Exemplo

```
(%i2) for b:0 thru 2 do print(b, "el cuadrado es igual a ", b^2);
0 el cuadrado es igual a 0
1 el cuadrado es igual a 1
2 el cuadrado es igual a 4
(%o2) done
```

Programación en Maxima: bucles

Para definir un bucle, o máis usual é utilizar a estrutura "for", cuxa sintaxe é

for *NombreVariable:valor inicial* **thru** *valor final de la variable* **do** *acción a realizar*

Ademais, a cantidade para incrementar a variable en cada etapa pódese determinar mediante a palabra chave **"step"**.

Exemplo

```
(%i2) for a:0 thru -5 step -2 do display (a);
```

$a = 0$

$a = -2$

$a = -4$

(%o2) *done*

Programación en Maxima: bucles

Para definir un bucle, o máis usual é utilizar a estrutura "for", cuxa sintaxe é

for *NombreVariable:valor inicial* **thru** *valor final de la variable* **do** *acción a realizar*

Tamén se pode introducir nun bucle unha condición de parada utilizando as palabras chaves **while** (mentres que) e **unless** (salvo que).

Exemplo

```
(%i2) for i:4 while i >0 step -1 do print ("i =",i);
```

$i = 4$
$i = 3$
$i = 2$
$i = 1$

```
(%o2) done
```

Programación en Maxima: bucles

Para definir un bucle, o máis usual é utilizar a estrutura "for", cuxa sintaxe é

for *NombreVariable:valor inicial* **thru** *valor final de la variable* **do** *acción a realizar*

Tamén se pode introducir nun bucle unha condición de parada utilizando as palabras chaves **while** (mentres que) e **unless** (salvo que).

Exemplo

```
(%i2) for i:1 unless i^3 >100 do display (i^3);
```

$1^3 = 1$
$2^3 = 8$
$3^3 = 27$
$4^3 = 64$

```
(%o2) done
```

Programación en Maxima: condicional

A sintaxe do condicional

if *condición* **then** *sentencia1* **else** *sentencia2*

Exemplo

```
(%i2) for b:0 thru 5 do if b^2>5 then;
print([b, "válido"]) else print ([b, "no es válido"]);
```

[0, *no es válido*]
[1, *no es válido*]
[2, *no es válido*]
[3, *válido*]
[4, *válido*]
[5, *válido*]
(%o2) *done*

Programación en Maxima: secuencia instrucións

A instrución `block(expr_1, ..., expr_n)` avalía expr_1, ..., expr_ n secuencialmente e devolve o valor da última expresión avaliada.

Exemplo

```
(%i2) (a:2, b:5, c:3, d : a+2*b-c);
(%o2) 9
(%i2) b;
(%o2) 5
(%i2) block([x,y], x:2, y:5, z:x+y);
(%o2) 7
```

Programación en Maxima: secuencia instrucións

Exemplo

- A continuación defínese unha función $f(x)$ e despois outra función, denominada *mifunc*, que recibe como parámetros de entrada entrémolos dun intervalo $[a, b]$ e un enteiro n.
- Defínese unha variable local $h = (b - a)/n$ e se evalua a función f nos puntos da forma $a + i * h$, cando o valor obtido é positivo móstrase por pantalla e en caso contrario imprímese unha mensaxe dicindo que non o é.

Programación en Maxima: secuencia instrucións

Exemplo

```
(%i2) f(x):= x^2-5*x+6;
```

(%o2) $f(x) := x^2 - 5x + 6$

```
(%i2) mifunc(a,b,n):=block(h:(b-a)/n, for i:0 thru n do
if f(a+i*h)>0 then display (f(a+i*h)) else
print ("no es positivo"));
```

(%o2) $mifunc(a, b, n) := block(h : \frac{b-a}{n},$ for i from 0 thru n do if $f(a + i * h) >$ 0 then display $(f(a + i * h))$ else print$(no\ es\ positivo)$)

Programación en Maxima: secuencia instrucións

Exemplo

```
mifunc(1,3,8);
```

$f(1) = 2$
$f(\frac{5}{4}) = \frac{21}{16}$
$f(\frac{3}{2}) = \frac{3}{4}$
$f(\frac{7}{4}) = \frac{5}{16}$
no es positivo
no es positivo
no es positivo
no es positivo
no es positivo
(%o2) *done*

Maxima

Software Libre Científico

María José Ginzo Villamayor, Alejandro Saavedra Nieves e Paula Saavedra Nieves

Departamento de Estatística, Análise Matemática e Optimización
Universidade de Santiago de Compostela

31/03/2022