PROGRAMACIÓN MATEMÁTICA

MAXIMA (apéndice)

Jesús Muñoz San Miguel

PROYECTO MATECO

Un sistema de computación algebraica o CAS (Computer Algebra System) es un programa informático que realiza cálculos simbólicos y consta de:

- Interfaz (*WxMaxima*): introduce los datos y recibe los resultados
- Núcleo (*Maxima*): procesa las órdenes y realiza los cálculos
- Hojas de trabajo (*worksheets*): comunican núcleo e interfaz mediante entradas y salidas con notación similar a la matemática que van siendo numeradas0
- Las entradas se denominan *inputs*
  - Se introducen con la tecla *Intro* <u>del teclado numérico</u>.
  - Si terminamos con un punto y coma se muestra el resultado
  - Si terminamos con un dolar ($) no se muestra el resultado
  - Si hay errores devuelve un mensaje y no realiza cálculos
- Las salidas se denominan *outputs*
  - Nos referimos al output número *n* por %*on* y al último calculado por %
- Las funciones pre-programadas del programa, cuyo nombre corresponde a la acción que hacen (en inglés y en minúsculas), se denominan *comandos*.
- Hay programas que permiten añadir nuevas funcionalidades, reciben el nombre de paquetes y se cargan con el comando *load(paquete)*.
  - Para la representación gráfica usamos el paquete *draw*, precargado en *wxMaxima*, que incluye los gráficos en la hoja de trabajo anteponiendo el prefijo *wx* a los comandos.

Para hacer operaciones básicas los signos **son obligatorios**

- Para sumar y restar usamos los signos + y -
- Para multiplicar y dividir usamos los signos * y /
- Para elevar un número a otro usamos el signo ˆ
- Se antepone un tanto por ciento a las constantes simbólicas
- Para obtener un valor numérico se utiliza el comando *float*
- Para desarrollar los cálculos simbólicos se utiliza el comando *expand*

(%i1) 2+5*3ˆ2;	(%i2)(a+b)ˆ2;	(%i3)expand(%);
47 (%o1)	$(b+a)^2$ (%o2)	$b^2+2ab+a^2$ (%o3)
(%i4) (a+b)*(a-b);	(%i5) expand(%);	(%i6) 2*cos(5*%pi/6);
$(a-b)\,(b+a)$ (%o4)	a^2-b^2 (%o5)	$-\sqrt{3}$ (%o6)
(%i7) float(%);	(%i8) log(2*%e);	(%i9) float(%);
$-0{,}86602540$ (%o7)	$\log(2\,\%e)$ (%o8)	$1{,}693147$ (%o9)

Para las entradas y salidas de algunos comandos se utilizan **listas**, que no son más que conjuntos ordenados de elementos de cualquier tipo encerrados entre corchetes y separados por comas (estos elementos también pueden ser listas)

$$[elem_1, \ldots elem_n]$$

Para extraer sus elementos utilizamos el comando *part*

- $part(list, i)$ extrae el elemento i de la lista $list$ (o $list[i]$)
- $part(list, i, j)$ extrae el elemento j del elemento i si es una lista (o $list[i][j]$).

Podemos generarlas siguiendo un patrón con los comandos *makelist* y *create_list*.

(%i1) makelist (x=i, i, [a, b, c]); /*create_list (x=i, i, [a, b, c]);*/

$$[x = a, x = b, x = c] \quad (\%o1)$$

(%i2) makelist (a^i, i, 1,10); /*create_list (a^i,i,1,10);*/

$$[a, a^2, a^3, a^4, a^5, a^6, a^7, a^8, a^9, a^{10}] \quad (\%o2)$$

(%i3) makelist (a^i, i, 1,10,2); /* con salto (2)*/

$$[a, a^3, a^5, a^7, a^9] \quad (\%o3)$$

(%i4) create_list (i^j, i, 1,3, j, 1,3); /* doble iteración*/

$$[1, 1, 1, 2, 4, 8, 3, 9, 27] \quad (\%o4)$$

Las igualdades se distinguen con distintos símbolos según a que correspondan:

- Para realizar la asignación de variables se utiliza el signo de dos puntos :
  - en una variable podemos almacenar cualquier tipo de elemento

(% i1) a:2;

$$2 \quad (\% o1)$$

(% i2) a^3;

$$8 \quad (\% o2)$$

(% i3) a:a+x;

$$x+2 \quad (\% o3)$$

- Para definir funciones se utiliza el signo igual precedido de dos puntos :=

(% i1) f(x):=x^2+1;

$$\mathrm{f}(x) := x^2 + 1 \quad (\% o1)$$

(% i2) f(2);

$$5 \quad (\% o2)$$

(% i3) 'f(2);

$$\mathrm{f}(2) \quad (\% o3)$$

(% i4) f(x,y):=x*y^2+x^2*y+2*x*y;

$$\mathrm{f}\,(x\,,y) := x\,y^2 + x^2 y + 2xy \quad (\% o4)$$

(% i5) f(1,2);

$$10 \quad (\% o5)$$

(% i6) 'f(1,2);

$$f\,(1,2) \quad (\% o6)$$

- Para construir ecuaciones y sistemas se utiliza el signo igual =

(% i1) x^2-2*x-3=0;

$$x^2 - 2x - 3 = 0 \quad (\% o1)$$

(% i2) [x+y=3,x-y=1];

$$[x+y=3, x-y=1] \quad (\% o2)$$

♦ La mayoría de ecuaciones y sistemas se resuelven con el comando *solve*

(% i1) solve([x^2-2*x-3=0], [x]);

$$[x=3\,, x=-1] \qquad (\%\ o1)$$

(% i2) solve([x+y=3,x-y=1], [x,y]);

$$[[x=2\,, y=1]] \qquad (\%\ o2)$$

▶ En un sistema lineal distingue si es incompatible o compatible, tanto determinado como indeterminado, en cuyo caso introduce los parámetros necesarios.

▶ Hay ecuaciones y sistemas que no se pueden resolver con el comando *solve*. En particular, algunas ecuaciones que envuelven potencias, tanto racionales como no racionales, valores absolutos y funciones trigonométricas.

▶ En este caso se utiliza el comando *to_poly_solve*, que cuando consigue determinar el conjunto de soluciones lo presenta en un conjunto %union.

(% i1) to_poly_solve([x^2-2*x-3=0], [x]);

$$\%union([x=-1],[x=3]) \qquad (\%\ o1)$$

(% i2) to_poly_solve([x+y=3,x-y=1], [x,y]);

$$\%union([x=2,y=1]) \qquad (\%\ o2)$$

▶ Si como entrada aparece una expresión ambos comandos la igualan a cero.

Para sustituir en una expresión podemos indicar los valores de las variables mediante ecuaciones y evaluar la expresión con los comandos *ev*, *at* y *subst*, cuya diferencia se aprecia al sustituir en ciertas expresiones:

(% i1) expr:x^2*y+2*sqrt(x-y);

$$x^2 y + 2\sqrt{x-y} \qquad (\%\ o1)$$

(% i2) subst([x=3,y=1],expr);

$$2^{\frac{3}{2}} + 9 \qquad (\%\ o2)$$

(% i3) at(expr,[x=3,y=1]);

$$2^{\frac{3}{2}} + 9 \qquad (\%\ o3)$$

(% i4) ev(expr,[x=3,y=1]);

$$2^{\frac{3}{2}} + 9 \qquad (\%\ o4)$$

- El comando *ev* realiza la sustitución y luego evalúa la expresión.
- El comando *at* evalúa la expresión y después realiza la sustitución.
- El comando *subst* es sinónimo de *substitute* y si son varias las ecuaciones que indican las sustituciones a realizar estas se sustituyen secuencialmente de izquierda a derecha (para sustituciones en paralelo se utilizan *sublis* y *psubst*).

▷ En todos los casos podemos impedir la evaluación de una expresión anteponiendo un apostrofe (') y forzarla anteponiendo dos ('').

Los vectores se representan por listas, pero las matrices tienen un formato específico y se definen con el comando *Matrix*. Para operar de múltiples formas se utilizan signos o combinaciones de símbolos, en particular: suma de vectores y matrices, "+", y producto escalar de vectores y producto de matrices, "." (punto normal).

(% i1) u:[u1,u2];

$$[u1, u2] \tag{u}$$

(% i2) v:[v1,v2];

$$[v1, v2] \tag{v}$$

(% i3)A:matrix ([a11,a12], [a21, a22]);

$$\begin{pmatrix} a11 & a12 \\ a21 & a22 \end{pmatrix} \tag{A}$$

(% i4) B:matrix ([b11,b12], [b21, b22]);

$$\begin{pmatrix} b11 & b12 \\ b21 & b22 \end{pmatrix} \tag{B}$$

(% i5) u+v; /*suma de vectores*/

$$[v1 + u1, v2 + u2] \tag{\% o5}$$

(% i6) A+B; /*suma de matrices*/

$$\begin{pmatrix} b11 + a11 & b12 + a12 \\ b21 + a21 & b22 + a22 \end{pmatrix} \tag{\% o6}$$

(% i6) u.v; /*producto escalar*/

$$u2\, v2 + u1\, v1 \tag{\% o6}$$

(% i7) A.B; /*producto de matrices*/

$$\begin{pmatrix} a12\, b21 + a11\, b11 & a12\, b22 + a11\, b12 \\ a22\, b21 + a21\, b11 & a22\, b22 + a21\, b12 \end{pmatrix} \tag{\% o7}$$

► De una matriz cualquiera calculamos su traspuesta y el rango con los comandos *transpose* y *rank* (no distingue casos si hay parámetros)

(% i1) M: matrix ([2, 1,1], [2,3,2],[3,3,4]);

$$\begin{pmatrix} 2 & 1 & 1 \\ 2 & 3 & 2 \\ 3 & 3 & 4 \end{pmatrix} \qquad (\% o1)$$

(% i2) transpose(M);

$$\begin{pmatrix} 2 & 2 & 3 \\ 1 & 3 & 3 \\ 1 & 2 & 4 \end{pmatrix} \qquad (\% o2)$$

(% i3) rank(M);

$$3 \qquad (\% o3)$$

► De una matriz cuadrada calculamos el determinante y la inversa con los comandos *determinant* e *invert* (si el determinante es cero da mensaje de error).

(% i4) determinant(M);

$$7 \qquad (\% o4)$$

(% i5) invert(M);

$$\begin{pmatrix} \frac{6}{7} & -\frac{1}{7} & -\frac{1}{7} \\ -\frac{2}{7} & \frac{5}{7} & -\frac{2}{7} \\ -\frac{3}{7} & -\frac{3}{7} & \frac{4}{7} \end{pmatrix} \qquad (\% o5)$$

► De una matriz cuadrada también calculamos sus autovalores y autovectores con los comandos *eigenvalues* y *eigenvectors*

(% i1) M: matrix ([2, 1,1], [2,3,2],[3,3,4]); (% i2) eigenvalues(M);

$$\begin{pmatrix} 2 & 1 & 1 \\ 2 & 3 & 2 \\ 3 & 3 & 4 \end{pmatrix} \qquad (\,\%\ o1)$$

$$([[7\,,1]\,,[1\,,2]] \qquad (\,\%\ o2)$$

(% i3)eigenvectors(M);

$$[[[7\,,1]\,,[1\,,2]]\,,[[[1\,,2\,,3]]\,,[[1\,,0\,,-1]\,,[0\,,1\,,-1]]]] \qquad (\,\%\ o3)$$

Las entradas y salidas de los comandos pueden ser tanto matrices como lista de listas, por lo que necesitamos intercambiar ambos formatos.
Así, para construir una matriz a partir de sus filas utilizamos el comando matrix, que indica a Maxima que tenemos una matriz, y para transformar una matriz en una lista de listas el comando args.

(% i1) A:matrix ([a,b], [c, d]);

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \qquad \text{(A)}$$

(% i2) args(A);

$$[[a, b], [c, d]] \qquad \text{(\% o2)}$$

Podemos construir una matriz mediante una fórmula basándonos en los subíndices de los elementos con el comando genmatrix y extraer partes de matrices con el comando submatrix

(% i1) A:genmatrix(lambda([i,j], a[i,j]), 3, 4);

$$\begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} \end{pmatrix} \qquad \text{(\% o1)}$$

(% i2) submatrix(1,A,2,3);

$$\begin{pmatrix} a_{2,1} & a_{2,4} \\ a_{3,1} & a_{3,4} \end{pmatrix} \qquad \text{(\% o2)}$$

Para calcular las derivadas y derivadas parciales de cualquier orden utilizamos el comando *diff*.

(% i1) f(x):=x^2+3*x+100;

$$f(x) := x^2 + 3 * x + 100 \quad (\% \text{ o1})$$

(% i2) g(x,y):=x*y^2+x^2*y+2*x*y;

$$g(x, y) := x\,y^2 + x^2y + 2xy \quad (\% \text{ o2})$$

(% i3) diff(f(x),x);

$$2 * x + 3 \quad (\% \text{ o3})$$

(% i4) diff(x^3,x,2);

$$6x \quad (\% \text{ o4})$$

(% i5) diff(f(x,y),x);

$$y^2 + 2xy + 2y \quad (\% \text{ o5})$$

(% i6) diff(f(x,y),y);

$$2xy + x^2 + 2x \quad (\% \text{ o6})$$

(% i7) diff(g(x,y),x,2);

$$2y \quad (\% \text{ o7})$$

(% i8) diff(g(x,y),x,1,y,1);

$$2y + 2x + 2 \quad (\% \text{ o8})$$

(% i9) diff(g(x,y),y,1,x,1);

$$2y + 2x + 2 \quad (\% \text{ o9})$$

(% i10) diff(g(x,y),y,2);

$$2x \quad (\% \text{ o10})$$

Para obtener directamente el gradiente y la matriz hessiana utilizamos los comandos *jacobian* y *hessian*

(% i1) f(x,y):=x*y^2+x^2*y+2*x*y;

$$f(x, y) := x\, y^2 + x^2 y + 2xy \quad (\%\ o2)$$

(% i2) jacobian([f(x,y)],[x,y]);

$$\begin{pmatrix} y^2 + 2xy + 2y & 2xy + x^2 + 2x \end{pmatrix} \quad (\%\ o2)$$

(% i3) hessian(f(x,y),[x,y]);

$$\begin{pmatrix} 2y & 2y + 2x + 2 \\ 2y + 2x + 2 & 2x \end{pmatrix} \quad (\%\ o3)$$

Para obtener directamente el desarrollo de Taylor utilizamos el comando *taylor*

(% i1) taylor(sin(x),x,0,5);

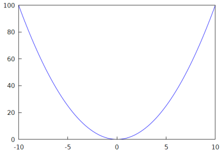
$$x - \frac{x^3}{6} + \frac{x^5}{120} + \ldots \quad (\%\ o1)/T$$

(% i2)taylor(f(x,y),[x,y],[1,1],2);

$$4+5\,(x-1)+5\,(y-1)+(x-1)^2+6\,(y-1)\,(x-1)+(y-1)^2+\ldots \quad (\%\ o10)/T$$

Para representar funciones en dos dimensiones se utiliza el comando *wxdraw2d*, mediante el cual podemos representar funciones explícitas en las que la variable dependiente depende directamente de la función, funciones implícitas en las que tenemos una relación entre las variables dependiente e independiente y funciones en forma paramétrica (válida para cualquier curva descrita por parámetros).

```
( % i1) wxdraw2d(explicit(x^2, x, -10,10))$
( % i2) wxdraw2d(implicit(x^2=y, x, -10,10,y,0,100))$
( % i3) wxdraw2d(parametric(k,k^2,k,-10,10));$
```



Para representar funciones en tres dimensiones se utiliza el comando *wxdraw3d*. La opción *contour* permite dibujar las curvas de nivel sobre la superficie y su proyección perpendicular sobre el plano XY (por separado o conjuntamente).

```
( % i1) wxdraw3d(explicit(x^2+y^2,x, -5,5,y,-5,5))$
( % i2) wxdraw3d(explicit(x^2+y^2,x,-5,5,y,-5,5),contour=map)$
( % i3) wxdraw3d(explicit(x^2+y^2,x, -5,5,y,-5,5),contour=both)$
```