Prácticas de Matemáticas I con Maxima.

Jose Salvador Cánovas Peña.
Departamento de Matemática Aplicada.

Contents

¿Qué es Maxima?

Maxima es un programa que permite hacer cálculos matemáticos complicados con gran rapidez. Para entendernos, es como una especie de calculadora gigante a la que no sólo podemos pedirle que haga cálculos numéricos, sino que también hace derivadas, cálculo de primitivas, representación gráfica de curvas y superficies, factorización de polinomios etc.

Abordamos en esta práctica una iniciación a Maxima partiendo desde cero, e intentaremos poner de manifiesto su utilidad a la hora de trabajar con expresiones matemáticas complicadas, permitiendo hacer éstas con poco coste de tiempo.

Será necesaria por parte del alumno una lectura previa de esta práctica antes de empezar a trabajar con el programa. Esta lectura previa tiene por objeto el conocimiento de ciertas sentencias clave que permiten el manejo del programa. Al igual que al aprender el manejo de una calculadora científica es necesario leer las instruciones de la misma, estas notas pueden ser útiles para aprender el manejo de Maxima.

Por otra parte, a pesar de la potencia evidente del programa, hemos de hacer notar que es necesario por parte del alumno un conocimiento matemático teórico de todas las funciones y sentencias que vamos a usar. Por ejemplo, aunque una calculadora multiplica números con suma facilidad, sólo nos percatamos de su potencia en cuanto conocemos dicha operación y somos capaces de realizarla de un modo mucho más lento. Con Maxima ocurre lo mismo. Sólo conociendo teóricamente las operaciones que Maxima realiza nos percataremos de su indudable utilidad.

Chapter 1

Preliminares

Cuando se arranca Maxima, aparece una pantalla blanca vacía. En ella podemos escribir aquellas operaciones que queremos que realice. Una vez tecleada la operación, hemos de pulsar las teclas *shift + enter* para obtener el resultado. Por ejemplo, supongamos que queremos hacer la operación 2+2. Teclearemos entonces

$$2 + 2$$

en la pantalla. A continuación pulsamos *shift + enter* y aparecerá lo siguiente en pantalla:

$$\begin{aligned} &(\%i1) \quad 2 + 2; \\ &(\%o1) \quad 4 \end{aligned}$$

siendo %i1 la entrada uno, proporcionando la salida %o1.

Además se pueden realizar las siguiente operaciones algebraicas:

$$\begin{aligned} x + y &\rightarrow \textit{suma} \\ x - y &\rightarrow \textit{resta} \\ x/y &\rightarrow \textit{división} \\ x * y &\rightarrow \textit{producto} \\ x\hat{}y &\rightarrow \textit{potencia}. \end{aligned}$$

Actividad 1 *Realizar las siguientes operaciones con Maxima:*

(a) $3.75 + 8.987 =$

(b) $(2-3.1)^{23} =$

(c) $\frac{2.4+3^2}{4*7.2^2} =$

(d) $2 \times 10^2 + 3 \times 10^{-3} =$

(e) $\left(\frac{2.3*4}{2-4.5^2}\right)^{56} =$

A la hora de trabajar con Maxima, hemos de tener en cuenta que hay dos modalidades admisibles. Tomemos por ejemplo la operacion

$$2 \times 10^2 + 3 \times 10^{-3}.$$

Si introducimos la sentencia

$$2 * 10^2 + 3 * 10^{-3}$$

obtendremos al ejecutarla

$$(\%i1) \quad 2 * 10^2 + 3 * 10^{-3};$$
$$(\%o1) \quad \frac{200003}{1000}$$

mientras que si escribimos

$$2.0 * 10^2 + 3 * 10^{-3}$$

obtendremos

$$(\%i2) \quad 2.0 * 10^2 + 3 * 10^{-3};$$
$$(\%o2) \quad 200.003$$

Como vemos, en la primera obtenemos una fracción, mientras en la segunda obtenemos un número decimal. Pero las diferencias van más allá de esta apariencia: en la primera el programa ha trabajado con precisión infinita y el resultado que presenta es exacto. En la segunda se ha trabajado (el programa lo hará automáticamente al detectar un número decimal, 2.0 en este caso) con precición finita y por lo tanto con errores de redondeo. Estos errores no afectan a esta operación, pero sí lo hacen a la operación

$$(1/10)\hat{}4,$$

que en precisión infinita se obtiene

$$(\%i1)\ \ (1/10)\hat{}4;$$
$$(\%o1)\ \ \frac{1}{10000}$$

mientras que en finita tenemos

$$(\%i2)\ \ (1.0/10)\hat{}4;$$
$$(\%o2)\ \ 1.0000000000000005\ 10^{-4}$$

La función **float** se puede utilizar para convertir un número en decimal de doble precisión, que es la que por defecto utiliza el programa. Así por ejemplo

$$(\%i3)\ \ \text{float}((1/10)\hat{}4);$$
$$(\%o3)\ \ 1.0\ \ 10^{-4}$$

Igualmente la operación

$$(\%i4)\ 2\hat{}100;$$
$$(\%o4)\ 1267650600228229401496703205376$$

pero si escribimos

$$(\%i5)\ \text{float}(2\hat{}100);$$
$$(\%o5)\ 1.2676506002282294\ 10^{30}$$

Los números de la forma 3.05×10^{-3} en coma flotante pueden expresarse escribiendo el exponente como "f", "d" o "e". Por ejemplo

$$3.05 \times 10^{-3},\ 3.05e^{-3},\ 3.05d^{-3},\ 3.05f^{-3}$$

Para saber si el programa está trabajando en precisión finita o infinita tenemos la sentencia **numer**. Esta sentencia, si la ejecutamos nos devuelve el valor por defecto **false**, lo que significa que el programa está trabajando en precisión infinita. Si introducimos

$$(\%i6)\ \text{numer:true};$$
$$(\%o6)\ true$$

pasamos a trabajar con precisión finita. Para activar la precisión infinita debemos teclear

$$(\%i7)\ \text{numer:false;}$$
$$(\%o7)\ false$$

Los números complejos se introducen teniendo en cuenta que la unidad imaginaria i se escribe %i. Así, el número complejo $2+3i$ se escribe $2+3*\%i$. Las operaciones con números complejos son análogas a las de con números reales, disponiendose además de las siguientes funciones propias de la naturaleza de los complejos:

- realpart -> Parte real del número complejo.
- imagpart -> Parte imaginaria del número complejo.
- rectform -> Devuelve la expresión rectangular del número complejo.
- polarform -> Devuelve la expresión polar del número complejo.
- cabs -> Devuelve el módulo del número complejo.
- carg -> Devuelve el argumento del número complejo.
- conjugate -> Devuelve el conjugado del número complejo.

No obstante, maxima no ejecuta multiplicaciones y divisiones si no las introducimos dentro de la sentencia **expand**, que se utiliza para desarrollar ciertas operaciones. Por ejemplo

$$(\%i1)\ (2+\%i)*(2-\%i);$$
$$(\%o2)\ (2+\%i)*(2-\%i)$$

mientras que

$$(\%i1)\ \exp and((2+\%i)*(2-\%i));$$
$$(\%o2)\ 5$$

Actividad 2 *Realizar las siguientes operaciones con Maxima:*

(a) $(3+4i)\cdot(8+i) =$

(b) $(3+4i)/(8+i) =$

(c) $(3+4i)\cdot(8+i) - i =$

(d) Obtener los módulos, argumentos, formas polar y rectangular, conjugados y partes real e imaginaria de los números obtenidos en las operaciones (a)–(c) anteriores.

1.1 Sobre las constantes

Las principales constantes matemáticas elementales se teclean en Maxima del siguiente modo

$$\begin{aligned}
\%e- &> e \approx 2.718281828459045d0\\
\%i- &> \sqrt{-1}\\
ind\ o\ und- &> \text{Representa un valor indeterminado}\\
\inf- &> +\infty\\
\min f- &> -\infty\\
\inf inity- &> \infty \text{ complejo (polo norte de la esfera de Reimann)}\\
\%phi- &> \frac{1+\sqrt{5}}{2} \text{ (razón aúrea)}\\
\%pi- &> \pi \approx 3.141592653589793d0
\end{aligned}$$

Aparte de estas constantes, pueden declarar otras que se puedan necesitar en un momento dado. Por ejemplo, si queremos asignar la constante de la gravitación universal $G = 6.67 \times 10^{-11}$ debemos teclear

$$\begin{aligned}
&(\%i1)\ G : 6.67 \times e - 11;\\
&(\%o1)\ 6.67\ 10^{-11}
\end{aligned}$$

Nótese que usamos : en vez del símbolo de igualdad =. Si tecleamos G y pulsamos shift +enter obtendremos en pantalla

$$\begin{aligned}
&(\%i2)\ G;\\
&(\%o2)\ 6.67\ 10^{-11}
\end{aligned}$$

Maxima distingue entre mayúsculas y minúsculas por lo que si escribimos g, esta no es igual a G. Para eliminar el valor asignado a G tenemos la función

kill. Tecleando

$$(\%i3)\ kill(G);$$
$$(\%o3)\ done$$

desposeemos a G de su valor. Si tecleamos como antes obtenemos en pantalla

$$(\%i4)\ G;$$
$$(\%o4)\ G$$

dado que hemos borrado la variable de entre las definidas.

Por otra parte, si en una misma línea queremos definir varias variables, o escribir varias expresiones debemos separar estas con ";". Por ejemplo

$$(\%i5)\ x:1;y:2;z:x+y$$
$$(\%o5)\ 1$$
$$(\%o6)\ 2$$
$$(\%o7)\ 3$$

que como vemos proporciona 3 salidas. Si queremos borrar todos estas variables declearemos

$$(\%i8)\ kill(all);$$
$$(\%o8)\ done$$

Si no deseamos escribir la variable en una salida basta escribir un "$" al final. Por ejemplo

$$(\%i1)\ x:1;y:2;z:x+y\$$$
$$(\%o1)\ 1$$
$$(\%o2)\ 2$$

y no proporciona la tercera salida de la variable z como anteriormente ocurría.

1.2 Sobre las funciones

Una primera apreciación sobre las funciones en maxima es que estas van definidas en minúsculas con los argumentos entre paréntesis. Las nociones matemáticas más notables las presentamos en la siguiente tabla:

$$\begin{aligned}
\text{sqrt}(x) &= \sqrt{x} \\
\exp(x) &= e^x \\
\log(x) &= \log x \\
\sin(x) &= \sin x \\
\cos(x) &= \cos x \\
\tan(x) &= \tan x \\
\text{asin}(x) &= \arcsin x \\
\text{acos}(x) &= \arccos x \\
\text{atan}(x) &= \arctan x \\
n! &= \textit{factorial de } n \\
\text{abs}(x) &= |x| \\
\text{entier}(x) &= \textit{parte entera de } x
\end{aligned}$$

Así, si escribimos

```
(%i1) sqrt(16);
(%o1) 4
(%i2) Sqrt(2);
(%o2) sqrt(2)
(%i3) float(sqrt(2));
(%o3) 1.414213562373095
```

Actividad 3 *Calcular los siguientes valores:*

(a) $\sin \pi/4 + \cos \pi/7 =$

(b) $\log_2 256 =$

(c) $\left|\arcsin 0.98 + \frac{\log 2}{\sqrt{2}}\right| =$

(d) $e^{10!} =$

(e) $\sqrt{\log 34 + e^{12}} =$

Aparte de las funciones que Maxima tenga integradas, podemos introducir nuevas funciones con la sentencia **define**. Su estructura es

$$define(f(x\_1, ..., x\_n), expr),$$

que definirá una función de nombre f dependiente de las variables $x\_1, ..., x\_n$ según la expresión $expr$, que puede ser escalar o vectorial. Por ejemplo, si queremos definir la función $f(x, y) = xy$, escribimos

$$(\%i1)\ define(f(x, y), x * y);$$
$$(\%o1)\ f(x, y) := x * y$$

con lo que la función estará introducida. Si ahora tecleamos

$$(\%i2)\ f(2, 3);$$
$$(\%o2)\ 6$$

Las funciones también pueden definirse de una forma más cómoda con la expresión

$$(\%i3)\ f(x, y) := x * y$$

1.3 Aprovechando cálculos anteriores

A veces, es posible que tengamos que hacer una sucesión de cálculos consecutivos de manera que cada nueva operación se basa en la anterior. Parece necesaria entonces algo que nos remita a resultados anteriores. Esto se realiza con maxima simplemente llamando a la entrada o salida correspondiente anteponiendo siempre el símbolo %. Por ejemplo, si queremos calcular $\cos(\sin \pi/7)$ tendríamos que calcular primero $\sin \pi/7$, para despues calcular el coseno de dicha cantidad. Esta operación podríamos hacerla del modo siguiente:

$$(\%i1)\ \sin(\%pi/7);$$
$$(\%o1)\ \sin\left(\frac{\pi}{7}\right)$$
$$(\%i2)\ \cos(\%o1)$$
$$(\%o2)\ \cos\left(\sin\left(\frac{\pi}{7}\right)\right)$$
$$(\%i3)\ \text{float}(\%o2)$$
$$(\%o3)\ 0.90733988115078$$

Obviamente, este ejemplo es bastante sencillo ya que la operación en cuestión podría haberse hecho en una sola línea de comandos, pero ilustra bien el modo de proceder cuando se estén realizando operaciones y cálculos más complejos. Finalmente, para llamar a la salida anterior basta usar el símolo %. Así, la operación anterior podría haberse escrito como

$$
\begin{aligned}
&(\%i1)\ \sin(\%pi/7);\\
&(\%o1)\ \ \sin\left(\frac{\pi}{7}\right)\\
&(\%i2)\ \cos(\%)\\
&(\%o2)\ \ \cos\left(\sin\left(\frac{\pi}{7}\right)\right)\\
&(\%i3)\ \ \mathrm{float}(\%)\\
&(\%o3)\ 0.90733988115078
\end{aligned}
$$

Chapter 2

Algebra lineal con Maxima

Una vez introducidas ciertas generalidades sobre el programa, vamos a ver cómo podemos utilizar el éste para agilizar algunos cálculos relativos a los contenidos explicados en la parte teórica de la asignatura.

2.1 Resolución de sistemas de ecuaciones lineales

Si tenemos que resolver un sistema de ecuaciones lineales de la forma $AX = b$ donde A es una matriz de n filas por m columnas, X es una vector columna incógnita de n componentes y b es un vector de m componentes, disponemos de la sentencia

$$linsolve([expr\_1, ..., expr\_m], [x\_1, ..., x\_n]),$$

donde expr_1,...,expr_m son las m ecuaciones lineales que definen el sistema y x_1,...,x_n son las incógnitas del mismo. Para introducir las ecuaciones utilizamos el signo de igualdad =, como es usual. Por ejemplo, si queremos resolver el sistema

$$\left.\begin{array}{l} x + y + z = 1 \\ x + 2y = 1 \end{array}\right\}$$

escribiremos

$$(\%i1)\ lin\,\text{solve}([x + y + z = 1, x + 2 * y = 1], [x, y, z]);$$
$$(\%o1)[x = 1 - 2 * \%r1, y = \%r1, z = \%r1]$$

con lo que la solución del sistema depende del parámetro $r1$ y presenta la forma paramétrica

$$\begin{cases} x = 2t, \\ y = t \quad \text{donde } t \in \mathbb{R}. \\ z = t, \end{cases}$$

Dentro de la resolución de ecuaciones existen una serie de alternativas que a continuación pasamos a describir. Las alternativas vienen dados por funciones que tienen asignados el valor true si están activadas y false si no lo están. Según esten activadas o no dichas funciones el programa efectúa las operaciones y presenta resultados de una u otra forma. Para cambiar la asignación true o false basta con teclear

$$(i\%1)\ nombre\_funcion : true;$$
$$(o\%1)\ true$$

que asigna a la funcion el valor true y el correspondiente

$$(i\%1)\ nombre\_funcion : false;$$
$$(o\%1)\ false$$

para false. Para ver qué valor tiene una función en un determinado momento basta con ejecutarla. Veamos los parámetros o funciones más relevantes.

- **linsolve_params** Valor por defecto: true. Si linsolve_params es true, la función linsolve genera símbolos %r para representar parámetros arbitrarios para representar la solución de forma paramétrica. Si vale false, el resultado devuelto para un sistema es indeterminado elimina las ecuaciones dependientes y se expresa con la forma general. Por ejemplo

$$(\%i1)\ lin\,\text{solve}\ \_params : false;$$
$$(o\%1)\ false$$
$$(\%i2)\ lin\,\text{solve}([x + y + z = 1, x + 2 * y = 1], [x, y, z]);$$
$$(\%o2)[x = 1 - 2 * z, y = z]$$

- **globalsolve** Valor por defecto: false. Si se activa a true, al resolver el sistema asigna a las incógnitas el valor de las soluciones, de igual forma

que se introducen las constantes. Por ejemplo

$$(\%i1)\ global\,\text{solve} : true;$$
$$(o\%1)\ true$$
$$(\%i2)\ lin\,\text{solve}([x+y+z=1, x+2*y=1],[x,y,z]);$$
$$(\%o2)[x : 1-2*\%r1, y : \%r1, z : \%r1]$$

- **programmode** Valor por defecto: true. Si cambiamos a false, linsolve muestra la solución con etiquetas de expresiones intermedias (%t). Por ejemplo

$$(\%i1)\ programmode : false;$$
$$(o\%1)\ false$$
$$(\%i2)\ lin\,\text{solve}([x+y+z=1, x+2*y=1],[x,y,z]);$$
$$(\%t2)\ x = 1-2*\%r1$$
$$(\%t3)\ y = \%r1$$
$$(\%t4)\ z = \%r1$$
$$(\%o2)[\%t2, \%t3, \%t4]$$

Finalemente, se admiten diferentes combinaciones de las funciones anteriores, es decir, algunas de ellas con valor false y otras true, lo que da lugar a diferentes maneras de representar las soluciones y ejecutar la funcion linsolve.

Actividad 4 *Resolver los siguientes sistemas de ecuaciones lineales:*

$$(a)\ \begin{cases} x-y=-1 \\ -x+y=1 \\ 2x-2y=-2 \end{cases} \qquad (b)\ \begin{cases} 2x+y+4z=0 \\ x-y+2z=4 \\ 2x+y-z=14 \\ 3x+z=18 \end{cases}$$

$$(c)\ \begin{cases} 2x+2y-3z=2 \\ -x+5y-4z=4 \\ x+7y-7z=7 \end{cases} \qquad (d)\ \begin{cases} x+2y+3z=0 \\ 2x+2y+3z=0 \\ 3x+2y+z=0 \end{cases}$$

$$(e)\ \begin{cases} x+2y-3z+16t=4 \\ y+2z-3t=6 \\ -x-y+z+9t=-2 \end{cases} \qquad (f)\ \begin{cases} x-2y+3z=0 \\ 2x+5y+6z=0 \end{cases}$$

2.1.1 Resolución de ecuaciones polinómicas

Supongamos que queremos resolver la ecuación polinómica $x^2 + 2x - 4 = 0$. El comando para resolverlas en Maxima es **solve**, de forma que para resolver la ecuación anterior escribiremos

$$(\%i1)\ \text{solve}(x\hat{}2 + 2 * x - 4 = 0, x);$$
$$(\%o1)\ [x = -\sqrt{5} - 1, x = \sqrt{5} - 1]$$

Si lo que buscamos son soluciones numéricas aproximadas, tanto reales como complejas, tenemos la sentencia **allroots**, de forma que al teclear

$$(\%i2)\ \text{allroots}(x\hat{}2 + 2 * x - 4 = 0, x);$$
$$(\%o2)\ [x = 1.23606797749979, x = -3.23606797749979]$$

obtenemos las soluciones aproximadas.

Actividad 5 *Calcular las raíces de los siguientes polinomios, dando el resultado exacto si lo hubiere, y un resultado aproximado.*

(a) $x^3 + 3x^2 - x - 1$.

(b) $x^{10} - 1$.

(c) $5x^4 - x^2 + x - 1$.

(d) $x^5 - x^3 + 1$.

2.2 Vectores y matrices

Para escribir vectores de $\mathbb{K}^n$ debemos de introducir cada una de las componentes del mismo entre corchetes, separándolas por comas. Así, el vector $v = (1, 2, -1, 5)$ de $\mathbb{R}^4$ se tecleará

$$(\%i1)\ v : [1, 2, -1, 5];$$
$$(\%o1)\ [1, 2, -1, 5]$$

Las matrices se introducirán por filas como un vector de vectores, ayudados de la sentencia **matrix**. Así la matriz $M = \begin{pmatrix} 1 & 2 & -1 \\ 0 & 3 & 1 \end{pmatrix}$ se escribirá

como

$$(\%i2)\ M : matrix([1,2,-1],[0,3,1]);$$
$$(\%i2)\ \begin{bmatrix} 1 & 2 & -1 \\ 0 & 3 & 1 \end{bmatrix}$$

Para la matriz identidad de $M_{n\times n}(\mathbb{K})$ tenemos asignado el nombre

$$ident(n),$$

mientras que tecleando

$$diagmatrix(n,x)$$

introducimos una matriz diagonal de tamaño $n \times n$ con el valor x en cada elemento de la diagonal principal. Si tenemos instroducida la matriz M, las sentencia

$$row(M,i)$$

devuelve la fila i-esima (con formato de matriz), mientras que

$$\mathrm{col}(M,i)$$

devuelve la columna i–esima. Por ejemplo

$$(\%i1)\ M : matrix([1,2,-1],[0,3,1]);$$
$$(\%i1)\ \begin{bmatrix} 1 & 2 & -1 \\ 0 & 3 & 1 \end{bmatrix}$$
$$(\%i2)\ \mathrm{col}(M,2);$$
$$(\%o2)\ \begin{bmatrix} 2 \\ 3 \end{bmatrix}$$
$$(\%i3)\ \mathrm{row}(M,1);$$
$$(\%o3)\ [1\ 2\ -1]$$

Otra sentencia interesante es

$$entermatrix(n,m)$$

con la que se introduce de forma interactiva una matriz $n \times m$. Si $n = m$ (matriz cuadrada), la función pregunta primero si la matriz que se va a introducir es diagonal, simétrica, antisimétrica o general, y una vez contestado se van introduciendo los valores de la misma.

2.3 Operaciones con vectores

Dados dos vectores v_1 y v_2 y un número real λ, se definen con Mathematica las siguientes operaciones:

$v_1 + v_2$	suma de vectores
$\lambda * v_1$	multiplicación del vector v_1 por λ
$v_1.v_2$	producto escalar de v_1 por v_2

Por ejemplo, si introducimos los vectores de $\mathbb{R}^3$ $v = (1, 2, 1)$ y $u = (0, 1, 0)$ podemos calcular la suma de ambos, $2v$ y el producto escalar de ambos vectores del siguiente modo:

$(\%i1)$ $v : [1, 2, 1]; u : [0, 1, 0];$
$(\%o1)$ $[1, 2, 1]$
$(\%o2)$ $[0, 1, 0]$
$(\%i3)$ $u + v;$
$(\%o3)$ $[1, 3, 1]$
$(\%i4)$ $2 * v;$
$(\%o4)$ $[2, 4, 2]$
$(\%i5)$ $u.v;$
$(\%o5)$ 2

Actividad 6 *Dados los vectores* $v = (1, 2, 3, 4)$, $u = (3, -1, 0, 2)$ *y* $w = (2, 0, 0, 1)$ *calcular:*

(a) $(w.v)u$ *(b)* $u - 2v + 3w$ *(c)* $(2u + v).w$

(d) $\|w\|$ *(e)* $\|u - v\|$ *(f)* $u - v + w$

2.4 Operaciones con matrices

Dadas dos matrices A, B definidas en Mathematica, podemos realizar las siguientes operaciones con ellas, siempre que éstas puedan hacerse:

$A + B$	suma de las matrices
$A.B$	producto de las matrices
$\lambda * A$	producto del número λ por la matriz A
transpose(A)	traspuesta de A
determinant(A)	determinante de A
invert(A)	inversa de A

Si por ejemplo tenemos las matrices

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

y

$$B = \begin{pmatrix} 2 & 1 & -1 \\ 0 & -1 & 0 \\ 1 & 2 & 3 \end{pmatrix},$$

podemos calcular su suma, producto, A^t, $|A|$ y A^{-1} de la siguiente forma:

$$(\%i1)\ A : matrix([1,2,1],[1,0,1],[0,1,1])\$;$$
$$(\%i2)\ B : matrix([2,1,-1],[0,-1,0],[1,2,3])\$;$$
$$(\%i3)\ A + B;$$
$$(\%o3)\ \begin{bmatrix} 3 & 3 & 0 \\ 1 & -1 & 1 \\ 1 & 3 & 4 \end{bmatrix}$$
$$(\%i4)\ transpose(A)$$
$$(\%o4)\ \begin{bmatrix} 1 & 1 & 0 \\ 2 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$
$$(\%i5)\ determinant(A)$$
$$(\%o5)\ -2$$
$$(\%i6)\ invert(A)$$
$$(\%o6)\ \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & -1 \\ \frac{1}{2} & -\frac{1}{2} & 0 \\ -\frac{1}{2} & \frac{1}{2} & 1 \end{bmatrix}$$

Actividad 7 *Dados las matrices* $A = \begin{pmatrix} 1 & 2 & 4 \\ 2 & 3 & -1 \end{pmatrix}$, $B = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$, $C = \begin{pmatrix} -1 & 1 \\ 2 & -1 \end{pmatrix}$ *y* $D = \begin{pmatrix} 0 & -1 & 0 \\ 2 & 0 & 5 \\ 2 & 1 & 0 \end{pmatrix}$ *calcular:*

(a) CAB *(b)* $A(B+C)$ *(c)* $(B^t + C)$

(d) CAB^tD^{-1} *(e)* AD^tB^{-1} *(f)* $B^{-1}D$

Actividad 8 *Calcular el determinante de las siguientes matrices cuadradas*

(a) $\begin{vmatrix} 3-i & 5 & 7 & 2 \\ 2-6i & 4 & 1 & 1 \\ -2 & 0 & 0 & 0 \\ 1 & 1-i & 3 & 4 \end{vmatrix}$ *(b)* $\begin{vmatrix} 1 & \cos x & \cos 2x \\ \cos x & \cos 2x & \cos 3x \\ \cos 2x & \cos 3x & \cos 4x \end{vmatrix}$ *(c)* $\begin{vmatrix} x & a & b & c \\ a & x & 0 & 0 \\ b & 0 & x & 0 \\ c & 0 & 0 & x \end{vmatrix}$

Actividad 9 *Calcular el rango de las siguientes matrices*

$$(a)\begin{pmatrix} 2 & 3 & 4 \\ 5 & 6 & 7 \\ 8 & 9 & 0 \\ 1 & 2 & 3 \end{pmatrix} \quad (b)\begin{pmatrix} 2 & 2 & 3 \\ 3 & 0 & 1 \\ 10 & 4 & 8 \end{pmatrix} \quad (c)\begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 0 & 3 & 4 \\ 2 & 3 & 0 & 4 \\ 2 & 3 & 4 & 0 \end{pmatrix}$$

2.5 Diagonalización de matrices cuadradas

Dada una matriz cuadrada $A \in M_{n\times n}(\mathbb{R})$ es posible a veces encontrar una matriz diagonal $D \in M_{n\times n}(\mathbb{R})$ de manera que existe una matriz invertible $P \in M_{n\times n}(\mathbb{R})$ de forma que

$$D = PAP^{-1}.$$

Los elementos de la matriz diagonal D se llaman vectores propios de A, y las matrices P y P^{-1} son matrices de cambio de base que nos pasan de la matriz A a la D. La ventaja de tener matrices diagonales es que éstas son muy fáciles de manejar. Hemos de remarcar que no siempre existe la forma diagonal de una matriz.

El polinomio caracteristico de A se puede obtener con la sentencia

$$charpoly(A, x)$$

donce x indica la variable en la que se construirá.

Maxima dispone de sentencias que permiten calcular rápidamente la forma diagonal de la matriz, en caso de que ésta exista, proporcionando además una base de vectores propios. También dispone de una sentencia para calcular la forma diagonal, y en caso de que ésta no exista, una forma canónica lo más parecido posible a la forma diagonal. Consideremos una matriz cuadrada arbitraria A. Las sentencias que vamos a usar son las siguientes:

- eigenvalues(A). Con esta sentencia se calculan los valores propios de la matriz A. El resultado ofrece dos listas, la primera con los valores propios y la segunda con sus respectivas multiplicidades.

- eigenvectors(A). Calcula los vectores propios de la matriz A. El resultado devuelto es una lista con dos elementos; el primero está formado por dos listas, la primera con los valores propios de A y la segunda

con sus respectivas multiplicidades, el segundo elemento es una lista de listas de vectores propios, una por cada valor propio, pudiendo haber uno o más vectores propios en cada lista.

Por ejemplo, si consideramos la matriz $A = \begin{pmatrix} 3 & -2 & -1 \\ -1 & 4 & 1 \\ 1 & 2 & 5 \end{pmatrix}$ podemos calcular

$$(\%i1)\ A : matrix([3,-1,-1],[-1,4,1],[1,2,5])\$;$$
$$(\%i2)\ eigenvalues(A);$$
$$(\%o2)[[6,3],[1,2]]$$
$$(\%i3)\ eigenvectors(A);$$
$$(\%o3)\ [[[6,3],[1,2]],[[[1,-\frac{4}{3},-\frac{5}{3}]],[[0,1,-1]]]]$$

Actividad 10 *Calcular los vectores y valores propios, y en caso de existir, la forma diagonal de las matrices B y D del ejercicio 7.*

2.6 Optimización lineal

Maxima permite resolver problemas de optimización lineal mediante el algoritmo del simplex. Para ello, hay que cargar un paquete especial, **simplex**, desarrollado para este menester, ejecutando la sentencia (añadimos $ al final para que no proporcione salida)

$$(i\%1)\ load("simplex")\$$$

Una vez caragdo, disponemos de los siguientes comandos:

- **linear_program.** Este comando tiene la sintaxis

  $$linear\_program(\mathbf{A},\mathbf{b},\mathbf{c})$$

  donde **A** es una matriz $n \times m$, y **b** y **c** son vectores de dimensiones n y m, respectivamente. Dicha sentencia minimiza la función objetivo

  $$\mathbf{c} \cdot \mathbf{x}$$

bajo las restricciones

$$\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$$

y si $\mathbf{x} = (x_1, ..., x_m)$, entonces $x_i \geq 0$, $i = 1, ..., m$. Por ejemplo, si tecleamos

$(i\%2)$ $A : matrix([1, 1, -1, 0], [2, -3, 0, -1], [4, -5, 0, 0])\$$
$(i\%3)$ $b : [1, 1, 6]\$$
$(i\%4)$ $c : [1, -2, 0, 0]\$$
$(\%i5)$ $linear\_program(A, b, c);$
$(\%o5)$ $[[\frac{13}{2}, 4, \frac{19}{2}, 0], \frac{3}{2}]$

que nos minimiza la función

$$f(x_1, x_2, x_3, x_4) = x_1 - 2x_2$$

con las restricciones anteriormente comentadas. El mínimo, con valor $\frac{3}{2}$, se alcanzará en el punto $(\frac{13}{2}, 4, \frac{19}{2}, 0)$.

- **minimize_lp**. Esta función permite encontrar los mínimos de la función objetivo (obj), bajo unas condiciones o restricciones lineales (cond), con argumentos opcionales (pos), según la siguiente sintaxis

$$\min imize\_lp(obj, cond, [pos]).$$

Dará las salidas "Problem not bounded!" si el problema es no acotado y "Problem not feasible!" si es no factible. Al contrario que en la sentencia anterior, no se suponen que todos los valores son no negativos. Si queremos que así sea, utilizaremos la sentencia opcional **nonegative_lp** asignándole el valor true.

Por ejemplo

$(\%i6)$ $minimize\_lp(x + y, [3 * x + y = 0, x + 2 * y > 2]);$
$(\%o6)$ $[4/5, [y = 6/5, x = -2/5]]$

$(\%i7)$ $nonegative\_lp : true;$
$(\%o7)$ $true$
$(\%i8)$ $minimize\_lp(x + y, [3 * x + y > 0, x + 2 * y > 2])$
$(\%o8)$ $[1, [y = 1, x = 0]]$

(%i9) $minimize\_lp(x+y,[3*x+y=0,x+2*y>2])$;
(%o9) "Problem not feasible!"

(%i10) $nonegative\_lp : false$;
(%o11) $false$
(%i12) $minimize\_lp(x+y,[3*x+y>0])$
(%o12) "Problem not bounded!"

- **maximize_lp**. Es análoga a la función anterior para obtener máximos.

Actividad 11 *Encontrar la solución de los siguientes problemas:*

$$(a)\begin{cases} \min imizar & x+y+z \\ suejto\ a & x+2*y>2 \\ & x,y,z\geq 0 \end{cases} \qquad (b)\begin{cases} \max imizar & x+2y+z \\ suejto\ a & x+y-z>2 \end{cases}$$

$$(c)\begin{cases} \min imizar & 2x+y \\ suejto\ a & x+2*y=2 \\ & x-y\geq 0 \end{cases} \qquad (d)\begin{cases} \max imizar & x+2y+z \\ suejto\ a & x+y-z>2 \\ & x,y,z\leq 0 \end{cases}$$

$$(e)\begin{cases} \min imizar & x+y+z \\ suejto\ a & x+2*y=2 \\ & x-y-z=0 \end{cases} \qquad (f)\begin{cases} optimizar & x+2y-z \\ suejto\ a & x-z>2 \\ & x\leq 0 \end{cases}$$