Apuntes de Maxima

José Ezequiel Gallardo Marín

Marzo 2018

1. Instalación

1.1. Instalación rápida

Para instalar Maxima rápidamente tan solo tendremos que ir a la siguiente dirección

http://maxima.sourceforge.net/download.html

En esta página encontramos directamente el binario para ejecutar la instalación, incluso podemos instalarlo para el sistema operativo Android.

1.2. Instalación en GNU/Linux desde código fuente enfocada a Debian

Si por alguna razón queremos compilar la fuente lo primero es descargar de la forja el programa Maxima en su versión más reciente en el siguiente link

https://sourceforge.net/projects/maxima/files/latest/download

Abrimos la terminal, navegamos al destino deseado y mediante el programa `tar` extraemos el contenido en dicha ubicación tal y como sigue

```
$ cd  /FisicaComputacional/Programas
$ tar -vxf  /Descargas/maxima-5.41.0.tar.gz
```

1.2.1. Compilando Maxima

El siguiente paso es compilar Maxima. Este puede ser compilado con SBCL, Clisp, CCL, GCL, CMUCL, ECL, Scieneer Common Lisp (SCL) o Allegro Common Lisp (ACL). Yo he elegido Clisp para compilar Maxima. Si no está instalado en nuestro sistema podemos instalarlo mediante Aptitude o Synaptic. Usando la siguiente orden podríamos instalar Clisp

$ sudo apt-get install clisp

Si por cualquier motivo no es posible instalarlo desde nuestra interfaz podemos descargar los paquetes desde la forja, siguiendo el enlace de abajo

https://clisp.sourceforge.io/

Dentro de la página miramos en el frame de la derecha titulado "Get CLISP" y un poco más abajo en ese mismo frame donde pone "Linux packages" están los enlaces dependiendo de nuestra distribución para descargar los paquetes necesarios. En mi caso, mi distribución está basada en debian y por tanto hago clic en debian lo cual nos lleva a la siguiente dirección

https://packages.debian.org/search?keywords=clisp

En la página web, el paquete buscado debe aparecer el primero en la búqueda bajo el nombre "Package clisp". Haríamos clic en "wheezy (oldoldstable)" ya que es la única versión estable que aparece.

En la siguiente página hacemos scroll hasta el final donde está el apartado de descargas que aparece titulado como "Download clisp" y elegimos la arquitectura de nuestro ordenador haciendo clic sobre ella, en mi caso amd64

Por último, se nos presentará una página desde donde podemos descargar el paquete haciendo clic en cualquiera de los enlaces espejos que se presentan ubicados alrededor del mundo.

Con el paquete ya descargado hacemos uso de la terminal para instalarlo en el sistema mediante el comando `dpkg` como sigue

sudo dpkg –install /home/Usuario/Descargas/clisp_2.49-8.1_amd64.deb

Si se encuentran dependencias sin satisfacer tendremos que instalarlas bien mediante aptitude o synaptic, o bien de forma manual yendo a la web https://packages.debian.org.

Normalmente las dos dependencias que suelen faltar son `libffcall1` y `libreadline5` cuyos paquetes podemos encontrar en las siguientes direcciones

https://packages.debian.org/wheezy/libffcall1
https://packages.debian.org/wheezy/libreadline5

1.2.2. Instalando Maxima

Volvamos ahora a la instalación de Maxima yendo a la ubicación donde extraímos el archivo tar

$ cd /FisicaComputacional/Programas/maxima-5.41.0

Usando GNU Autotools introducimos los siguientes comandos

$ ./configure
$ make
$ make check
$ sudo make install

Por último iniciamos el programa en la terminal escribiendo en ella la orden `maxima`.

2. Primeros pasos

Abrimos una terminal e introducimos el comando `maxima` y pulsamos `enter` para que comience a ejecutarse el programa.

Cada operación debe finalizar con un `';'`. A continución, pulsamos `enter` para que se lleve a cabo.

Por ejemplo realizamos la siguiente suma:

(%i1) 35+65;

(%o1) 100

La letra `i` en `(%i1)` significa *input* y por tanto es donde tecleamos nuestra operación.

La letra `o` en `(%o1)` significa *output* e indica la salida de la operación.

2.1. Variables

Para almacenar números y operaciones usamos los dos puntos `':'` y *no* el signo igual `'='`, ya que este último se usa para ecuaciones, por ejemplo

(%i2) x:10;

(%o2) 100

(%i3) y:x*2;

(%o3) 200

Acabamos de almacenar en `'x'` el número 100 y en la variable `'y'` el doble del valor de `'x'`, es decir 200.

Para reiniciar Maxima usamos la orden kill(all). Esto hace que se limpie la memoria y por tanto que se pierdan todas las asignaciones que hemos realizado. Además el programa empieza desde el input 1, (%i1).

2.2. Operaciones básicas

2.2.1. Sumar y restar

Vamos a almacenar en las variables ‘x‘ e ‘y‘ los valores 73 y 96 y vamos a sumarlos y a restarlos, pero lo vamos a realizar todo el input en una sola línea

(%i1) x:73; y:96; x+y; x-y;
(%o1) 73
(%o2) 96
(%o3) 169
(%o4) -23

2.2.2. Multiplicación y división

Aprovechando que tenemos los valores almacenados en ‘x’ e ‘y’ relizamos igual que antes operaciones con estas variables, ahora multiplicando y dividiendo por diversas expresiones, por ejemplo

(%i5) 10*x; y/2; x*y; y/x;
(%o5) 73
(%o6) 96
(%o7) 169
(%o8) -23

Como observación, hay que usar siempre el símbolo ‘*’ para multiplicar. Es decir, si en vez de teclear ‘2*x’ tecleamos ‘2x’ entonces nos saldrá un error. Y si tecleamos ‘xy’ en vez de ‘x*y’ entonces Maxima interpreta que ‘xy’ es una variable. Así es, las variables pueden ser cadenas de caracteres también, por ejemplo

(%i9) juan:2; juan*5;
(%o9) 2
(%o10) 10

Hay que tener cuidado de no usar variables que ya han sido asignadas sin darnos cuenta porque esto inducirá errores en nuestras operaciones.

2.2.3. Potencias

Para elevar un número o expresión a una potencia cualquiera usaremos el acento circunflejo '^'.

```
(%i11) y^2;
(%o11) 9216
```

La raíz cuadrada podemos hacerla de dos formas posibles, elevando a un medio la expresión o usando la función sqrt()

```
(%i12) 81^(1/2);sqrt(9);
(%o12) 9
(%o13) 3
```

2.3. Órdenes útiles de Maxima

Maxima nos da la posibilidad de hacer referencia a una expresión anterior haciendo uso de la numeración de los inputs y los outputs. Por ejemplo si queremos hacer el logaritmo natural del resultado que ha salido en el output número 12 hacemos lo siguiente

```
(%i14) log(%o12)
(%o14) log(9)
```

3. Álgebra lineal con Maxima

3.1. Operaciones con matrices

Definir matriz
A:matrix([1,2,1],[2,1,0],[0,3,1]);
B:matrix([2,1,0],[2,5,1],[3,2,4]);
Suma A+B
Resta A-B
Multiplicación entre matrices A.B
Potencia A^2
Transpuesta transpose(B)
Inversa invert(A)
Mas ordenes utiles
A[i, j] nos da el elemento de la matriz A que está en la posición (i, j).
matrix_size(A) devuelve una lista con el número de filas y columnas de la matriz A.

submatrix(i1, . . . , im, A, j1, . . . , jn): devuelve la submatriz de A que resulta al eliminar las filas i1, . . . , im y las columnas j1, . . . , jn.

A[i]: devuelve la i-ésima fila de la matriz A.

col(A,i): devuelve la i-ésima columna de la matriz A.

addrow(A, fila1, . . . ,filan): añade las filas dadas por las listas (o matrices fila) a la matriz A.

addcol(A, col1, . . . ,coln): añade las columnas dadas por las listas a la matriz A.

determinant(A): calcula el determinante de la matriz A.

ident(n): devuelve la matriz indentidad de orden n.

3.2. Transformaciones elementales por filas

rowswap(A,i,j): intercambia las filas i y j de la matriz A

rowop(A,i,j,$-\alpha$): trasnformacion $F_i + \alpha F_j$. Devuelve la matriz equivalente con la fila F_i actualizada

Para transformaciones tipo αF_j haremos: $M : 1 * A\ M[j] : \alpha * A[j]$ M;

Por ejemplo

$M : 1 * A\ M[1] : 2 * A[1]$ M;

Multiplica por dos la primera fila de la matriz A.

3.3. Matriz escalonada por filas

Para hallar una matriz escalonada por filas equivalente por filas a A usamos la orden echelon(A) Por ejemplo C:matrix([5,1,2],[4,1,0],[1,0,1]); echelon(C);

Para la hallar la forma normal de Hermite por filas descargaos el fichero normalHermite.mac load(/ruta/normalHermite.mac); normalHermiteFilas(C); Devuelve en este caso la matriz identidad de orden 3

3.4. Sistemas de ecuaciones lineales

Para resolver sistemas lineales de ecuaciones usamos la orden linsolve linsolve([x-2*y-z = 1,x-y=1,2*x+y-3*z=4],[x,y,z]); insolve([x-2*y-z = 1,x-y=1,-z-3*y+2*x=2],[x,y,z]); solve: dependent equations eliminated: (3) [x = 1 − %r1, y = − %r1, z = %r1] Obsérvese que los parámetros que aparecen en la solución de sistemas compatibles indeterminados se denotan en la forma %r1, %r2, ...

3.5. Factorización LU de matrices cuadradas

Descargamos el fichero factorizacionLU.mac y lo cargamos en Maxima con la orden load(/ruta/factorizacionLU.mac); Invocando la orden factoriza-LU(A); Nos devuelve las matrices L y U