Universidad Tecnológica Nacional
Facultad Regional Buenos Aires

ÁLGEBRA Y GEOMETRÍA ANALÍTICA EN wxMaxima

Primera Parte

Laboratorio de Matemática
UDB Matemática
2011

Realizado por: *Daniela Scarcella*
Supervisado por: *Susana Estévez*

Introducción

MAXIMA, distribuido bajo licencia GNU-GPL, es un programa escrito en Lisp con especial énfasis en computación simbólica, tanto el código fuente como los manuales son de libre acceso a través de la página Web del proyecto: http://maxima.sourceforge.net

Uno de los aspectos más relevantes de este programa es su naturaleza libre; la licencia GPL en la que se distribuye brinda al usuario ciertas libertades: Libertad para utilizarlo, modificarlo y adaptarlo a sus propias necesidades, para distribuirlo, para estudiarlo y aprender su funcionamiento.

Debemos aclarar que las líneas con etiquetas `(%i…)` corresponden a entradas del usuario y las líneas con etiquetas `(%o…)` representan las salidas de Maxima.

Vectores

Usualmente un vector lo definimos de la forma: $\vec{v} = (v_1, v_2, v_3)$, pero en Maxima se escriben corchetes en vez de paréntesis y el operador igual (=) se reemplaza por los dos puntos (:). Por ejemplo el vector $\vec{v} = (1, 0, 2)$ se escribe de la siguiente manera: v:[1,02] con.

Veamos a continuación un ejemplo de cómo se define el vector $\vec{v} = (1, 0, 2)$ en Maxima:

En la pantalla principal de Maxima se ingresa el vector y se presiona Shift+Enter para que se evalúe la instrucción.

```
(%i1)   v:[1,0,2];
(%o1)   [1,0,2]
```

Adición

Sean $\vec{u}$ y $\vec{v}$, vamos a calcular $\vec{w} = \vec{u} + \vec{v}$:

$$\vec{u} = (2, 1, -2) \qquad \vec{v} = (1, 0, 2)$$

1. Introducir los vectores $\vec{u}$ y $\vec{v}$.

```
(%i1)   u:[2,1,-2]
(%o1)   [2,1,-2]

(%i2)   v:[1,0,2];
(%o2)   [1,0,2]
```

2. En la pantalla principal de Maxima escribir: `w:u+v` y presionar Shift+Enter.

```
(%i3)   w:u+v;
(%o3)   [3,1,0]
```

Producto de un vector por un escalar

Una vez que está ingresado el vector $\vec{u}$, para realizar el producto de un vector por un escalar, por ejemplo 2, basta con introducir en la pantalla principal de Maxima: **`2*u`** y presionar Shift+Enter.

```
(%i1)   u:[2,1,0];
(%o1)   [2,1,0]

(%i2)   2*u;
(%o2)   [4,2,0]
```

Si queremos expresar en forma simbólica, veamos el siguiente ejemplo:

```
(%i1)   v:[a,b,c];
(%o1)   [a,b,c]

(%i2)   v*n;
(%o2)   [a n,b n,c n]
```

Módulo

Para realizar el calculo de modulo de un vector, vamos a crear un programa. Maxima permite realizar programas que luego para ser utilizados, deben ser cargados bajo el comando `load()`. Observemos como crear el módulo de un vector, en Maxima ingresamos lo siguiente y guardamos el archivo con el nombre que se desea bajo la extensión `.mac`.

```
-->    modulo(v):=

       if listp(v)

       then sqrt(apply("+",v^2))$
```

Una vez que se haya guardado el archivo, en este caso lo llamamos `modulo`, cerramos el Maxima. Ahora bien, para aplicar esta función debemos cargar el archivo pero bajo la extensión `.mac`. Observemos un ejemplo:

```
(%i1)   load("C:/Documents and Settings/Administrador/Escritorio/modulo.mac");
        v:[2,-1,-2];
(%o1)   C:/Documents and Settings/Administrador/Escritorio/modulo.mac

(%i2)   modulo(v);
(%o2)   3
```

Aclaración: Para cargar el archivo `modulo`, accedemos a menú desplegable *Archivo* y elegimos la opción *Load Package…*, se abre una ventana en donde debemos buscar y seleccionar dicho archivo. Una vez que se encuentra el archivo, Maxima lo carga y ya puede utilizarse.

Producto escalar

El operador **.** realiza el producto escalar. Una vez que se ingresaron los vectores, para realizar el producto escalar, se introduce en Maxima: **v.u** y se presiona Shift+Enter.

```
(%i1)   v:[2,0,1];
(%o1)   [2,0,1]

(%i2)   u:[2,1,0];
(%o2)   [2,1,0]

(%i3)   v.u;
(%o3)   4
```

Otra forma de realizar el producto escalar de los vectores ***X*** e ***Y*** es mediante la función **innerproduct(*X*, *Y*)** que requiere el paquete **eigen** que se carga en memoria con la orden **load("eigen")**. Un sinónimo de **innerproduct** es **inprod**

```
(%i1)   load("eigen");
        innerproduct([5,3,2],[1,8,0]);

(%o1)   C:/ARCHIV~1/MAXIMA~1.1/share/maxima/5.19.1/share/matrix/eigen.mac
(%o2)   29
```

Producto vectorial

El paquete `vect` permite realizar el producto vectorial entre dos vectores. Veamos como se utiliza:

```
(%i1)   load(vect);
        a:[1,2,-1];
        b:[1,1,-1];

(%o1)   C:/ARCHIV~1/MAXIMA~1.1/share/maxima/5.19.1/share/vector/vect.mac
(%o2)   [1,2,-1]
(%o3)   [1,1,-1]

(%i4)   express(a~b);
(%o4)   [-1,0,-1]
```

Producto mixto

El producto mixto se realiza con el paquete `vect` y los operadores **.** y ~

```
(%i1)   load(vect);
        a:[1,2,-1];
        b:[1,1,-1];
        c:[1,-1,4];
(%o1)   C:/ARCHIV~1/MAXIMA~1.1/share/maxima/5.19.1/share/vector/vect.mac
(%o2)   [1,2,-1]
(%o3)   [1,1,-1]
(%o4)   [1,-1,4]

(%i6)   c.express(a~b);
(%o6)   -5
```

Gráfico de Vectores:

Para graficar vectores vamos a necesitar conocer el origen y el extremo. Por ejemplo, sean: $\vec{v}=(1,1,1)$ $\vec{u}=(1,-1,0)$ $\vec{w}=(1,1,-2)$

Estos vectores se grafican con la carga del paquete `draw` y la utilización del comando `draw3d`. Observemos cómo es la manera:

```
(%i1)   load(draw);
        draw3d(
            color = cyan,
            vector([1,4,-2],[1,3,1]),
            color=red,
            vector([-1,2,4],[1,-1,-3]),
            color=blue,
            vector([0,0,0],[3,1,-2]),
            xlabel="Eje x",
            ylabel="Eje y",
            zlabel="Eje z",
            xaxis=true,
            yaxis=true,
            zaxis=true,
            title="Tres vectores");
(%o1)   C:/ARCHIV~1/MAXIMA~1.1/share/maxima/5.19.1/share/draw/draw.lisp
(%o2)   [gr3d(vector,vector,vector)]
```

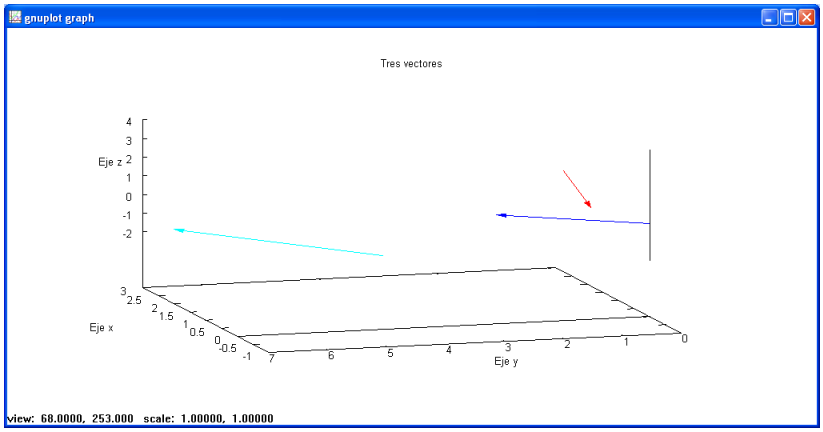


Figura 1

Forma genérica de graficar un vector en el **espacio**:

```
load(draw);
draw3d(
       color = color,
       vector (origen,extremo))$
```

Rectas en R^2

Existen dos maneras diferentes de graficar rectas en el plano en Maxima. Una es a través de la función plod2d, veamos cómo se utiliza:
En la parte superior de Maxima hacer un clic en el menú desplegable *Plot* -> *Plot 2d...*.

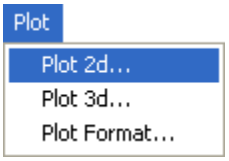


Figura 2

Donde se abrirá la siguiente ventana:

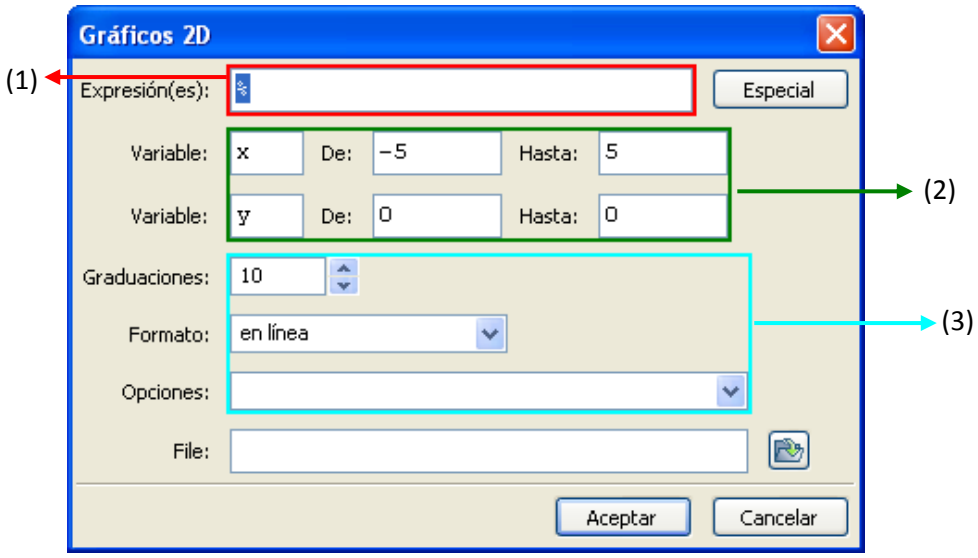


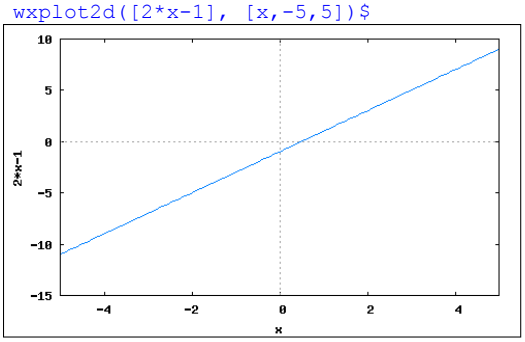
Figura 3

La **expresión** (1) debe ser introducida como una función de una variable. La sección (2) la vamos a llamar **rangos** que indica los valores que tomarán cada una de las variables. También existen distintas **opciones** (3) para el gráfico.

Veamos un ejemplo si se quiere realizar de la forma explícita:

Dada recta $r: y = 2x - 1$ entonces nuestra **expresión** será de la forma: `2*x-1`.
Vamos al menú *Plod* → *Plod 2d...* y aparece la ventana de la *Figura 3*, colocamos la expresión, los rangos y las distintas opciones y hacemos clic en *Aceptar*.
Maxima al graficar nos muestra:

```
(%i1)  wxplot2d([2*x-1], [x,-5,5])$
```



(%t1)

La otra manera de graficar rectas en el plano es cargando el paquete `draw` y usando la función `draw2d`:

Veamos un ejemplo:

Dadas las rectas $r: y = -\frac{2}{3}x - 1$ y $r: y = 4x + 3$ vamos a graficarlas.

Para realizar este gráfico solo basta con ingresar las siguientes líneas en la ventana principal de Maxima:

```
(%i1)   load(draw);
        draw2d(
                color = blue,
                explicit (-(2/3)*x-1,x,-5,5),
                color = red,
                explicit (4*x+3,x,-5,5))$
(%o1)   C:/ARCHIV~1/MAXIMA~1.1/share/maxima/5.19.1/share/draw/draw.lisp
```

Una vez que se ingresa lo anterior definido se presiona Shift+Enter y aparece lo solicitado:

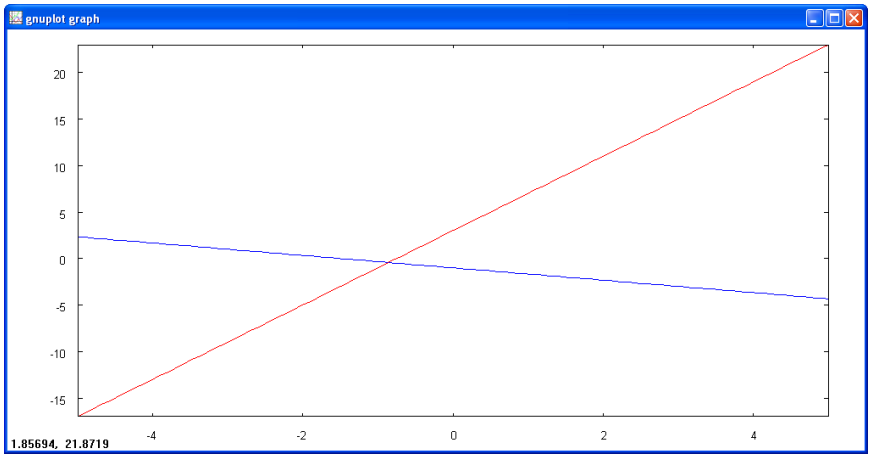


Figura4

Forma genérica de graficar una recta en el **plano**:

```
load(draw);
draw2d(
       color = color,
       explicit (expresion,variable,rangoMin,rangoMax))$
```

Plano

Una forma de graficar planos es a través de la función llamada **`plot3d`**.
Comencemos:
En la parte superior de la pantalla hacer clic en el menú desplegable *Plot* → *Plot 3d…*

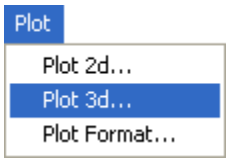


Figura 5

Aparecerá la siguiente ventana:

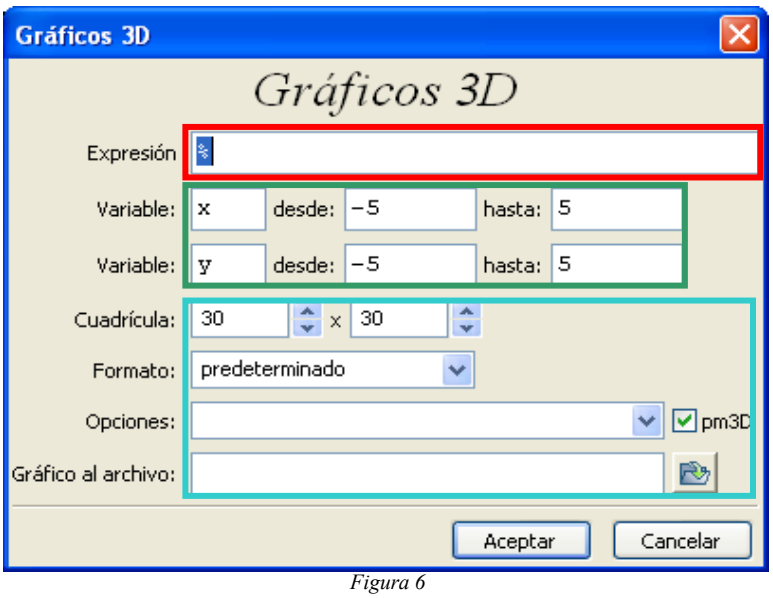


Figura 6

(1) La **expresión** (1) debe ser introducida como una función de dos variables.

(2) El **rango** (2) de cada una de las variables.

(3) También existen distintas **opciones** (3) para el gráfico que veremos más adelante.

Forma genérica de graficar planos:
Si la ecuación del plano es:

$\alpha : v_1x + v_2y + v_3z + d = 0 \Rightarrow$ Expresión: **`- (v1*x+v2*y+d) /v3`**

```
plot3d(expresion,[variable1,min,max],[variable2,min,max],opciones)$
                 └──────────────────────────────────────┘
                              rangos
```

Veamos un ejemplo:
Dada la ecuación del plano $\alpha : 4x - 2y - z + 13 = 0$ vamos a graficarlo.

Si $\alpha : 4x - 2y - z + 13 = 0 \Rightarrow$ `4*x-2*y+13=z` $\Rightarrow$ Expresión: **`4*x-2*y+13`**

La ventana deberá quedar de esta manera:

Figura 7

Una vez que Máxima finaliza de graficar, se abre en una ventana con el gráfico pedido:

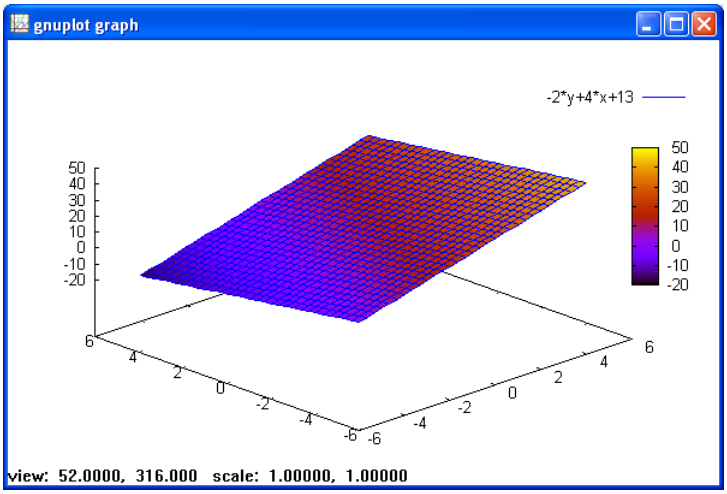


Figura 8

Observación: Es importante saber que haciendo clic en el gráfico, manteniendo apretado el mouse y moviéndolo de lado a lado se puede ***rotar*** el gráfico.

Otra forma de graficar planos es con la función **draw3d**. Recordemos que debemos utilizar el paquete **draw**, por lo tanto en la primera línea se ingresa el comando **load(draw);**. Esta función, se utiliza igual que para graficar las rectas en el espacio, en la ventana principal de Maxima, escribir los siguientes comandos:

```
(%i1)   load(draw);
        draw3d(
        color = blue,
        explicit (4*x-2*y+1,x,-5,5,y,-5,5),
        color = orange,
        explicit ((-3*x-2*y-5)/2,x,-5,5,y,-5,5))$
(%o1)   C:/ARCHIV~1/MAXIMA~1.1/share/maxima/5.19.1/share/draw/draw.lisp
```

Presionando Shift+Enter aparece el grafico pedido:

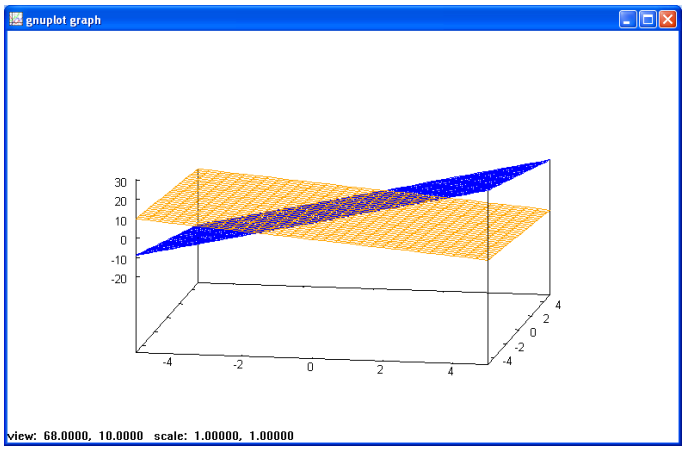


Figura 9

La ventaja de esta manera de graficar es que se pueden realizar más de un grafico.
Observemos una **forma genérica** de esta función:

```
load(draw);
draw3d(
       color = color,
       explicit (expresion,variable1,min,max,variable2,min,max))$
```

Rectas en R^3

Veamos una forma genérica de graficar una recta en el **espacio**:

$$r:\begin{cases} x = x_1 + v_1 * t \\ y = y_1 + v_2 * t \\ z = z_1 + v_3 * t \end{cases}$$

Cabe aclarar que para graficar rectas, se utilizan las ecuaciones cartesianas paramétricas de la recta. En este caso vamos a utilizar la función **`draw3d`**.

```
load(draw);
draw3d(
        color = color,
        line_width = numeroDeEspesorDeLaRecta,
        parametric (x1+v1*t,y1+v2*t,z1+v3*t, t,min,max))$
```

x **y** **z** **rango**

parámetro

Veamos cómo se grafica una recta:

Sea $r:\begin{cases} x = 1 + 2\lambda \\ y = 3 + \lambda \\ z = \lambda \end{cases}$ vamos a graficarla:

```
(%i1)   load(draw);
        draw3d(
                color = red,
                line_width = 3,
                parametric (1+2*t,3+t,t,t,-5,5))$
(%o1)   C:/ARCHIV~1/MAXIMA~1.1/share/maxima/5.19.1/share/draw/draw.lisp
```

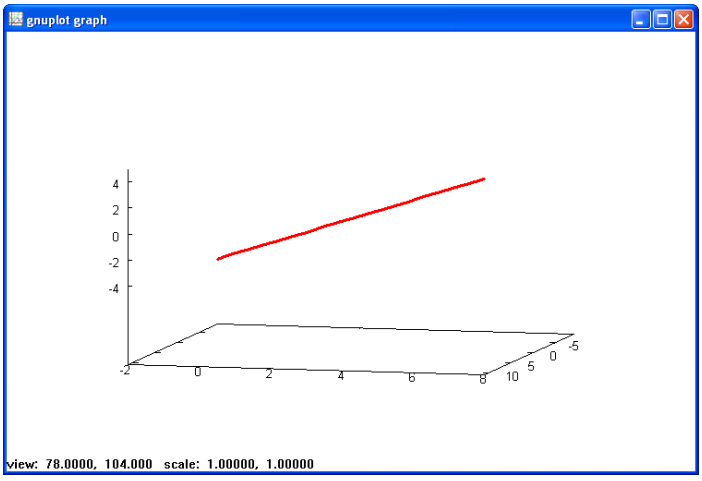


Figura 10

Como ya sabemos graficar una recta, pasamos a graficar dos:

Ejemplo 1

Sean $r_1: \begin{cases} x = 1+2\lambda \\ y = 2-\lambda \\ z = 3+\lambda \end{cases}$ y $r_2: \begin{cases} x = 4\mu \\ y = 1-2\mu \\ z = 2\mu \end{cases}$

```
(%i1)   load(draw);
        draw3d(
                color = blue,
                line_width = 2,
                parametric (1+2*t,2-t,3+t,t,-5, 5),
                color = red,
                line_width = 2,
                parametric (4*t,1-2*t,2*t,t,-5,5))$
(%o1)   C:/ARCHIV~1/MAXIMA~1.1/share/maxima/5.19.1/share/draw/draw.lisp
```

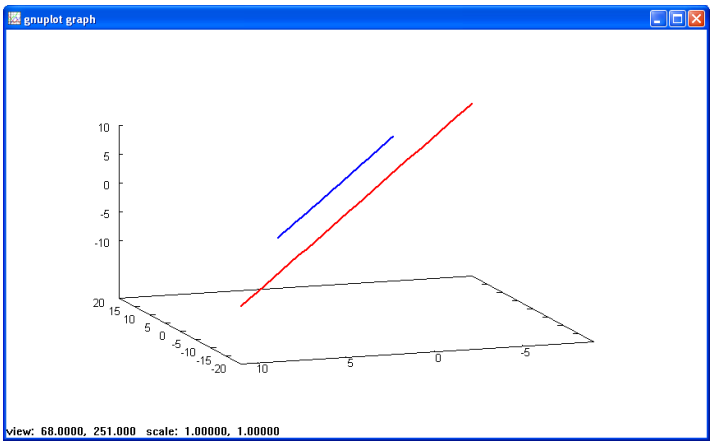


Figura 11

Observación: Las dos rectas graficadas son paralelas, ya que no se cortan en ningún punto y los vectores dirección son proporcionales.

Ejemplo 2

Dadas las rectas a continuación, las graficamos:

$$r_1: \begin{cases} x = 1-2\lambda \\ y = \lambda \\ z = 2 \end{cases} \qquad r_2: \begin{cases} x = 3-\mu \\ y = 2+2\mu \\ z = \mu \end{cases}$$

```
(%i1)   load(draw);
        draw3d(
                color = blue,
                line_width = 2,
                parametric (1-2*t,t,2,t,-5, 5),
                color = red,
                line_width = 2,
                parametric (3-t,2+2*t,t,t,-5,5))$
(%o1)   C:/ARCHIV~1/MAXIMA~1.1/share/maxima/5.19.1/share/draw/draw.lisp
```

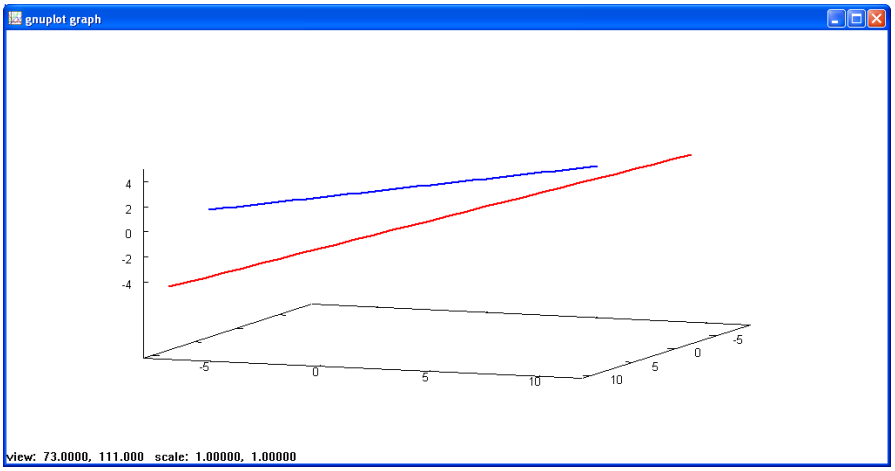


Figura 12

Rotando el grafico se obtienen distintas vistas, miremos la siguiente:

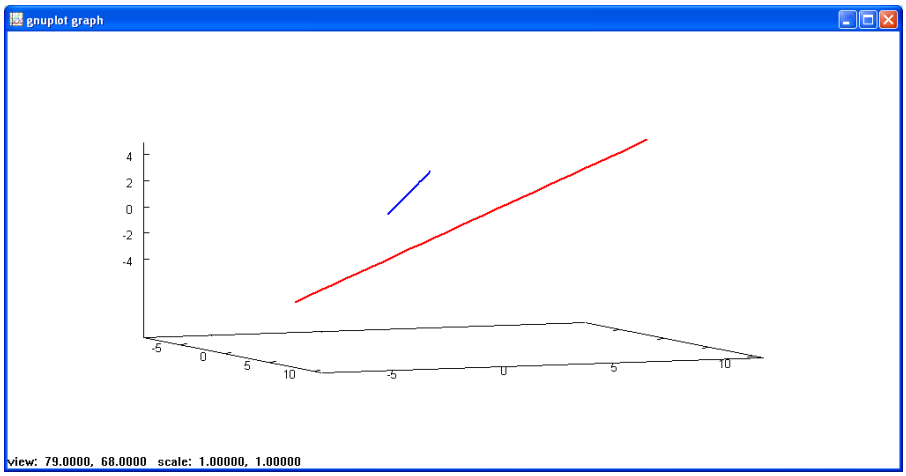


Figura 13

Observando los dos gráficos anteriores (*Figura 12 y 13*) podemos concluir en que las rectas no son paralelas y no se cortan, por lo tanto, r_1 y r_2 son alabeadas.

Matrices

Para operar con Matrices en Maxima vamos a utilizar el menú desplegable *Álgebra (Figura 14)* que se encuentra en la parte superior de la pantalla.

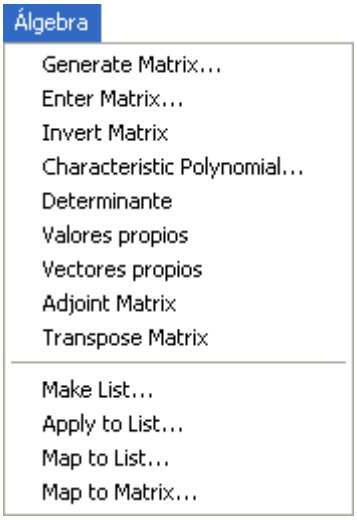


Figura 14

Comencemos primero con las distintas formas de construir la matriz M:

1. Mediante la opción *Enter Matriz…* del menú mencionado anteriormente.

$$M = \begin{pmatrix} 1 & -2 & 3 \\ 4 & 0 & 1 \\ 2 & 3 & 7 \end{pmatrix}$$

Una vez que se elige la opción *Enter Matriz…*, aparece la siguiente ventana:

Figura 15

En esta ventana (*Figura 15*) se pueden definir las diferentes características que va a tener la matriz. La cantidad de **filas** que va a tener la matriz, en este caso 3. La cantidad de **columnas** en este caso 3. El **tipo** en este caso es general pero existen otro tipos descriptos en la *Figura 16* .

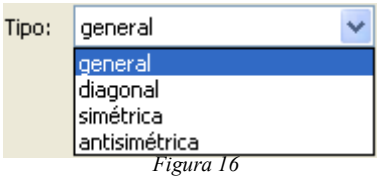


Figura 16

Una vez que se ingresan las dimensiones de la matriz, se colocan los datos en la siguiente ventana y se hace clic en *Aceptar*:

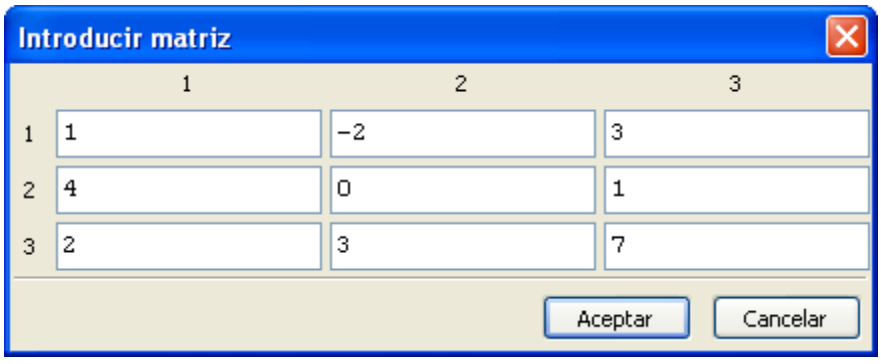


Figura 17

```
(%i1)   M: matrix(
          [1,-2,3],
          [4,0,1],
          [2,3,7]
        );
(%o1)   [ 1  -2  3 ]
        [ 4   0  1 ]
        [ 2   3  7 ]
```

2. Declarando sus elementos mediante listas, una para cada fila:

`matrix ([a11, a12, a13,...],[a21,a22,a23,...],...[an1,an2,...])`

```
(%i1)   matrix([1,2,%pi], [5,9,2]);
(%o1)   [ 1  2  %pi ]
        [ 5  9   2  ]
```

3. Introduciendo cada uno de los elementos a medida que Maxima los va pidiendo. En el caso de matrices cuadradas que sean diagonales, simétricas o antisimétricas no es necesario introducir todos los elementos.

`entermatrix(NúmeroFilas,NúmeroColumnas)`

```
    (%i1)     entermatrix(2,2);  /*Signifca que vamos a crear una matriz de 2x2 */
Is the matrix  1. Diagonal  2. Symmetric  3. Antisymmetric  4. General
Answer 1, 2, 3 or 4 : 4;   /*Nos pide seleccionar el tipo de matriz en este caso vamos a
Row 1 Column 1: 2;         hacer una general. */
Row 1 Column 2: 1;         /*Entonces ingresamos la opción 4 y presionamos Shift+Enter. */
Row 2 Column 1: 7;         /*A continuación ingresar cada elemento y Shift+Enter hasta que
Row 2 Column 2: 5;         se completa la matriz. */
Matrix entered.            /*Y Maxima genera la matriz. */
    (%o1)     [ 2  1 ]
              [ 7  5 ]
```

4. Definamos mediante una fórmula el elemento genérico de la matriz

`a[i,j]:=formula de i y j$ genmatrix (a,numeroFilas,numeroColumnas)`

```
(%i1)   a[i,j]:=i^2 + j$ genmatrix(a,3,2);
(%o1)   [ 2   3 ]
        [ 5   6 ]
        [ 10  11 ]
```

Las ***Matrices especiales***, como las diagonales, las nulas o la identidad, pueden construirse también con comandos específicos:

- `diagmatrix(numero,valor)`: Genera una matriz diagonal con cierto número de elementos todos ellos con el mismo valor.

```
(%i1)   diagmatrix(3,2);
(%o1)   [ 2  0  0 ]
        [ 0  2  0 ]
        [ 0  0  2 ]
```

- `zeromatrix(n,m);`: Genera la matriz nula de n filas y m columnas.

```
(%i1)   zeromatrix(2,3);
(%o1)   [0  0  0]
        [0  0  0]
```

- `ident(numero);`: Genera la matriz identidad cuyo tamaño viene dado por el número.

```
(%i1)   ident(4);
        [1  0  0  0]
        [0  1  0  0]
(%o1)   [0  0  1  0]
        [0  0  0  1]
```

Una vez que hemos entendido como se definen matrices en Maxima, se pueden realizar las operaciones básicas entre ellas:

Adición

Dadas A y B, vamos a calcular C=A+B:

$$A=\begin{pmatrix}1 & 2 & 3\\ -1 & x & 0\end{pmatrix} \qquad B=\begin{pmatrix}-2 & 1 & -2\\ 1 & 3 & x\end{pmatrix}$$

1. Introducir las matrices A y B.

```
(%i1)   A: matrix(
          [1,2,3],
          [-1,x,0]
        );
        [ 1   2  3 ]
(%o1)   [          ]
        [ -1  x  0 ]

(%i2)   B: matrix(
          [-2,1,-2],
          [1,3,x]
        );
        [ -2  1  -2 ]
(%o2)   [           ]
        [ 1   3  x  ]
```

2. En la pantalla principal de Maxima escribir: **`C:A+B`** y presionar Shift+Enter.

```
(%i3)   C:A+B;
        [ -1    3     1 ]
(%o3)   [               ]
        [ 0   x + 3   x ]
```

Producto de una matriz por un escalar

Una vez que esta ingresada la matriz M, para realizar el producto de una matriz por un escalar N, basta con introducir en la pantalla principal de Maxima: **N*M** y presionar Shift+Enter.

```
(%i1)  A: matrix(
         [3,2],
         [-1,4]
       );
(%o1)  [ 3   2 ]
       [ -1  4 ]

(%i2)  2*A;
(%o2)  [ 6   4 ]
       [ -2  8 ]

(%i3)  X*A;
(%o3)  [ 3X   2X ]
       [ -X   4X ]
```

Producto de matrices

1. Se introducen las matrices A y B.

   Es importante recordar que para poder realizar el producto AB, la cantidad de columnas de A debe ser igual a la cantidad de filas de B. La matriz producto tiene tantas filas como A y tantas columnas como B.
2. Se ingresa en la ventana principal de Maxima: **A.B** y luego Shift+Enter.

```
(%i1)  A: matrix(
         [1,-2,3],
         [4,0,1],
         [2,3,7]
       );
       [ 1  -2  3 ]
(%o1)  [ 4   0  1 ]
       [ 2   3  7 ]

(%i2)  B: matrix(
         [0,1,3,5],
         [1,4,1,2],
         [2,3,-1,0]
       );
       [ 0  1   3   5 ]
(%o2)  [ 1  4   1   2 ]
       [ 2  3  -1   0 ]

(%i3)  A.B;
       [ 4   2   -2   1  ]
(%o3)  [ 2   7   11   20 ]
       [ 17  35  2    16 ]
```

Potencia de una matriz

Con **^^** su puede elevar una matriz a una potencia.

```
(%i1)   A: matrix(
          [3,2],
          [-1,4]
        );
(%o1)   [ 3   2 ]
        [ -1  4 ]

(%i2)   A^^2;
(%o3)   [ 7   14 ]
        [ -7  14 ]
```

Cálculo de la Transpuesta de una Matriz

1. Se introduce la matriz A.
2. Se escribe en la pantalla principal el nombre de la matriz a trasponer en este caso A y se hace clic en el menú desplegable *Algebra -> Transpose Matriz* (*Figura 14*).

```
(%i1)   A: matrix(
          [1,2],
          [3,4],
          [5,6]
        );

(%o1)   [ 1  2 ]
        [ 3  4 ]
        [ 5  6 ]

 -->    A;
(%i2)   transpose(A);

(%o2)   [ 1  3  5 ]
        [ 2  4  6 ]
```

Cálculo de una matriz inversa

1. Se ingresa la matriz M.
2. Se escribe en la ventana principal el nombre de la matriz a invertir y se hacer clic en el menú desplegable *Algebra -> Invert matriz* (*Figura 14*).

```
(%i1)   M: matrix(
          [1,-2,3],
          [4,0,1],
          [2,3,7]
        );
(%o1)   [ 1  -2  3 ]
        [ 4  0   1 ]
        [ 2  3   7 ]

(%i2)   invert(%);

(%o2)   [ -3/85   23/85   -2/85  ]
        [ -26/85  1/85    11/85  ]
        [ 12/85   -7/85   8/85   ]
```

Veamos otro ejemplo para calcular la matriz inversa:

```
(%i1)   A:matrix([1,2,-1],[3,4,1],[2,2,2]);
        invert(A);
                                [ 1  2  -1 ]
(%o1)                           [ 3  4   1 ]
                                [ 2  2   2 ]
Division by 0
 -- an error.  To debug this try debugmode(true);
```

Observando la respuesta de Maxima podemos concluir que a la matriz ingresada no se le puede calcular su inversa.

Determinantes

Determinante de una matriz

1. Se ingresa la matriz A.
2. Se hacer clic en el menú desplegable *Algebra -> Determinante*.

```
(%i1)   A: matrix(
          [1,-2,3],
          [4,0,1],
          [2,3,7]
        );
        [ 1  -2  3 ]
        [          ]
(%o1)   [ 4   0  1 ]
        [          ]
        [ 2   3  7 ]
(%i2)   determinant(%);
(%o2)   85
```

Otra forma es mediante el comando **`determinant`** *`(nombreMatríz)`*

```
(%i1)   A:matrix([1,X,X^2],[1,Y,Y^2],[1,Z,Z^2]); determinant(A); factor(%);
        [ 1  X  X^2 ]
        [           ]
(%o1)   [ 1  Y  Y^2 ]
        [           ]
        [ 1  Z  Z^2 ]
(%o2)   -X(Z^2-Y^2) + Y Z^2 + X^2(Z-Y)-Y^2 Z
(%o3)   (Y-X) (Z-X) (Z-Y)
```

Observamos que se utilizo el comando **`factor(expresion);`** nunca antes visto. Para acceder al comando se debe seleccionar la expresión a factorizar y hacer un clic en el menú desplegable *Simplificar* y seleccionar la opción *Factor Expression* (*Figura 17*).

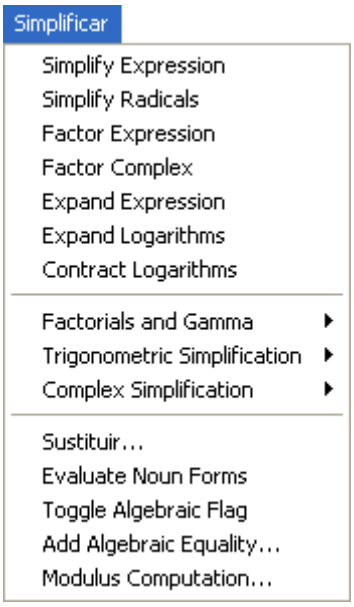


Figura 18

Calculo de la matriz adjunta

1. Se ingresa la matriz A.
2. Se escribe en la ventana principal el nombre de la matriz y se hacer clic en el menú desplegable *Algebra -> Adjoint Matrix*.

```
(%i1)   A: matrix(
          [1,-2,3],
          [4,0,1],
          [2,3,7]
        );
(%o1)   [ 1  -2  3 ]
        [ 4   0  1 ]
        [ 2   3  7 ]

(%i2)   adjoint(A);
(%o2)   [ -3   23   -2 ]
        [ -26  1    11 ]
        [ 12   -7   8  ]
```

Rango de una matriz

El comando **`rank`** *`(nombreMatríz)`* calcula el rango de la Matriz.

Ejemplo

```
(%i1)   A: matrix(
          [1,5,-8],
          [2,0,4],
          [3,-1,1],
          [2,0,1]
        );
(%o1)   [ 1   5   -8 ]
        [ 2   0   4  ]
        [ 3   -1  1  ]
        [ 2   0   1  ]

(%i2)   rank(A);
(%o2)   3
```

Veamos un ejemplo que nos devuelve el determinante y rango de la matriz A:

```
   (%i1)     A:matrix([1,2,3],[4,8,5],[3,6,2]);
             print("Determinante=")$ determinant(A);
             print("Rango=")$ rank(A);
   (%o1)     [ 1  2  3 ]
             [ 4  8  5 ]
             [ 3  6  2 ]
Determinate=
   (%o3)     0
Rango=
   (%o5)     2
```

Sistemas de Ecuaciones Lineales

Maxima tiene la capacidad de resolver cualquier tipo de sistemas de ecuaciones a través del menú desplegable *Ecuaciones (Figura 19)*.

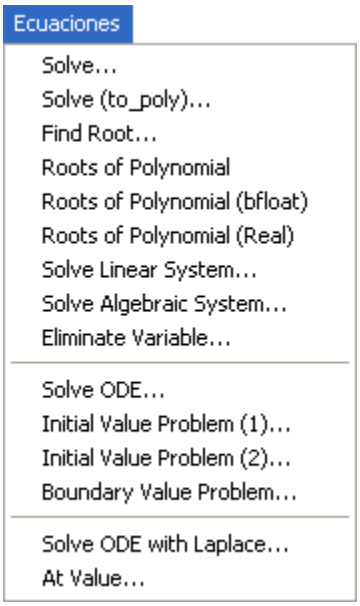


Figura 19

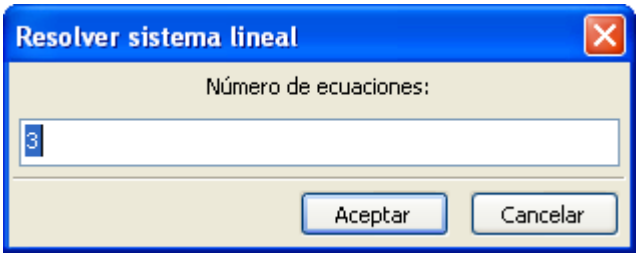


Figura 20

Para ingresar un sistema de ecuaciones lineales debemos hacer clic en el menú desplegable *Ecuaciones* y elegir la opción *Solve Linear System*....Luego de seleccionar dicha opción aparecerá una ventana donde hay que ingresar la cantidad de ecuaciones de nuestro sistema.

Finalmente se abre una nueva ventana en donde ingresamos las ecuaciones y las incógnitas.

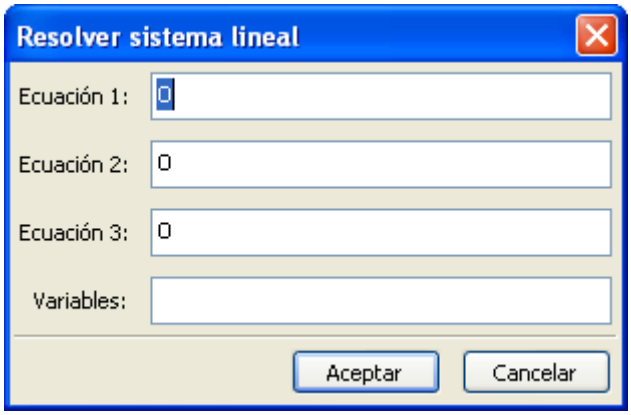


Figura 21

Una vez ingresados todos los datos Maxima calcula la solución.

Ejemplo

Sea el sistema $\begin{cases} x+2y-3z=9 \\ 2x-y+z=0 \\ 4x-y+z=4 \end{cases}$ calcular su solución.

```
(%i1)   linsolve([x+2*y-3*z=9, 2*x-y+z=0, 4*x-y+z=4], [x,y,z]);

                                /* Es un sistema compatible determinado ya que tiene una
(%o1)   [x=2,y=5,z=1]           única solución */
```

Como sabemos existen además estos sistemas:

1. Sistema Compatible Indeterminado,
2. Sistema Incompatible.

Vamos a ver cómo Maxima expresa los resultados para los distintos sistemas por medio de dos ejemplos.

Sean los siguientes sistemas de ecuaciones lineales, verificar si tienen solución. De ser así, calcularlas:

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

Los tres sistemas dados se ingresan y calculan de la misma manera explicada anteriormente o sino a través del comando `linsolve([ecuacion1], [ecuacion2], …, [ecuacionN], [ingognita1, incognita2, …, incognitaM])`

Para el sistema $a)$:

```
(%i1)   linsolve([x+2*y-z=3, 2*x+3*y+z=1], [x,y,z]);

                                        /* Es un sistema compatible indeterminado ya que
(%o1)   [x=-5*%r1-7,y=3*%r1+5,z=%r1]    tiene infinitas soluciones, donde %r1 significa
                                        cualquier real */
```

Para el sistema $b)$:

```
(%i1)   linsolve([x+2*y+z=3, 2*x+y+z=2, -x+y=3], [x,y,z]);

(%o1)   []      /* Es un sistema incompatible ya que no tiene una solución */
```

Ahora observemos cómo el siguiente ejercicio se puede resolver muy fácilmente en Maxima:
Resolver el sistema de ecuaciones lineales A.X=N, tal que

$$A = \begin{pmatrix} 1 & 2 & -2 \\ -2 & 1 & 4 \\ -1 & 3 & 2 \end{pmatrix}$$

Para realizar este ejercicio debemos realizar $A \cdot X = N$ siendo $X = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$ y $N = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$ resulta un sistema homogéneo donde A es la matriz de coeficientes.

Ahora en Maxima:

1. Ingresamos el sistema a través la opción del menú *Ecuaciones -> Solve Linear System…* en donde nos aparece la ventana *(Figura 20)* para ingresar el número de ecuaciones (En este caso ingresamos 3).
2. Luego debemos ingresar cada una de las ecuaciones como muestra la *Figura 21*.
3. Maxima calcula la solución:

```
    (%i1)      linsolve([x+2*y-2*z=0, -2*x+y+4*z=0, -x+3*y+2*z=0], [x,y,z]);
solve: dependent equations eliminated: (3)
    (%o1)      [x=2*%r2,y=0,z=%r2]
```