Actividad 9

Fernando Leyva Cárdenas

May 24, 2018

1 Introduccion

Maxima es un paquete de software libre. Se puede descargar libremente para varios sistemas diferentes y hay una amplia documentación que también se puede copiar libremente. Maxima es uno de los sistemas informáticos de álgebra más antiguos (CAS). Fue creado por el grupo MAC del MIT en la década de 1960 y se llamó inicialmente Macsyma (project Symbol MAmbolifier MAC). Macsyma fue desarrollado originalmente para las computadoras de gran escala DEC-PDP-10 que se utilizaron en diversas instituciones académicas en ese momento. En la década de 1980, su código fue portado a varias plataformas nuevas y una de esas versiones derivadas se llamó Maxima. En 1982, el MIT decidió vender Macsyma como software propietario y simultáneamente el profesor William Schelter de la Universidad de Texas continuó desarrollando la versión de Maxima. A fines de la década de 1980, aparecieron otros sistemas patentados de CAS similares a Macsyma, como Maple y Mathematica. En 1998, el profesor Schelter obtuvo la autorización del DOE (Departamento de Energía), que tenía los derechos de autor de la versión original de Macsyma, para distribuir el código fuente de Maxima como software libre.

Cuando el profesor Schelter falleció en 2001, se formó un grupo de voluntarios para continuar desarrollando y distribuyendo Maxima como software libre. En el caso del software CAS, las ventajas del software libre son muy importantes. Cuando un método falla o da respuestas muy complicadas, es bastante útil tener acceso a los detalles de la implementación subyacente de los métodos utilizados. Por otro lado, como la investigación y la enseñanza se vuelven dependientes de los resultados de un CAS, es deseable tener una buena documentación de los métodos implicados y su implementación y asegurarse de que no existen barreras legales que impidan el examen y la modificación de ese código.

Hay varias interfaces diferentes para trabajar con Maxima. Se puede ejecutar desde un shell de comandos o desde una de las interfaces gráficas como wxMaxima, imaxima o Xmaxima. Xmaxima establece una conexión con el programa Maxima (utilizando un socket), envía los comandos que el usuario escribe a Maxima y muestra los resultados que devuelve. Xmaxima generalmente abre dos ventanas. Uno de ellos, llamado navegador, muestra un tutorial y permite al usuario leer el manual u otras páginas web. La segunda ventana es la consola, donde los comandos de Maxima deben escribirse y su resultado aparecerá. En el menú "Editar" hay opciones para navegar por la lista de comandos anteriores ("entrada anterior") o para copiar y pegar texto; también se puede acceder a algunas opciones en los menús con las teclas de método abreviado que se muestran junto a ellas.

2 Entrada y salida de datos

Cuando se inicia una sesión de Maxima, aparecerá la etiqueta (% i1), que significa entrada 1. Se debe escribir un comando válido al lado de esa etiqueta, que termina con un punto y coma y cuando se pulsa la tecla Intro, esa entrada se analizará , simplificado, vinculado a una variable interna %i1 y su resultado se mostrará después de una etiqueta (% o1), que significa salida 1. Ese resultado también estará vinculado a una variable interna %o1. Aparecerá otra etiqueta (% i2) a continuación, para marcar el lugar donde debe escribirse un segundo comando y así sucesivamente; El uso más básico de Maxima es como una calculadora.

Por ejemplo:

```
(%i1) 2.5*3.1;
(%o1)         7.75
(%i2) 5.2*log(2);
(%o2)         5.2 log 2
```

El resultado (% o2) muestra dos aspectos importantes de Maxima. Primero, el logaritmo natural de 2 no se calculó, porque su resultado es un número irracional que no se puede representar exactamente con un número finito de dígitos numéricos. El segundo aspecto importante es que el símbolo * que siempre se requiere cuando se ingresa un producto y los paréntesis, que deben usarse para especificar el argumento de una función, no se incluyeron en el resultado. Eso sucedió porque, de forma predeterminada, la salida se muestra en un modo llamado display2d, en el que la salida intenta parecerse a la forma en que las expresiones matemáticas se muestran generalmente en los libros. La expresión "5.2 log 2" muy probablemente será interpretada correctamente por un lector, como el producto de 5.2 veces el logaritmo de 2; sin embargo, si esa misma expresión ambigua se le dio como entrada a Maxima, desencadenaría un error, porque la sintaxis de Maxima requiere un operador entre 5.2 y la función de logaritmo, y el argumento del logaritmo debe estar entre paréntesis. A pesar de la forma del resultado, la variable %o2 se ha vinculado a una expresión con sintaxis correcta, por lo que puede reutilizarse en comandos de Maxima posteriores sin errores de sintaxis.

3 Numeros

Maxima acepta números reales y complejos. Los números reales en Maxima pueden ser enteros, racionales, como 3/5, o números de coma flotante, por ejemplo, 2.56 y 25.6e-1, que es una notación corta para 25.6x10-1. Los números irracionales, como sqrt (2) (raíz cuadrada de 2) o log (2) (logaritmo natural de 2) se dejan en esa forma, sin ser aproximados por números de coma flotante, y cálculos posteriores, como sqrt (2) 2 o exp (log (2)) conducirá al resultado exacto 2.

Los números de coma flotante son "contagiosos"; es decir, las operaciones en las que ingresen se llevarán a cabo en ese formato. Por ejemplo, si en lugar de escribir log (2) uno escribiera log (2.0), el logaritmo se calcularía aproximadamente en coma flotante. Otra forma de forzar que una expresión se compute como un número de coma flotante consiste en usar la función flotante.

Una fuente frecuente de confusión surge del hecho de que esos números están siendo representados internamente en base binaria y no en base decimal; por lo tanto, ciertos números que pueden representarse en decimal con algunos dígitos, por ejemplo 0.1, necesitarían un número infinito de dígitos binarios para ser representados con precisión en base binaria. Es lo mismo que ocurre con la fracción 1/3 en base decimal, que en forma de coma flotante tiene un número infinito de dígitos: 0,333 ... (en el sistema base 3, esa fracción se puede representar fácilmente). Las fracciones que conducen a un número infinito de dígitos no son las mismas en los sistemas de base decimal y binario. Considere los siguientes resultados, que son perfectamente correctos y aparecerían en cualquier sistema que use dígitos binarios y formato de precisión doble, pero podrían parecer desconcertantes para alguien que solía trabajar con el sistema decimal:

```
(%i5) 2*0.1;
(%o5)         0.2
(%i6) 6*0.1;
(%o6)         0.60000000000000001
```

La explicación de este último resultado es que el número 0.1 no se puede escribir exactamente utilizando 64 bits binarios. Por lo tanto, multiplicar 0.1 por 2 no da exactamente 0.2, pero el número decimal con 16 dígitos significativos que está más cerca del resultado obtenido es 0.2000000000000000, dando la impresión de que el resultado del producto es exactamente eso, cuando no lo es. En el caso de 6 * 0.1, con formato de doble precisión, el número más cercano con 16 dígitos significativos decimales es 0.6000000000000001. Algunos sistemas informáticos ignoran los últimos dígitos en los resultados obtenidos de los cálculos de

precisión doble, mostrando el resultado como 0.6, pero siempre que se usa la precisión doble binaria, el resultado de 6 * 0.1 nunca será exactamente 0.6.

Si el número 1/3 debe representarse en sistema decimal, usando solo 3 dígitos significativos, la representación más aproximada del número sería 333/103, es decir, 0.333. En el sistema binario con doble precisión 52 se usan dígitos binarios significativos, lo que significa que el numerador debe ser menor que 2^{52} y el denominador debe ser un número de la forma 2^n. La función racionalizar de Maxima muestra la representación aproximada que se usa para un número, en forma de una fracción. Por ejemplo:

```
(%i7) rationalize (0.1);
3602879701896397/36028797018963968
(%o7)
```

el numerador es menor que 2^{52} (y mayor que 2^{51}), mientras que el denominador es exactamente igual a 2^{55}. Para que esa fracción represente exactamente 0.1, el denominador debe ser diez veces más grande que el numerador, es decir, debe terminar en 70 en lugar de 68, pero el poder de 2 más cerca de ese número tuvo que ser utilizado.

Para evitar los errores numéricos inherentes a la representación de coma flotante, se pueden usar fracciones; por ejemplo, 1/10 en lugar de 0.1. También hay otro formato específico de Maxima que acepta cualquier número arbitrario de dígitos significativos para representar números de coma flotante. Ese formato se llama big float y se usa escribiendo "b", en lugar de "e" para los exponentes; por ejemplo, 2.56×1020, escrito como 2.56e20 estaría representado internamente en formato de precisión doble, con 16 dígitos significativos y cualquier cálculo realizado con él daría como resultado otros números de precisión doble; pero si el mismo número se escribe como 2.56b20, se representará internamente en formato de Big-Float y cuando haga parte de los cálculos numéricos, el resultado será otro número en el mismo formato, que puede tener muchos dígitos significativos hasta un número máximo determinado por el valor de la variable interna fpprec (precisión de coma flotante).

4 Variables

Para vincular un valor u otros objetos a una variable, se usa el símbolo ":" y no el signo igual "=", que se usará para definir ecuaciones matemáticas. El nombre de las variables puede ser cualquier combinación de letras, números y uno de los símbolos % o _ , pero el primer carácter no puede ser un número. Maxima es sensible a mayúsculas y minúsculas.

Por ejemplo:

```
(%i11) a: 2$
(%i12) [b, c]: [-2, -4];
(%o12)          [-2,4]
(%i13) c;
(%o13)         -4
(%i14) Root1: (-b + sqrt(b^2 - 4*a*c))/(2*a);
(%o14)         2
(%i15) d: sqrt(z^2 + a*c);
```

las variables a, b, c y Root1 estaban relacionadas con los valores numéricos 2, -2, -4 y 2, mientras que la variable d estaba vinculada a una expresión.

Observe que la entrada (% i11) finalizó con un signo de dólar $, en lugar de un punto y coma. Eso hará que el comando se ejecute sin mostrar su resultado en la pantalla. En cualquier caso, la variable %o11 se vinculó con el resultado de la entrada (% i11) y se puede consultar más adelante, aunque su valor no se haya mostrado. La entrada (% i12) muestra cómo vincular varias variables con un solo comando. Cuando se escribe el nombre de una variable, como en la entrada (% i13), el resultado será el valor vinculado a esa variable o el nombre de la misma si no se ha vinculado a ningún valor. En la expresión dada a vincularse con

Root1, las variables a, byc se reemplazaron por los valores vinculados a ellas, y el resultado se simplificó y se vinculó a la variable, mientras que la variable d se relacionó con una expresión que depende de z, porque esa variable no estaba vinculada a ningún valor.

Para eliminar el valor vinculado a una variable, se puede usar la función remvalue; en el siguiente ejemplo, el valor vinculado a a se elimina y una expresión que depende de a se vincula a Root1:

```
(%i16) remvalue (a)$
(%i17) Root1: (-b + sqrt(b^2 - 4*a*c))/(2*a);
```

Para eliminar todos los valores vinculados a variables, se usa el comando remvalue (all). Tenga en cuenta que una variable se puede vincular a un valor numérico, a una expresión algebraica o a cualquier otro objeto de Maxima. Para sustituir una variable en una expresión por un valor dado, se usa el comando subst; Tenga en cuenta que cuando varias variables se reemplazan por valores, las variables y los valores deben colocarse entre corchetes y separados por comas.

5 Listas

Una variable también se puede vincular a una lista de valores, que se colocan entre corchetes y separados por comas. Por ejemplo, el siguiente comando vincula cuadrados de variables a una lista con los cuadrados de los primeros 5 números enteros positivos:

```
 (%i25) squares: [1, 4, 9, 16, 25]$
```

Los elementos de una lista están numerados por números enteros que comienzan por 1. Para hacer referencia a un elemento de la lista, el índice correspondiente se escribe entre corchetes; por ejemplo, el tercer elemento de la lista vinculado a cuadrados es 9, que se puede extraer de esta manera:

```
(%i27) squares[3];
(%o27)         9
```

Una función muy útil para crear listas es makelist, que expande una expresión, reemplazando varios valores diferentes para una variable determinada. El primer argumento dado a makelist debe ser la expresión a expandir y el segundo argumento es el nombre de la variable que será reemplazada por una secuencia de valores desde un valor inicial hasta un valor máximo definido por el tercer y cuarto argumentos. Si se proporciona un quinto argumento, se usará como el incremento en la secuencia de valores que se reemplazarán; de lo contrario, se usará el incremento predeterminado de 1.

6 Constantes

Hay algunas constantes matemáticas predefinidas en Maxima. Los nombres de variable vinculados a esas constantes generalmente comienzan con el símbolo %. Tres constantes importantes son el número π , vinculado a % pi, el número de Euler, la base de los logaritmos naturales, vinculado a % y el número imaginario, vinculado a % i. Tanto %pi como %e son números irracionales que no se pueden representar exactamente con un número finito de dígitos, pero se puede obtener una aproximación de coma flotante, con 16 dígitos significativos, utilizando la función float; también se puede encontrar una representación numérica con dígitos más significativos utilizando la función bfloat y la variable fpprec.

7 Archivos de comando

Para guardar todos los comandos que se han ingresado durante una sesión de trabajo en Xmaxima, hay una opción "Guardar entrada máxima en el archivo" en el menú "Archivo". El archivo creado con esa opción se puede cargar más adelante en Maxima, haciendo que todos los comandos en el archivo se ejecuten como

si se hubieran ingresado secuencialmente, usando la opción "Archivo por lotes" en el menú "Archivo". Las funciones y el lote de las funciones de Maxima también se pueden usar para realizar las mismas tareas, sin utilizar las opciones de menú de Xmaxima.

El archivo creado contiene texto simple simple, que se puede editar con un editor de texto. Los comandos ingresados aparecerán sin las etiquetas (% i1), (% i2), etc. por lo tanto, se debe tener cuidado con los comandos que hacen referencia a los resultados previos % o1,% o2, etc., ya que la secuencia de números asignados a esas salidas puede ser diferente. Los comentarios se pueden incluir en ese archivo, comenzándolos con los símbolos / * y terminando con * /, que pueden aparecer varias líneas debajo del inicio del comentario. Los comandos ingresados directamente en Maxima o escritos en ese archivo también pueden contener espacios en blanco entre números, operadores, variables y otros objetos, para hacerlos más legibles, y cada comando también puede expandir varias líneas.

Una forma eficiente de trabajar con Maxima consiste en comenzar escribiendo un archivo de texto, llamado archivo "por lotes", con los comandos que se van a utilizar, que luego se cargarán con la opción "Archivo por lotes". De esta forma, si aparece un error que hace necesario volver a ingresar los mismos comandos, será suficiente corregir el comando incorrecto en el archivo y cargarlo nuevamente. Los comandos en ese archivo se deben escribir sin etiquetas (% i1), (% i2), ... que se asignarán automáticamente cuando se ejecute el archivo. La opción de Xmaxima "Guardar Consola a Archivo", en el menú "Editar", guarda toda la información que se muestra en la pantalla, incluidas las etiquetas (% i1), (% o1), (% i2), (% o2), etc. Ese archivo puede ser útil para la documentación, pero no puede reutilizarse como archivo por lotes.

Algunos comandos que se usan repetidamente en diferentes sesiones de trabajo, por ejemplo, la definición de una función de uso frecuente, se pueden colocar dentro de un archivo que luego se cargaría utilizando lotes ("archivo"), donde "archivo" es el nombre completo y la ruta del archivo utilizado. Si el nombre del archivo no incluye una ruta al directorio donde está ubicado, primero se buscará en el directorio actual y luego en un directorio donde Maxima espera encontrar los archivos por lotes del usuario. La ubicación predeterminada de ese directorio se puede ver examinando los contenidos de la variable del sistema maxima_userdir.

Un archivo por lotes también se puede cargar automáticamente cada vez que se inicia Maxima, si recibe el nombre maxima-init.mac y se coloca en el directorio donde Maxima espera encontrar los archivos por lotes del usuario.

8 Algebra

Las expresiones pueden incluir operaciones matemáticas con variables abstractas. Por ejemplo:

```
 (%i33) 3*x^2 + 2*cos(t)$
```

para encontrar las raíces de una función polinómica, se pueden usar allroots; por ejemplo:

```
(%i36) allroots(%);
(%o36) [x=0.9073i+0.2776, x=0.2776-0.9073i, x=-2.222]
```

Hay dos raíces complejas y una verdadera. Las tres raíces se colocaron dentro de una lista. Para extraer, por ejemplo, el lado derecho de la tercera raíz de la lista, se usa el comando rhs (abreviatura de la derecha):

```
(%i37) rhs(%[3]);
(%o37)         -2.222
```

La variable x permanece indefinida, ya que el signo igual no vincula la variable con el valor del otro lado. Los resultados dados en (% o36) son aproximaciones numéricas y no las raíces exactas. En algunos casos, las expresiones algebraicas exactas para las raíces se pueden encontrar utilizando el comando resolver, que también puede resolver otros tipos de ecuaciones, no solo polinomios. Por ejemplo, las raíces que se encuentran arriba también podrían haberse obtenido con los siguientes comandos:

```
 (%i38) solve ( 3*x^3 + 5*x^2 = x - 6, x )$
(%i39) float ( rectform (%));
(%o39) [x=0.9073i+0.2776, x=-2.222, x=0.2776-0.9073i]
```

El resultado exacto dado por la función resolver toma varias líneas y no se mostró en la pantalla; solo la aproximación de esas raíces como números de coma flotante se mostró en este caso.

Recuerde que cuando un nombre de variable ya ha sido vinculado a un valor, será necesario escribir una comilla simple antes del nombre de la variable, para poder usarla como una variable algebraica abstracta. También se puede eliminar el valor vinculado a esa variable utilizando la función remvalue.

Para resolver un sistema de ecuaciones, que puede ser lineal o no lineal, el primer argumento dado para resolver debe ser una lista con las ecuaciones y el segundo argumento debe ser otra lista con los nombres de las variables; la lista de ecuaciones o cada ecuación que contiene puede estar vinculada previamente a alguna variable, por ejemplo:

```
(%i40) eqA: (4 + 8)*x1 - 8* x2 = 6 + 4$
(%i41) eqB: (2 + 8 + 5 + 1)*x2 - 8*x1 = -4$
(%i42) solve ( [eqA, eqB], [x1, x2] );
```

El resultado fue una lista dentro de otra lista, porque la primera lista incluye los valores de las variables y la segunda lista incluye las diversas soluciones posibles para el sistema, que en este caso era solo una. El sistema anterior también podría haberse resuelto con el comando linsolve, en lugar de resolver, porque las ecuaciones son lineales; Maxima incluye muchas otras funciones para trabajar con expresiones algebraicas.

El factor de función se usa para factorizar expresiones. Otras funciones utilizadas para simplificar expresiones son ratsimp, radcan y xthru. Entre varias expresiones equivalentes, el concepto de simplicidad es relativo y es más una cuestión de gusto; por lo tanto, diferentes funciones de simplificación pueden dar expresiones diferentes, aunque deberían ser equivalentes. Es conveniente probar varias funciones simplificadoras en cada caso y luego elegir una forma preferida de una expresión. Además, en algunos casos, como sucede con ratsimp, los resultados pueden ser diferentes cuando la misma función se aplica nuevamente.

Las expresiones algebraicas se representan internamente como listas; por lo tanto, algunas funciones de Maxima para listas también se pueden usar con expresiones. Por ejemplo, la longitud de la función da la longitud de una lista y también se puede usar para calcular el número de términos en una expresión; por ejemplo:

```
(%i46) length(res);
(%o46)         2
```

Una expresión que no puede separarse más en otras subexpresiones, por ejemplo, x, se llama átomo; las funciones que esperan una lista como su argumento generalmente desencadenarán un error cuando se les dé un átomo como argumento. Función átomo dice si el argumento dado es un átomo o no.

Otra función que es muy útil para tratar listas es el mapa, que aplicará una función determinada a cada elemento de una lista. En el caso de una expresión racional, se puede usar para aplicar una función al numerador y denominador de la expresión.

9 Trigonometría

La Tabla A.1 muestra los nombres de las principales funciones trigonométricas en Maxima. Las funciones que esperan un ángulo como su argumento de entrada interpretan ese ángulo en radianes y no en grados, ya que Maxima también conoce algunas propiedades de esas funciones, incluidas sus series de potencias, que solo son válidas cuando el ángulo se expresa en radianes. Los resultados dados por las funciones inversas son ángulos en radianes.

```
Table A.1: Trigonometric functions Function         Description
sin(x)                                               Sin
cos(x)                                             Cosine
tan(x)                                             Tangent
sec(x)                                              Secant
csc(x)                                             Cosecant
```

```
cot(x)                                    Cotangent
asin(x)                                   Arc sine
acos(x)                                 Arc cosine
atan(x)                                 Arc tangent
atan2(y,x)                               Arc tangent
asec(x)                                 Arc secant
acsc(x)                               Arc cosecant
acot(x)                             Inverse cotangent
```

las cuales se usan para diversos problemas que se pueden resolver con maxima.

10 Calculo

La forma más sencilla de representar funciones matemáticas en Maxima es mediante el uso de expresiones. Por ejemplo, para representar la función f(x)=$3x^2$-5x, la expresión en el lado derecho está vinculada a la variable f:

```
(%i60) f: 3*x^2 - 5*x;
(%o60)
```

$3x^2$-5x

La derivada de f con respecto a x se calcula usando la función diff:

```
(%i61) diff (f, x);
(%o61)
```

6x-5

y la antiderivada con respecto a x se obtiene con integración:

```
(%i62) integrate (f, x);
(%o62)
```

x^3-(5/2)x^2

El valor de la función en un punto, por ejemplo f (1), puede obtenerse sustituyendo x por 1 usando subst, o con función en:

```
(%i63) at (f, x=1);
(%o63)         -2
```

Lo mismo puede hacerse para integrales definidas, es decir, con intervalos en los extremos de la integral.

11 Funciones

Una función de Maxima es un programa con algunas variables de entrada y una salida. Maxima tiene un lenguaje de programación simple que se utiliza para definir esas funciones y también es posible usar Lisp, que es el idioma en el que se escribe Maxima, para definir funciones. Incluso es posible redefinir cualquiera de las funciones que se han referido; por ejemplo, si en la versión de Maxima se está utilizando alguna función que tiene un error que ya se ha corregido en una versión más reciente, es posible cargar la nueva versión de la función y, a menos que introduzca conflictos con otras funciones anteriores, debería trabajar correctamente.

Varios comandos de Maxima se pueden agrupar tipeando dentro de paréntesis y separados por comas. Esos comandos se ejecutan secuencialmente y el resultado del último comando será el resultado de todo el grupo. Cada comando puede sangrarse y puede expandir más de una línea. El siguiente ejemplo define una función que agrega todos los argumentos que se le otorgan:

```
(%i74) add([v]) := block([s: 0],
       for i:1 thru length(v) do
              (s : s + v[i]),
       s)$
(%i75) add (45,2^3);
(%o75)         53
(%i76) add (3,log(x),5+x);
(%o76)          Logx+x+8
```

Se utilizó una lista como argumento para la función, lo que hace que la función acepte cualquier cantidad de variables de entrada (o ninguna) y todas las variables de entrada se colocarán en una lista vinculada a la variable local v. El bloque de funciones se usó para definir otra variable local s, con un valor inicial de 0, que al final de la función tendrá la suma de las variables de entrada. El primer elemento dado al bloque debe ser una lista, con cualquier cantidad de variables locales, cada una con o sin un valor inicial y después de esa lista sigue la parte restante de la definición de la función. El comando itera la variable local i -en este caso desde 1 hasta la longitud de la lista v- con incrementos, por defecto, igual a 1 (se puede dar un paso opcional para modificar el valor predeterminado de ese incremento). Cuando se realizan las iteraciones, se muestra el valor de la variable para que se convierta en la salida de la función.

Cuando se usa una función desconocida, no se desencadenan errores; en cambio, la función desconocida se repite en la salida; por ejemplo:

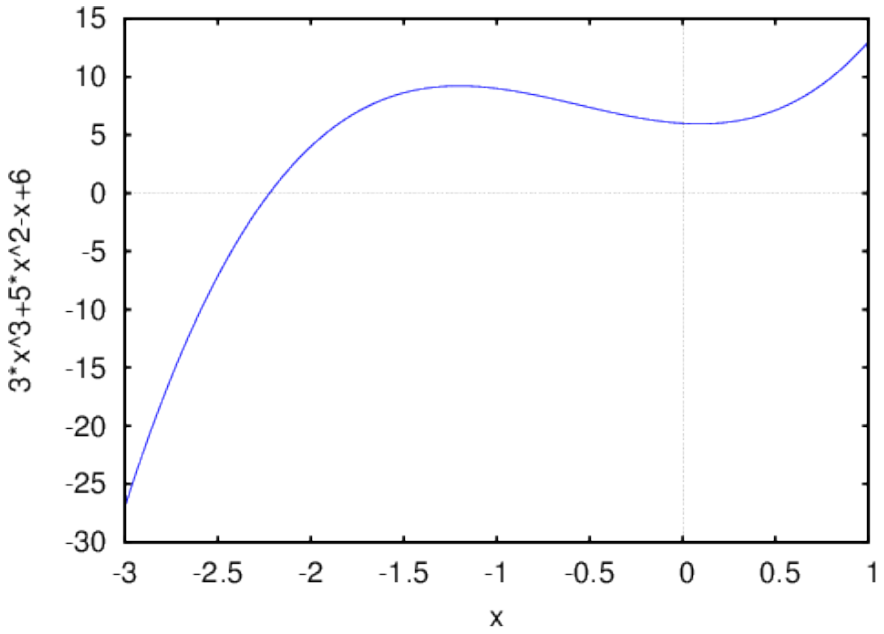
```
(%i77) 2*4*maximum(3,5,2);
(%o77)         8 Maximum(3,5,2)
```

12 Graficas

1. **Funciones de una variable**
   plot2d se usa para mostrar la representación de una o varias funciones de una variable. Por ejemplo la grafica del polinomio $3x^3+5x^2-x+6$ para valores de **x** entre -3 y 1 se muestra con el siguiente comando:

   ```
   (%i80) plot2d(3*x^3 + 5*x^2 - x + 6, [x, -3, 1]);
   ```

   el resultado del comando (% o80) es el nombre de un archivo auxiliar que se creó y luego se pasó a un programa externo (Gnuplot) que interpretará los comandos en él y mostrará el diagrama en un archivo separado ventana; a saber:



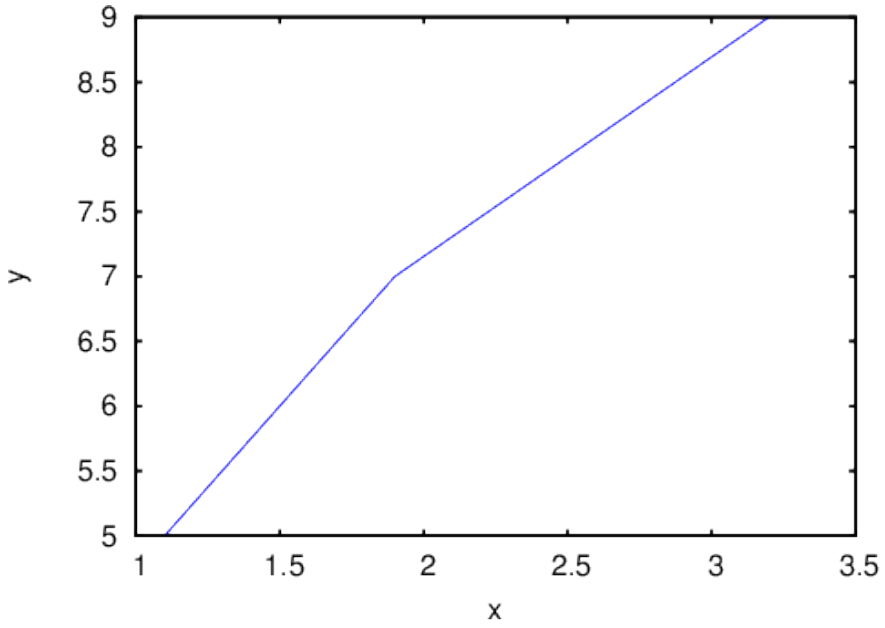
Para trazar varias funciones en la misma ventana, esas funciones se colocan dentro de una lista. Por ejemplo:

```
(%i81) plot2d ( [sin(x), cos(x)], [x, -2*%pi, 2*%pi] );
```

2. **graficas con puntos**
   Ahora para cuando tenemos puntos de la grafica, ¿como la podemos obtener?; También es posible crear gráficos con listas de puntos en un sistema de dos coordenadas. Las dos coordenadas de cada punto se pueden dar como una lista, dentro de otra lista con todos los puntos. Por ejemplo, para mostrar los tres puntos (1.1, 5), (1.9, 7) y (3.2.9) en una gráfica, las coordenadas de los puntos se pueden colocar dentro de una lista vinculada a p:

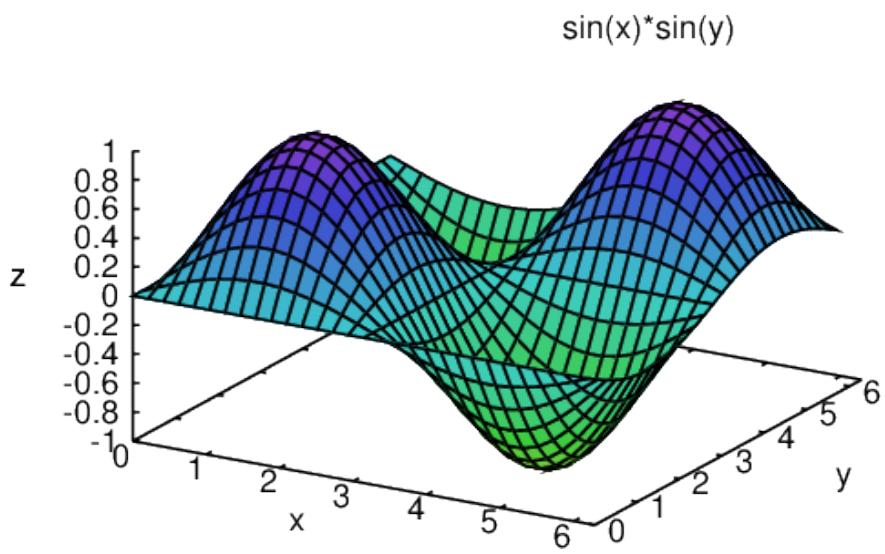
```
(%i84) p: [[1.1, 5], [1.9, 7], [3.2, 9]]$
```

Por defecto, los puntos están vinculados por segmentos de línea; para mostrar solo los puntos, sin segmentos de línea, el estilo de opción se debe usar con un valor igual a los puntos de palabra clave.



3. **Funciones de dos variables**
   el Comando plot3d se usa para trazar funciones de dos variables. Por ejemplo, el siguiente comando crea el diagrama que se muestra en la Figura:

```
(%i88) plot3d ( sin(x)*sin(y), [x, 0, 2*%pi], [y, 0, 2*%pi] );
```



13 Conclusion

Esto es lo mas fundamental al usar maxima, podemos ver y notar que tiene muchas herramientas utiles para el estudio de la fisica y matematicas que nos puede servir para estudiar entre otras cosas, por lo que es una gran herramienta estudiantil muy recomendable ademas de ser software libre.

14 Apendice

1. **¿Cuál fue tu primera impresión de wxmaxima?** que esta muy padre
2. **¿Crees que esta herramienta puede ser útil en otros de tus cursos?** la verdad si
3. **¿Qué se te dificultó mas en esta actividad?** casi nada, solo el formato del codigo
4. **¿Se te hizo compleja esta actividad? ¿Cómo la mejorarías?** Lo unico complejo que senti fue el ingles, caeria muy bien bibliografia en español :')

15 Bibliografia

- maxima tutorial, URL: $https://def.fe.up.pt/dynamics/maxima_tutorial.html$, 24/05/18
- tutorial de wxmaxima, URL: $http://www.scotchildress.com/wxmaxima/$, 24/05/18
- Maxima by example, URL: $http://web.csulb.edu/\ woollett/$, 24/05/18