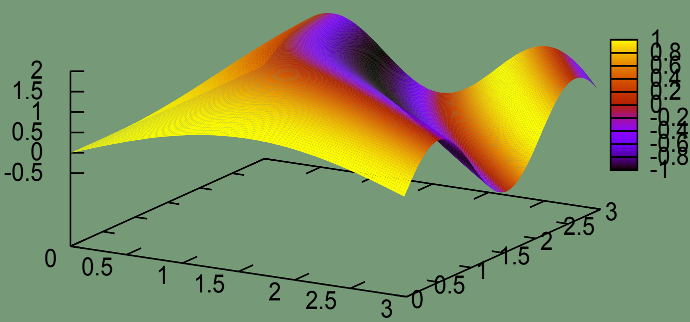
BREVE MANUAL DE

MAXIMA

Segunda Edición



R. Ipanaqué

publicaciones eumed.net

Breve Manual de *Maxima*

Segunda Edición

Breve Manual de *Maxima*

Segunda Edición

R. Ipanaqué

Departamento de Matemática
Universidad Nacional de Piura

publicaciones eumed.net

Robert Ipanaqué Chero

Departamento de Matemática
Universidad Nacional de Piura
Urb. Miraflores s/n, Castilla, Piura
PERÚ

https://sites.google.com/site/ripanaque
robertchero@hotmail.com

La composición de BREVE MANUAL DE MAXIMA, Segunda Edición,
*se ha hecho en LaTeX, usando el editor libre TeX**MAKER** 3.2.2.*

Primera Edición: Mayo 2010,
Segunda Edición: Enero 2012.
Publicado por el grupo **eumed•net**.
Grupo de Investigación de la Universidad de Málaga, España
http://www.eumed.net

Hecho el depósito legal en la Biblioteca Nacional de España
con Registro N° 10/101865

ISBN-13: 978-84-693-7160-2

En memoria de mi padre,
Juan A. Ipanaqué Vargas

Índice general

Prólogo

Este manual da una introducción al Software Libre *Maxima v5.25.1*, presentándolo como un potente Sistema de Álgebra Computacional (Computer Algebra System, o CAS) cuyo objeto es la realización de cálculos matemáticos, tanto simbólicos como numéricos; además de ser expandible, pues posee un lenguaje de programación propio.

Las razones para apostar por el uso de Software Libre pueden deducirse de las cuatro libertades asociadas a este tipo de Software: *libertad de ejecutarlo, para cualquier propósito; libertad de estudiar cómo trabaja, y cambiarlo a voluntad de quien lo usa; libertad de redistribuir copias para ayudar al prójimo; y libertad de mejorarlo y publicar sus mejoras, y versiones modificadas en general, para que se beneficie toda la comunidad.*

De hecho, las libertades asociadas a todo Software Libre y, en particular, al CAS *Maxima* hacen de éste una formidable herramienta pedagógica accesible a todos los presupuestos, tanto institucionales como individuales. No obstante, somos sinceros en señalar que no posee toda la versatilidad de sus símiles comerciales; pero el hecho que sea gratuito minimiza tal carencia. Hay que señalar, también, que cualquier actualización de un Software Libre puede obtenerse sin obstáculo alguno y así es posible contar inmediatamente con la última versión del mismo. Algo que no sucede con el Software Comercial, a menos que se tenga disponibilidad inmediata de dinero para pagar la actualización.

La idea de elaborar este manual surge de la necesidad de contar con bibliografía propia acerca de un CAS Libre para trabajar con alumnos de un curso de pregrado, los cuales ya estaban familiarizados con el uso de un CAS Comercial. La experiencia ha sido bastante satisfactoria y quedan en el tintero los borradores para la futura elaboración de un libro en el que se plasmen los resultados obtenidos en tal curso.

Este manual se compone de diecinueve capítulos en los cuales se describen resumidamente las principales características de las funciones incorporadas en el núcleo de *Maxima*, así como de algunos paquetes que son de gran utilidad. Además, en el último capítulo se dan los lineamientos generales para la elaboración de paquetes de funciones. Esto con la finalidad que el usuario obtenga el máximo provecho en el uso de *Maxima*.

Las nuevas herramientas incluidas en las últimas versiones de *Maxima*, así como la constante revisión y mejora de los aportes hechos por diferentes usuarios ha motivado esta segunda edición del manual.

R. IPANAQUÉ

Piura, Perú

CAPÍTULO 1

Obtención de *Maxima*

Maxima puede funcionar en distintos sistemas operativos, entre ellos diversas variantes de Windows y de GNU/Linux. En este capítulo se tratará acerca de la descarga e instalación de *Maxima* en el sistema operativo Windows (95 o posterior). El lector interesado en utilizar *Maxima* en alguna variante de GNU/Linux, puede acceder a la sección Download de la web de *Maxima* y seguir las instrucciones que en ella se indican.

1.1 Descarga

Maxima se descarga gratuitamente desde la página de **sourceforge** que alberga a una gran cantidad de instaladores de softwares de código abierto[1]. Debemos destacar que por el hecho de ser gratuito no requiere de ningún password que siempre está asociado con el software comercial (también llamado software propietario o más acertadamente *software privativo*).

La dirección específica donde esta alojado *Maxima* es la siguiente:

`http://sourceforge.net/projects/maxima/files`

[1] Código abierto (en inglés open source) es el término con el que se conoce al software distribuido y desarrollado libremente. El código abierto tiene un punto de vista más orientado a los beneficios prácticos de compartir el código que a las cuestiones morales y/o filosóficas las cuales destacan en el llamado software libre.

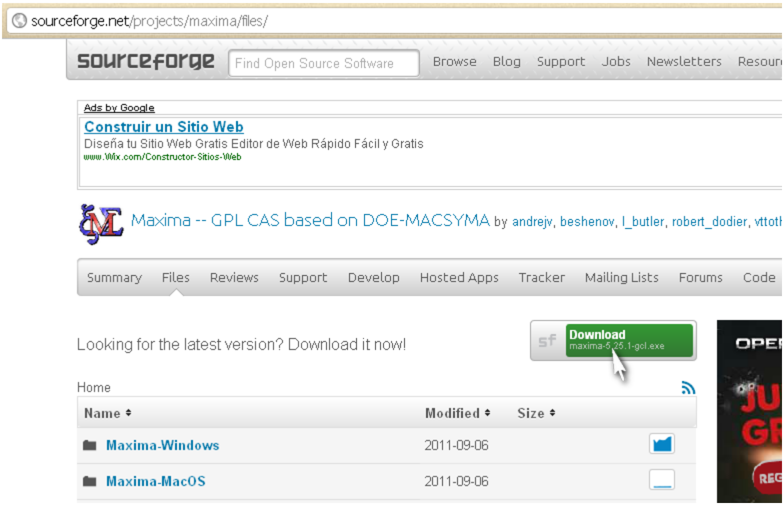


Figura 1.1: Porción de la página de descarga de `Maxima-5.25.1.exe`. El botón señalado permite la descarga directa de *Maxima* para Windows.

desde donde puede descargarse el archivo `Maxima-5.25.1.exe` que es el instalador de *Maxima* para Windows. Este instalador ocupa 29.0 MB de espacio en memoria.

Una vez descargado el instalador se verá un icono, como el que se aprecia en la figura 1.2, en la carpeta donde éste se haya descargado.

1.2 Instalación

Después de la descarga se procede a instalar *Maxima*, lo cual debe hacerse siguiendo los pasos que se detallan a continuación.

Figura 1.2: Icono del instalador de `Maxima-5.25.1.exe`.

1. Hacer doble clic sobre el icono del instalador.

2. Si aparece un cuadro como el de la figura 1.3, hacer clic sobre el botón Ejecutar.

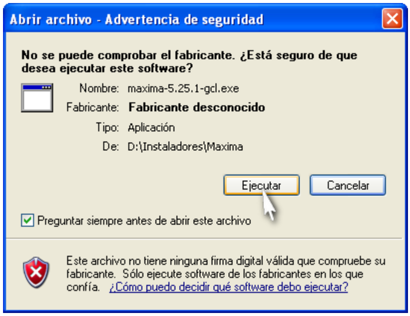


Figura 1.3: Cuadro de verificación.

3. Seleccionar el idioma Español y hacer clic sobre el botón Aceptar (fig. 1.4).

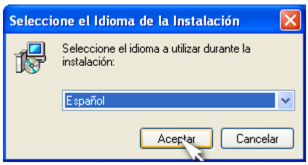


Figura 1.4: Cuadro para seleccionar el idioma.

4. Hacer clic sobre el botón Siguiente del cuadro **Bienvenido al asistente de instalación de Maxima** (fig. 1.5).

5. Seleccionar la opción Acepto el acuerdo del cuadro **Acuerdo de Licencia**. Luego hacer clic en el botón Siguiente del mismo cuadro (fig. 1.6).

6. Hacer clic en el botón Siguiente del cuadro **Información** (fig. 1.7).

7. Seleccionar la carpeta en la cual se quiere instalar Maxima (generalmente se deja la carpeta que aparece por defecto) y luego

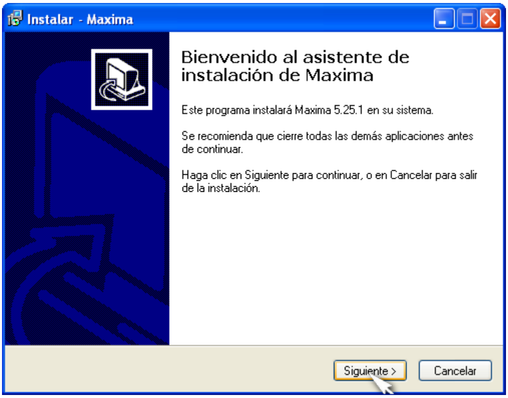


Figura 1.5: Cuadro **Bienvenido al asistente de instalación de Maxima**.

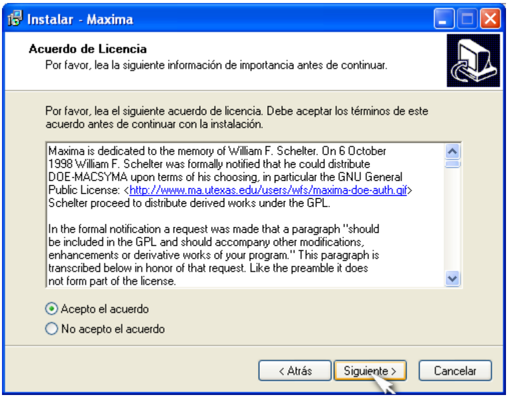


Figura 1.6: Cuadro **Acuerdo de Licencia**.

hacer clic en el botón Siguiente del cuadro **Seleccione la Carpeta de Destino** (fig. 1.8).

8. En el cuadro **Seleccione los Componentes** desmarcar las casillas Portugués y Portugués Brasileño ya que sólo utilizaremos el idioma Español. Esto permite, a su vez, el ahorro de memoria (fig. 1.9).

9. Seleccionar la carpeta del menú Inicio en la cual se quiere ubi-

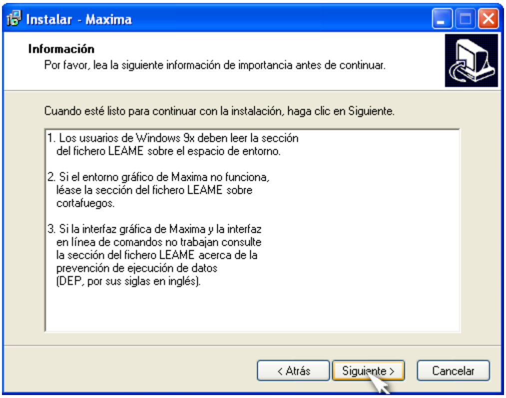


Figura 1.7: Cuadro **Información**.

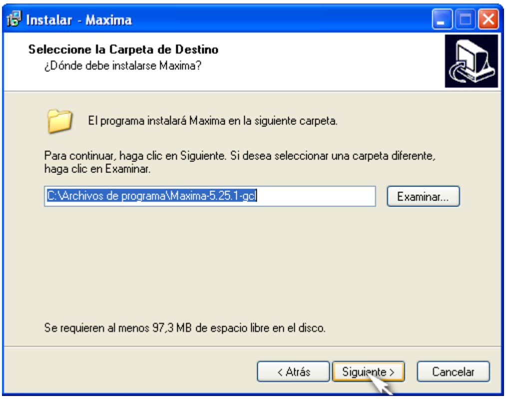


Figura 1.8: Cuadro **Seleccione la Carpeta de Destino**.

car el icono de acceso a *Maxima* (generalmente se deja la carpeta que aparece por defecto) y luego hacer clic en el botón Siguiente del cuadro **Seleccione la Carpeta del Menú Inicio** (fig. 1.10).

10. Hacer clic en el botón Siguiente del cuadro **seleccione las Tareas Adicionales** para que el asistente cree un icono de acceso directo a *Maxima* en el escritorio (fig. 1.11).

11. Hacer clic en el botón Instalar del cuadro **Listo para Instalar**

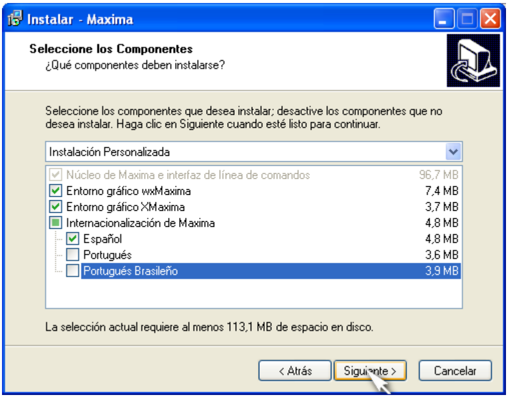


Figura 1.9: Cuadro **Seleccione los Componentes**.

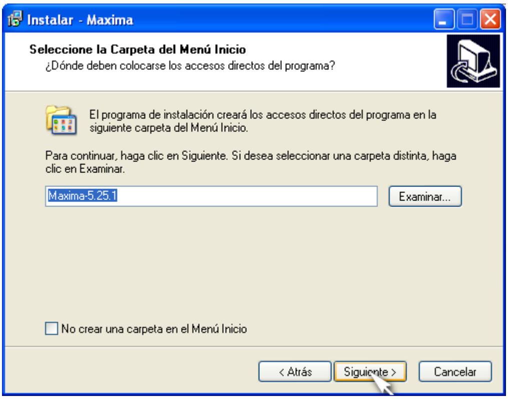


Figura 1.10: Cuadro **Seleccione la Carpeta del Menú Inicio**.

(fig. 1.12).

12. Hacer clic en el botón [Siguiente] del cuadro **Información** (fig. 1.13).

13. Por último, hacer clic en el botón [Finalizar] del cuadro **Completando la Instalación de Maxima** (fig. 1.14).

Después de seguir el procedimiento anterior deben haberse instalado: el núcleo de *Maxima* que es el responsable de todos los cálculos

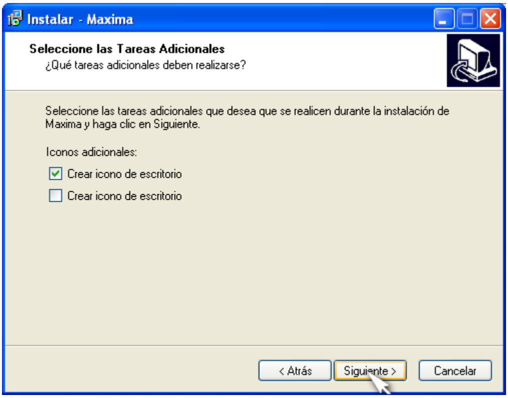


Figura 1.11: Cuadro **Seleccione las Tareas Adicionales**.

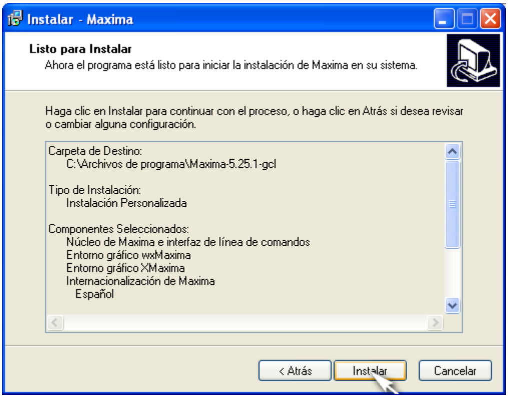


Figura 1.12: Cuadro **Listo para Instalar**.

y permite una interfaz de texto, el entorno gráfico *wxMaxima* que permite una interfaz gráfica (o interfaz de "cuaderno") bastante amigable, el entorno gráfico *XMaxima* que también permite una interfaz gráfica (aunque menos amigable que *wxMaxima*) y una aplicación para usuarios de habla hispana. Además, debe haberse creado automáticamente, en el escritorio de su ordenador (computadora), el icono de acceso directo al entorno gráfico *wxMaxima* (fig. 1.15).

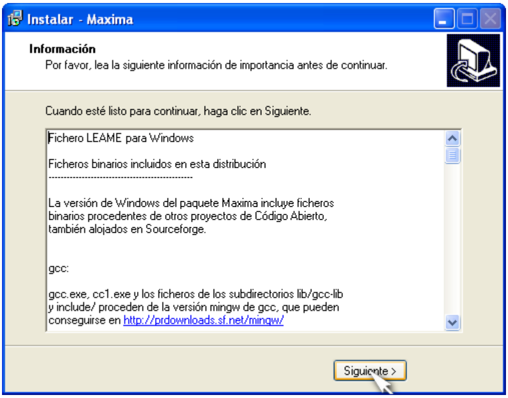


Figura 1.13: Cuadro **Información**.

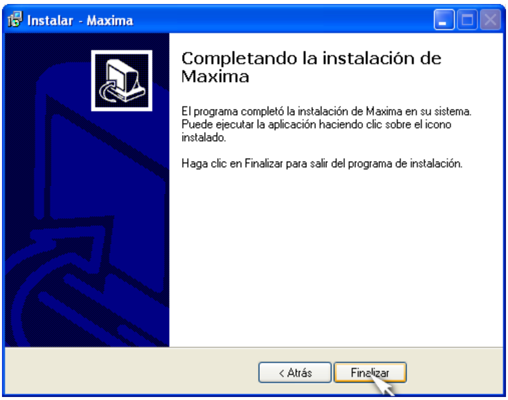


Figura 1.14: Cuadro **Completando la Instalación de *Maxima***.

Figura 1.15: Icono de acceso directo a *wxMaxima*.

CAPÍTULO 2

Funcionamiento de *Maxima*

2.1 Interfaz de cuaderno

utilice un icono o el menú de Inicio	formas gráficas de inicializar *Maxima*
finalizar texto con Shift Enter	entrada para *Maxima*
elegir el ítem salida del menú	salir de *Maxima*

Funcionamiento de *Maxima* en una interfaz de cuaderno.

El acceso a una interfaz de “cuaderno” es factible en un ordenador usado vía una interfaz puramente gráfica (como Windows). En una interfaz de cuaderno, es posible interactuar con *Maxima*, a través de *wxMaxima*, creando documentos interactivos. Para ello el usuario debe hacer doble clic en el icono de inicio de *wxMaxima*, después de lo cual se desplegará un cuaderno en blanco. En este cuaderno el usuario digita la entrada (input), luego presiona (en simultáneo) las teclas Shift Enter y *Maxima* añade punto y coma al final de tal entrada, etiqueta la entrada con (%i*n*), la procesa y devuelve la correspondiente salida (output) etiquetada con (%o*n*).

Maxima

El usuario digita `1+1`, luego finaliza su entrada con Shift Enter. *Maxima* añade punto y coma al final de ésta, la etiqueta con `(%i1)`, la procesa e inmediatamente después devuelve la respectiva salida etiquetada con `(%o1)`.

```
(%i1) 1+1;
(%o1) 2
```

Debe recordarse que los cuadernos corresponden al entorno gráfico *wxMaxima*. El núcleo de *Maxima* es el que realiza realmente los cálculos (sección 3.1).

Para salir de *wxMaxima*, el usuario elige el ítem salida del respectivo menú en la interfaz de cuaderno.

2.2 Interfaz basada en texto

`maxima`	comando del sistema operativo para inicializar *Maxima*
finalizar texto con ";" y Enter	entrada para *Maxima*
quit();	salir de *Maxima*

Funcionamiento de *Maxima* en una interfaz basada en texto.

Con una interfaz basada en texto, el usuario interactúa con su ordenador digitando texto mediante el teclado.

Para inicializar *Maxima* en una interfaz basada en texto, se digita el comando `maxima` en el prompt del sistema operativo. Cuando *Maxima* ha inicializado, imprimirá el prompt `(%i1)`, esto significa que esta lista para que el usuario haga su entrada. Éste puede entonces digitar su entrada, terminándola con ";" y presionando luego Enter.

Maxima procesa la entrada y genera un resultado, el mismo que etiquetará con `(%o1)`.

Obsérvese que la mayor parte de los diálogos dados en el libro muestran salidas en la forma que se obtendrían con una interfaz de cuaderno de *Maxima*; la salida con una interfaz basada en texto luce similar, pero carece de características tales como caracteres especiales y cambio de tamaño de fuente.

Para salir de *Maxima*, debe digitarse `Quit();` en el prompt de la entrada.

CAPÍTULO 3

Uso del sistema *Maxima*

3.1 La estructura de *Maxima*

Maxima	núcleo responsable de todos los cálculos
wxMaxima	interfaz de cuaderno que se ocupa de interactuar con el usuario (muy amigable)
XMaxima	interfaz gráfica que se ocupa de interactuar con el usuario (menos amigable que *wxMaxima*)

Partes básicas del Sistema *Maxima*.

Maxima es un potente motor de cálculo simbólico aunque, en su origen, no destacaba por tener una interfaz gráfica más amigable para los usuarios que la simple consola de texto. Con el tiempo este hecho ha ido cambiando y han aparecido distintos entornos de ejecución que intentan facilitar la interacción con los usuarios. Entre ellos, están *XMaxima* y *wxMaxima*.

XMaxima es la primera interfaz gráfica que fue desarrollada, es mantenida "oficialmente" por el equipo de desarrollo de *Maxima*. En Windows se instala automáticamente. Presenta algunas ventajas como la integración en formato HTML de manuales de ayuda. Sin em-

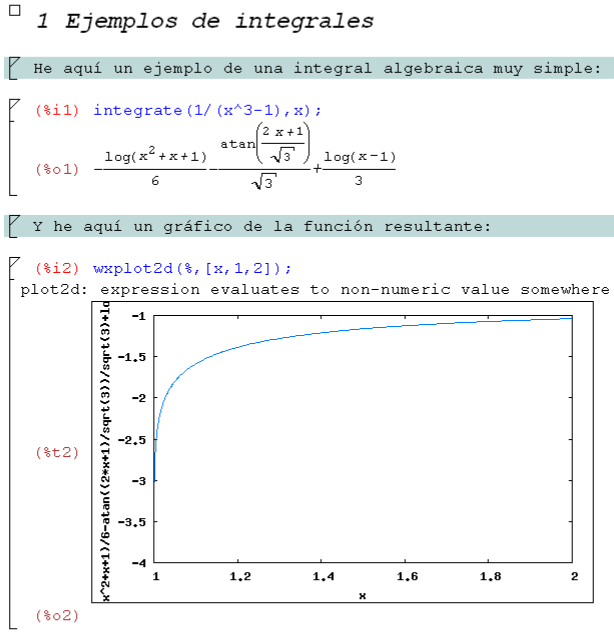


Figura 3.1: Un cuaderno que mezcla texto, gráficos con entradas y salidas de *Maxima*.

bargo, también tiene algunas desventajas con respecto a otras interfaces más modernas.

wxMaxima[1], basada en la biblioteca gráfica *wxwidgets*, gracias a la cual existen versiones nativas tanto para sistemas operativos GNU/Linux como para Windows. Integra elementos específicos para la navegación de la ayuda, introducción de matrices, creación de gráficas, cálculo de límites, derivadas o integrales, etc. Actualmente también se instala automáticamente en Windows.

[1] *wxMaxima* fue desarrollada por Andrej Vodopivec y está disponible en http://wxmaxima.sourceforge.net

```
Línea de comandos de Maxima
Maxima 5.25.1 http://maxima.sourceforge.net
using Lisp GNU Common Lisp (GCL) GCL 2.6.8 (a.k.a. GCL)
Distributed under the GNU Public License. See the file COPYING.
Dedicated to the memory of William Schelter.
The function bug_report() provides bug reporting information.
(%i1) 2^100;
(%o1)                    1267650600228229401496703205376
(%i2) integrate(1/(x^3-1),x);
                                            2 x + 1
                          2           atan(-------)
                     log(x  + x + 1)        sqrt(3)    log(x - 1)
(%o2)              - --------------- - ------------- + ----------
                            6             sqrt(3)           3
(%i3) diff(%,x);
                             2               2 x + 1           1
(%o3)             - ------------------ - -------------- + ----------
                                  2               2        3 (x - 1)
                         (2 x + 1)         6 (x  + x + 1)
                     3 (---------- + 1)
                             3
(%i4) radcan(%);
                                              1
(%o4)                                       ------
                                             3
                                            x  - 1
(%i5) %o1/(2^98);
(%o5)                                         4
(%i6) factor(%o4);
                                              1
(%o6)                              --------------------
                                                 2
                                   (x - 1) (x  + x + 1)
(%i7) _
```

Figura 3.2: Un diálogo con *Maxima* usando una interfaz basada en texto.

interfaz de cuaderno con *wxMaxima*	documentos interactivos
interfaz basada en texto	texto desde el teclado

Tipos comunes de interfaz con *Maxima*.

En algunos casos, puede que el usuario no necesite usar la interfaz de cuaderno, y que desee en cambio interactuar directamente con el núcleo de *Maxima*. Es posible hacer esto usando la interfaz basada en texto, en la cual se digita el texto en el teclado y éste va directamente al núcleo.

3.2 Cuadernos como documentos

Los cuadernos de *wxMaxima* permiten crear documentos que pueden verse interactivamente en la pantalla o imprimirse en papel. En los

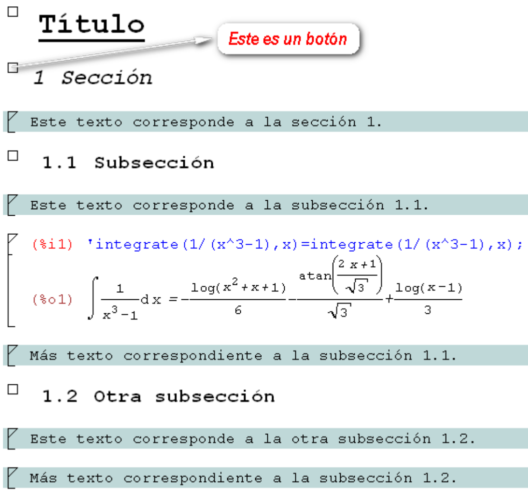


Figura 3.3: Un cuaderno de *wxMaxima* como documento.

cuadernos extensos, es común tener los capítulos, secciones etc., representados cada uno en grupos de celdas. La extensión de estos grupos de celdas es indicada por el botón asociado a la celda dominante que es una celda de estilo título, sección o subsección.

Un grupo de celdas puede estar abierto o cerrado. Cuando está abierto se puede ver todas sus celdas explícitamente. Pero cuando está cerrado, sólo puede verse la celda que encabeza el grupo de celdas.

Los cuadernos extensos son a menudo distribuidos con muchos grupos de celdas cerradas, para que cuando sean vistos por primera vez el cuaderno muestre solamente una lista de su contenido. Es posible abrir las partes en las que el usuario esté interesado haciendo clic sobre el botón apropiado.

A cada celda dentro de un cuaderno se le asigna un estilo en particular que indica su rol dentro del cuaderno.

La interfaz de *wxMaxima* provee menúes y métodos abreviados de teclas para insertar celdas con diferentes estilos todos ellos están disponibles en el último bloque del menú Cell.

Así, por ejemplo, el material entendido como entrada para ser ejecutado por el núcleo de *Maxima* está en el estilo de Input (entrada),

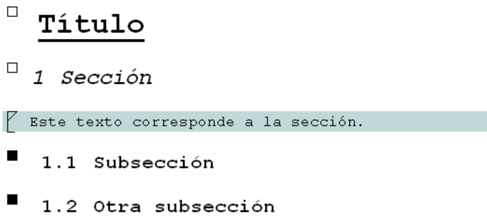


Figura 3.4: Haciendo clic sobre el botón que corresponde a la celda dominante se cierra el grupo, dejando sólo la primera celda visible.

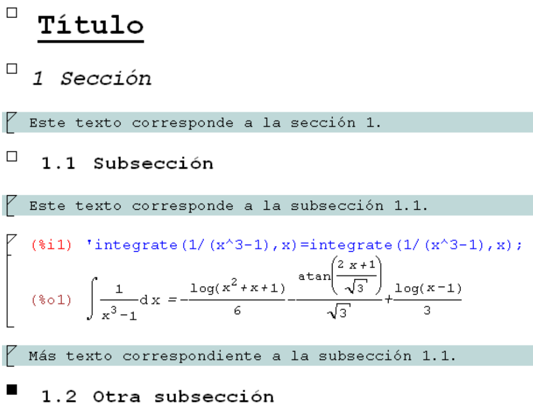


Figura 3.5: Cuando el grupo está cerrado, el botón que le corresponde aparece relleno en color negro. Haciendo clic sobre este botón se abre nuevamente el grupo.

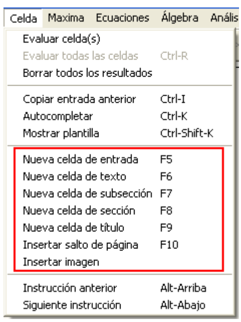


Figura 3.6: El recuadro muestra los menúes y métodos abreviados de teclas para insertar celdas con diferentes estilos.

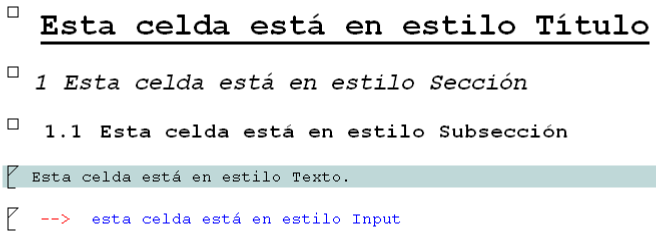


Figura 3.7: Esto muestra celdas en diferentes estilos. Los estilos no sólo definen el formato del contenido de las celdas, sino que también su ubicación y espaciado.

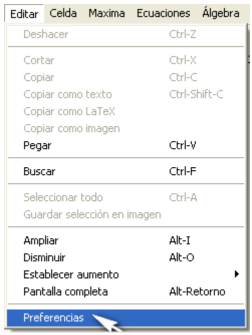


Figura 3.8: Primer paso para configurar las opciones y estilos.

mientras que el que se entiende para ser leído como solamente de texto está en estilo Text (texto).

3.3 Configuración de opciones y estilos

Como se vio en la sección 3.2 los cuadernos pueden editarse a manera de documentos. *wxMaxima* incorpora una configuración predefinida para los estilos de los títulos, secciones, etc. Sin embargo, es posible cambiar algunos aspectos de dicha configuración haciendo clic en la opción Preferencias del menú Editar.

Después de hacer clic en la opción Preferencias se despliega la ventana Configuración de wxMaxima que incorpora dos pestañas: Opciones y Estilo.

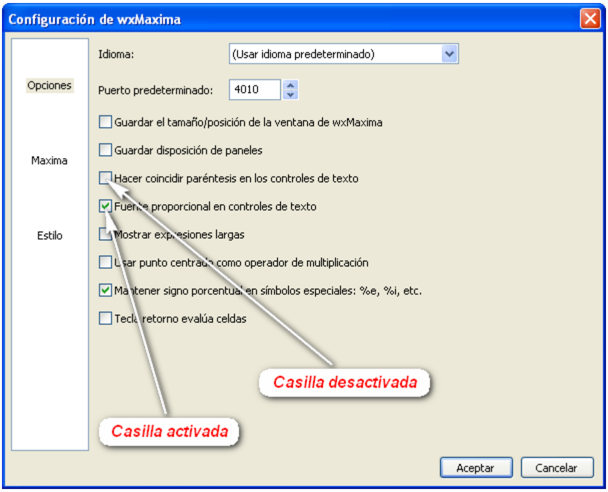


Figura 3.9: Activando o desactivando las casillas de verificación se cambia la configuración de las opciones.

Cuadro 3.1: Valores asignados en la configuración de Fuentes

Fuentes	Tipo
Fuente predeterminada	Courier New (12)
Fuente matemática	Courier New (12)

Por ejemplo, cuando está activa la casilla de verificación de la opción Hacer coincidir los paréntesis en los controles de texto (de la pestaña Opciones), *wxMaxima* cierra automáticamente cualquier paréntesis que se abra en una celda de estilo Input. Al desactivar esta casilla, y hacer clic en Aceptar, *wxMaxima* no volverá a cerrar automáticamente ningún paréntesis sino que esperará a que el usuario lo haga.

En la pestaña Estilo se presenta una lista de todos los estilos que pueden configurarse a gusto del usuario y la forma de hacerlo es bastante intuitiva.

Por ejemplo, configurando los estilos con los valores indicados en los cuadros 3.1 y 3.2 se obtienen cuadernos con un aspecto elegante.

Cuadro 3.2: Valores asignados en la configuración de Estilos

Estilos	Color	Fuente	Aspecto	Tam.
Nombre de funciones	rgb(0,0,0)	Courier New	Gruesa, Itálica	12
Celda de texto	rgb(0,0,0)	Tahoma	Normal	12
Celda de subsección	rgb(188,73,18)	Tahoma	Gruesa	16
Celda de sección	rgb(188,73,18)	Tahoma	Gruesa	18
Celda de título	rgb(54,95,145)	Tahoma	Gruesa	24
Fondo de celda de texto	rgb(252,250,245)			
Fondo	rgb(252,250,245)			

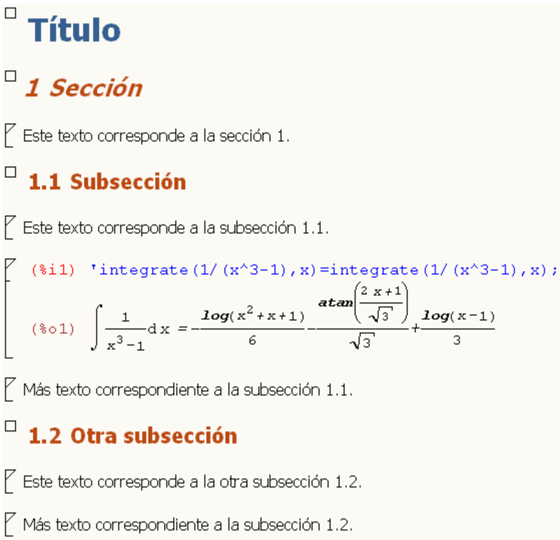


Figura 3.10: Un cuaderno de *wxMaxima* como documento, después de haber editado la configuración de estilo.

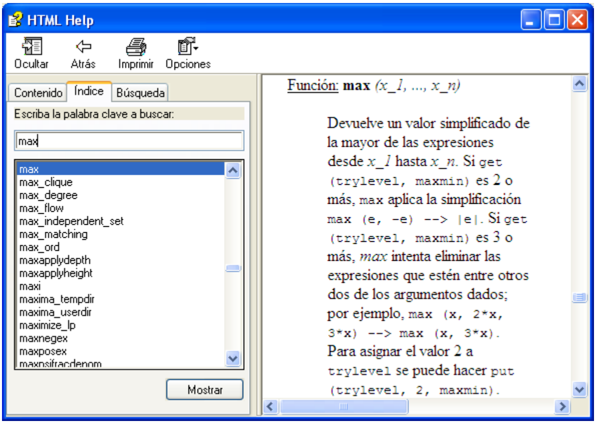


Figura 3.11: Un ejemplo de búsqueda de información básica sobre una función en el Índice de la Ayuda de *Maxima*

3.4 Búsqueda de ayuda

Todas las funciones incorporadas en *Maxima* están descritas en el manual en línea del usuario, el cual puede ser consultado en diferentes formas. La más usada es desde el menú Ayuda, de la barra de menúes, que da acceso a la opción Ayuda de Maxima la cual sirve como un punto de entrada a la gran cantidad de documentación en línea para *Maxima*.

También es factible buscar ayuda desde un cuaderno de trabajo. Para ello puede utilizarse la función `describe`[2] o también el símbolo especial `?`.

[2]`describe` no evalúa su argumento. La función `describe` devuelve `true` si encuentra la documentación solicitada y `false` en caso contrario.

`describe(`*string*`)`	encuentra el elemento, si existe, cuyo título coincide exactamente con *string* (ignorando la diferencia entre mayúsculas y minúsculas)
`describe(`*string,exact*`)`	equivale a `describe(`*string*`)`
`describe(`*string,inexact*`)`	encuentra todos los elementos documentados que contengan *string* en sus títulos

Sintaxis de la función `describe`, la cual permite recibir información de las funciones de *Maxima*.

`?name`	equivale a `describe("name")`
`??name`	equivale a `describe("name",inexact)`

Otras formas de recibir información.

Maxima

Esta sentencia da información de la función incorporada `max`.

```
(%i1) describe("max");
```

--Función: max(<x_1>,...,<x_n>)

Devuelve un valor simplificado de la mayor de las expresiones desde <x_1> hasta <x_n>. Si ‘get(trylevel,maxmin)’ es 2 o más, ‘max’ aplica la simplificación ‘max(e,e)–>|e|’. Si ‘get(trylevel,maxmin)’ es 3 o más, ‘max’ intenta eliminar las expresiones que estén entre dos de los argumentos dados; por ejemplo, ‘max(x,2*x,3*x)–>max(x,3*x)’. Para asignar el valor 2 a ‘trylevel’ se puede hacer ‘put(trylevel,2,maxmin)’.

There are also some inexact matches for ‘max’.

Try ‘?? max’ to see them.

(%o1) true

Maxima

Esta sentencia encuentra todos los elementos documentados que contienen `"plus"` en sus títulos. No devuelve `true` o `false` hasta que el usuario seleccione las opciones que desee consultar (aquí las opciones disponibles son: 0,1,2,3, all y none).

```
(%i2) describe("plus",inexact);
```

0: doscmxplus (Funciones y variables para las matrices y el álgebra lineal).
1: poisplus (Series de Poisson)
2: region_boundaries_plus (Funciones y variables para worldmap)
3: trigexpandplus (Funciones y variables para trigonometría)

Enter space-separated numbers, 'all' or 'none':

Maxima

Una vez elegidas las opciones (en este caso 0 y 2) la sentencia devuelve `true`.

```
(%i3) describe("plus",inexact);
```

0: doscmxplus (Funciones y variables para las matrices y el álgebra lineal).
1: poisplus (Series de Poisson)
2: region_boundaries_plus (Funciones y variables para worldmap)
3: trigexpandplus (Funciones y variables para trigonometría)

Enter space-separated numbers, 'all' or 'none': 0 2:

--Variable opcional: doscmxplus

Valor por defecto: 'false'.
Cuando 'doscmxplus' vale 'true', las operaciones entre escalares y matrices dan como resultado una matriz.

--Función : region_boundaries_plus(<x1>,<y1>,<x2>,<y2>)

Detecta los segmentos poligonales almacenados en la variable global ‘boundaries_array’ con al menos un vértice dentro del rectángulo definido por los extremos (<x1>,<y1>) -superior izquierdo- y (<x2>,<y2>) -inferior derecho-.
Ejemplo.

```
(%i1) load(worldmap)$;
(%i2) region_boundaries(10.4,41.5,20.7,35.4);
(%o2) [1846, 1863, 1864, 1881, 1888, 1894]
(%i3) draw2d(geomap(%))$
```

(%o3) *true*

3.5 Reinicio

La forma brusca de reiniciar *Maxima* es saliendo de *wxMaxima*. No obstante, en muchos casos resulta útil reiniciar *Maxima* sin salir de *wxMaxima*. Para reiniciar *Maxima* sin salir de *wxMaxima* se elige la opción Reiniciar Maxima del menú Maxima.

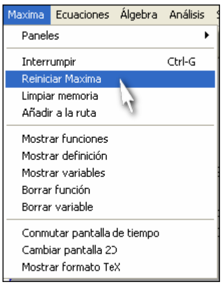


Figura 3.12: Reiniciando *Maxima* en una interfaz de cuaderno.

3.6 Comentarios

Los comentarios son toda una serie de caracteres que no afectan los cálculos. En *Maxima* los comentarios se escriben entre las marcas /* y */.

$/*comentario/*$	con esta sintaxis *comentario* es interpretado como un comentario

Escribiendo comentarios.

Maxima

Aquí se muestra un cálculo y un comentario.

```
(%i1) 4+5 /*esto es una suma*/;
(%o1) 9
```

3.7 Paquetes en *Maxima*

Una de las características más importantes de *Maxima* es que es un sistema extensible, hay una cierta cantidad de funciones incorporadas en *Maxima* pero, usando el lenguaje de programación de *Maxima*, siempre es posible añadir más funciones.

Para muchos tipos de cálculos, lo incorporado en la versión estándar de *Maxima* será suficiente. Sin embargo, si se trabaja en particular en un área especializada, es posible encontrarse en la necesidad de utilizar ciertas funciones no incorporadas en *Maxima*.

En tales casos, podría ser factible encontrar (o elaborar) un package (paquete) de funciones de *Maxima* que contenga las funciones que sean necesarias.

`load`(“*paquete*”)	lee un paquete de *Maxima*

Leyendo paquetes de *Maxima*.

Si el usuario quiere usar las funciones de un paquete en particular, primero debe inicializar el paquete en *Maxima*.

Maxima

Con estas sentencias se está inicializando y utilizando una función de un paquete en particular de *Maxima*.

```
(%i1) load(“simplex”)$
```

```
(%i2) minimize_lp(x+y,[3*x+2*y>2,x+4*y>3]);
```

(%o2) $[\frac{9}{10}, [y = \frac{7}{10}, x = \frac{1}{5}]]$

El hecho de que *Maxima* pueda extenderse usando paquetes significa que las posibilidades de *Maxima* son ilimitadas. En lo que al uso concierne, no hay en realidad ninguna diferencia entre las funciones definidas en paquetes y las funciones incorporadas en *Maxima*.

3.8 Advertencias y mensajes

Maxima “sigue” el trabajo del usuario silenciosamente, dando salida solamente cuando éste lo requiere. Sin embargo, si *Maxima* se percata de algo que se pretende hacer y que definitivamente no entiende, imprimirá un mensaje de advertencia.

Maxima

La función para calcular la raíz cuadrada debe tener solamente un argumento. *Maxima* imprime un mensaje para advertir que, en este caso, se ha errado en el número de argumentos.

```
(%i1) sqrt(4,5);
```

sqrt: wrong number of arguments. -- an error. To debug this try: debugmode(true);

3.9 Interrupción de cálculos

Probablemente habrá veces en que el usuario desee detener *Maxima* en medio de un cálculo. Tal vez él se da cuenta que pidió a *Maxima* hacer un cálculo incorrecto. O quizás el cálculo tarda demasiado, y quiere saber que es lo que pasa. La forma en que se interrumpe un cálculo en *Maxima* depende de qué clase de interfaz está utilizando.

Clic en el botón	interfaz de cuaderno
Ctrl + C	interfaz basada en texto

Formas de interrumpir cálculos en *Maxima*.

CAPÍTULO 4

Cálculos numéricos

4.1 Aritmética

Los cálculos aritméticos se realizan con números literales (enteros, racionales, decimales ordinarios y decimales grandes). Excepto en el caso de la exponenciación, todas las operaciones aritméticas con números dan lugar a resultados en forma de números.

El usuario puede hacer aritmética con *Maxima* tal y como lo haría con una calculadora

Maxima

Aquí tenemos la suma de dos números.

```
(%i1) 5.6+3.7;
(%o1) 9.3
```

Maxima

Con $*$ indicamos el producto de dos números.

```
(%i2) 5.6*3.7;
(%o2) 20.72
```

Maxima

Es posible digitar operaciones aritméticas haciendo uso de los paréntesis.

```
(%i3) (2+3)^3-4*(6+7);
```

(%o3) 73

$x\hat{}y$ ó $x{*}{*}y$	potencia
$-x$	menos
x/y	división
$x{*}y{*}z$	producto
$x+y+z$	suma

Operaciones aritméticas en *Maxima*.

Las operaciones aritméticas en *Maxima* se agrupan de acuerdo con las convenciones estándares de la matemática. Como es usual, `2+3/7`, por ejemplo, significa `2+(3/7)`, y no `(2+3)/7`. El usuario siempre puede controlar la forma de agrupar explícitamente usando los paréntesis.

4.2 Resultados exactos y aproximados

Una calculadora electrónica hace todos sus cálculos con una precisión determinada, digamos de diez dígitos decimales. Con *Maxima*, en cambio, es posible obtener resultados exactos.

Maxima

Maxima da un resultado exacto para 2^{300}.

```
(%i1) 2^300;
(%o1) 2037035976334486086268445688409378161051468393 6
      659362506361404493543812997633367061833973 76
```

El usuario puede pedir a *Maxima* que devuelva un resultado aproximado, tal como lo daría una calculadora, para ello puede usar la función `float` o la variable `numer` o una combinación de ambos.

Maxima

Esto da un resultado numérico aproximado.

```
(%i2) 2^300,float;
(%o2) 2.0370359763344861 10^90
```

`float`(*expr*)	da un valor numérico aproximado para ciertas *expr*
expr, `float`	equivale a `float`(*expr*)
expr, `numer`	da un valor numérico aproximado para ciertas *expr*
`float`(*expr*), `numer`	da un valor numérico aproximado para cualquier *expr* que no sea una constante

Obteniendo aproximaciones numéricas.

Maxima

Esta forma también da un resultado numérico aproximado.

(%i3) `float(2^300);`

(%o3) $2.0370359763344861\ 10^{90}$

Maxima

Para el cálculo previo la constante **`numer`** no es útil.

(%i4) `2^300,numer;`

(%o4) 2037035976334486086268445688409378161051468393665936250636140449354381299763336706183397376

Maxima

Maxima puede dar resultados en términos de números racionales.

(%i5) `1/3+2/7;`

(%o5) $\frac{13}{21}$

Maxima

En este caso, tanto con **`float`** como con **`numer`**, se obtiene un resultado numérico aproximado.

(%i6) `1/3+2/7,float;`

(%o6) 0.61904761904762

```
(%i7) 1/3+2/7,numer;
(%o7) 0.61904761904762
```

Cuando el usuario digita un entero como 7, *Maxima* asume que es exacto. Si digita un número como 4.5, con un punto decimal explícito, *Maxima* asume que desea efectuar cálculo numérico aproximado.

Maxima

Esto es tomado como un número racional exacto, y es llevado a una fracción irreducible.

```
(%i8) 26/78;
```
(%o8) $\frac{1}{3}$

Maxima

Cuando el usuario digita un número con un punto decimal explícito, *Maxima* produce un resultado numérico aproximado.

```
(%i9) 26.7/78;
(%o9) 0.34230769230769
```

Maxima

Aquí, la presencia del punto decimal seguido del cero hace que *Maxima* dé un resultado numérico aproximado.

```
(%i10) 26.0/78;
(%o10) 0.33333333333333
```

Maxima

Cuando cualquier número en una expresión aritmética es digitado con un punto decimal seguido del cero, el usuario obtiene un resultado numérico aproximado.

```
(%i11) 5.0+9/78-5/8;
(%o11) 4.490384615384615
```

4.3 Algunas funciones matemáticas

Maxima incluye una gran colección de funciones matemáticas. A continuación se mencionan las más comunes.

`sqrt`(x)	raíz cuadrada ($\sqrt{x}$)
`exp`(x)	exponencial (e^x)
`log`(x)	logaritmo neperiano ($\log_e x$)
`sin`(x), `cos`(x), `tan`(x), `cot`(x), `sec`(x), `csc`(x)	funciones trigonométricas (con argumentos en radianes)
`asin`(x), `acos`(x), `atan`(x), `acot`(x), `asec`(x), `acsc`(x)	funciones trigonométricas inversas
$n!$	factorial de n (producto de los enteros $1, 2, \ldots, n$)
$n!!$	$1\times3\times\ldots\times n$ (n impar) ó $2\times4\times\ldots\times n$ (n par)
`abs`(x)	valor absoluto
`round`(x)	redondeo
`mod`(n,m)	n módulo m (resto de la división de n entre m)
`floor`(x)	mayor entero menor o igual que x
`ceiling`(x)	menor entero mayor o igual que x
`random`(x)	número seudo aleatorio r, tal que $0 \leq r < x$, si $x \in \mathbb{N}$ ó $0 < r < x$, si $x \in \mathbb{R}^+$
`max`(x,y,$\ldots$), `min`(x,y,$\ldots$)	máximo, mínimo de x,y,$\ldots$
`ifactor`(n)	factores primos de n

Algunas de las funciones matemáticas más comunes.

- Los argumentos de todas las funciones en *Maxima* se colocan entre paréntesis.
- Los nombres de las funciones incorporadas en *Maxima* empiezan con letra minúscula.

Dos puntos importantes acerca de funciones en *Maxima*.

Maxima

Esto da $\log_e 15.7$.

```
(%i1) log(15.7);
(%o1) 2.753660712354262
```

Maxima

Maxima no incluye una función para el logaritmo de base 10 u otras bases. Para salvar esta dificultad el usuario puede hacer uso de la fórmula $\log_b x = \frac{\log_e x}{\log_e b}$. Así, por ejemplo, lo siguiente devuelve un resultado numérico para $\log_2 1024$.

```
(%i2) log(1024)/log(2),numer;
(%o2) 10.0
```

Maxima

Esto devuelve $\sqrt{64}$ como un número exacto.

```
(%i3) sqrt(64);
(%o3) 8
```

Maxima

Esto da un valor numérico aproximado para $\sqrt{6}$.

```
(%i4) sqrt(6),numer;
(%o4) 2.449489742783178
```

Maxima

La presencia explícita de un punto decimal seguido de un cero le indica a *Maxima* que dé un resultado numérico aproximado.

```
(%i5) sqrt(6.0);
(%o5) 2.449489742783178
```

Maxima

En este caso *Maxima* devuelve un número en forma simbólica exacta.

```
(%i6) sqrt(6);
```

(%o6) $\sqrt{6}$

Maxima

Aquí tenemos un resultado entero exacto para $40 \times 39 \times \ldots \times 1$.

```
(%i7) 40!;
(%o7) 815915283247897734345611269596115894272000000000
```

Maxima

Esto da un valor numérico aproximado del factorial.

```
(%i8) float(40!);
(%o8) 8.1591528324789768 10^47
```

`%e`	$e \approx 2.718281828459045$
`%i`	$i = \sqrt{-1}$
`inf`	representa al infinito real positivo
`minf`	representa al infinito real negativo
`infinity`	representa al infinito complejo
`und`	representa un resultado indefinido
`%pi`	$\pi \approx 3.141592653589793$

Algunas constantes matemáticas comunes.

Maxima

Este es el valor numérico de π^2.

```
(%i9) %pi^2,numer;
(%o9) 9.869604401089358
```

Maxima

Esto devuelve el valor exacto para sen$(\pi/2)$.

```
(%i10) sin(%pi/2);
(%o10) 1
```

4.4 Cálculos con precisión arbitraria

Cuando el usuario utiliza `float`, `numer` o una combinación de ambos para obtener un resultado numérico, *Maxima* devuelve el resultado con un número fijo de cifras significativas. No obstante, es posible indicar a *Maxima* las cifras significativas con las que se desea operar. Esto permite obtener resultados numéricos en *Maxima* con cualquier grado de precisión.

`fpprec :`n`$ bfloat(`*expr*`)` ó `fpprec :`n`$` *expr*`, bfloat`	valor numérico de *expr* calculado con n dígitos de precisión (el valor por defecto es 16)

Evaluación numérica con precisión arbitraria.

Maxima

Esto devuelve el valor numérico de π con un número fijo de cifras significativas.

```
(%i1) float(%pi);
(%o1) 3.141592653589793
```

Maxima

Esto devuelve π con 50 dígitos.

```
(%i2) fpprec : 50$ bfloat(%pi);
(%o2) 3.14159265358979323846264338327950288419716939937
      3751b0
```

Cabe mencionar que el símbolo de dolar que aparece después del número que indica la cantidad de dígitos significativos se utiliza, en general, para finalizar una sentencia y, a diferencia del punto y coma, no permite que aparezca ninguna salida en pantalla (subsec. 5.3).

Al realizar cualquier tipo de cálculo numérico el usuario puede introducir pequeños errores de redondeo en sus resultados. Cuando se aumenta la precisión numérica estos errores se hacen más pequeños. Asegurarse que se obtiene la misma respuesta al aumentar la precisión numérica es a menudo una buena forma de verificar los resultados.

Maxima

La cantidad $e^{\pi\sqrt{163}}$ esta bastante próxima a ser entera. Para verificar que el resultado no es un entero, el usuario tiene que usar la precisión numérica suficiente.

```
(%i3) fpprec:40$  bfloat(exp(%pi*sqrt(163)));
```

(%o3) $2.625374126407687439999999999992500725972b17$

El usuario que desee, por ejemplo, visualizar π con una precisión de 200 cifras significativas, o 2^{1000} con el total de las cifras, utilizando las sentencias aquí descritas encontrará que la salida se muestra truncada en la parte central, en donde aparece un número que indica la cantidad de dígitos faltantes.

Maxima

Esto devuelve una salida de π con 200 dígitos, la cual ha sido truncada. El dato entre los corchetes indica que se han obviado 443 dígitos

```
(%i4) fpprec:200$  bfloat(%pi);
```

(%o4) $3.141592653589793238462643383 2[443 digits]885752724$
$89122793818301194912b0$

Para obtener una salida completa se selecciona ascii como el *algoritmo de salida matemática* de la opción Cambiar pantalla 2D del menú Maxima y luego se ejecutan las sentencias de `(%i4)`.

Téngase presente que el algoritmo de salida matemática, previamente seleccionado (ascii), prevalecerá hasta que el usuario vuelva a seleccionar como algoritmo de salida matemática a xml.

Maxima

Después de cambiar el *algoritmo de salida matemática* se devuelve la salida π con 200 dígitos.

```
(%i5) fpprec : 200$ bfloat(%pi);
```

(%o5) 3.14159265358979323846264338327950288419716939 9
37510582097494459230781640628620899862803482534
21170679821480865132823066470938446095505822317
25359408128481117450284102701938521105559644622
948954930382b0

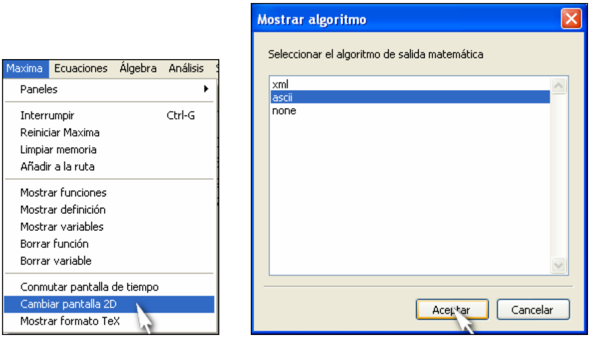


Figura 4.1: Seleccionando ascii como el *algoritmo de salida*.

4.5 Números complejos

Es posible ingresar números complejos en *Maxima* con sólo incluir la constante `%i`, igual a $\sqrt{-1}$. Note que una `i` ordinaria significa una variable llamada i, pero no $\sqrt{-1}$.

x+ `%i`*y	el número complejo $x + i\ y$
`realpart(`z`)`	parte real
`imagpart(`z`)`	parte imaginaria
`conjugate(`z`)`	complejo conjugado z^* ó $\bar{z}$
`cabs(`z`)`	módulo de z
`carg(`z`)`	el argumento φ en $\lvert z\rvert e^{i\varphi}$

Operaciones con números complejos.

Maxima

Esto devuelve como resultado el número imaginario $2i$.

```
(%i1) sqrt(-4);
(%o1) 2 %i
```

Maxima

Esto devuelve la división de dos números complejos.

```
(%i2) (8+4*%i)/(-1+%i),rectform;
```

(%o2) $-6\,\%i-2$

Maxima

Aquí tenemos el valor exacto de un exponencial complejo.

```
(%i3) exp(11+5*%i),rectform;
```

(%o3) $\%e^{11}\,\%i\sin(5)+\%e^{11}\cos(5)$

Maxima

Aquí tenemos el valor numérico de un exponencial complejo.

```
(%i4) exp(11+5*%i),numer;
```

(%o4) $16984.02989166794-57414.76791532402\,\%i$

CAPÍTULO 5

Generación de cálculos

5.1 Uso de entradas y salidas previas

Al realizar cálculos, muchas veces se necesita usar expresiones previamente ingresadas u obtenidas. En *Maxima*, _ y % siempre hacen referencia a la última expresión de entrada y salida, respectivamente.

_	la última expresión de entrada
%	la última expresión de salida
%in	la expresión de la entrada %in
%on	la expresión de la salida %on
%th(i)	la expresión de la i-ésima salida anterior

Formas de hacer referencia a expresiones de entrada y salida previas.

Para efectos didácticos, a partir de esta sección se supone un reinicio de *Maxima* (sec. 3.5).

Maxima

Aquí se tienen las expresiones de la primera entrada y salida.

```
(%i1) 5^3;
(%o1) 125
```

Maxima

Esto agrega 6 a la expresión de la última salida.

```
(%i2) %+6;
(%o2) 131
```

Maxima

Esto utiliza las dos expresiones de las salidas previas.

```
(%i3) 5+%o1+%;
(%o3) 261
```

Maxima

Esto suma las expresiones de las salidas `(%o1)` y `(%o3)`.

```
(%i4) %o1+%o3;
(%o4) 386
```

Maxima

Aquí se eleva al cuadrado la expresión de la salida `(%o2)`.

```
(%i5) %o2^2;
(%o5) 17161
```

Si se utiliza una interfaz basada en texto en *Maxima*, entonces las líneas sucesivas de entradas y salidas aparecerán siempre en orden. Sin embargo, si se utiliza una interfaz de cuaderno, varias líneas sucesivas de entradas y salidas no necesariamente aparecen en orden. Es posible, por ejemplo, "volver atrás" e insertar el cálculo siguiente dondequiera que se desee en el cuaderno. Téngase en cuenta que % siempre invoca el último resultado que *Maxima* generó. Éste puede o no ser el resultado que aparece inmediatamente encima de su actual posición en el cuaderno. Con una interfaz de cuaderno, la única manera de saber cuándo un resultado particular fue generado es mirar la etiqueta de `(%on)` que tiene. Como es posible insertar y suprimir en todas partes en un cuaderno, de acuerdo a la necesidad del usuario, el ordenamiento de los resultados, por lo general, no tiene ninguna relación con el orden en el cual los resultados fueron generados.

5.2 Definición de variables

Cuando se efectúan cálculos extensos, muchas veces es conveniente dar nombre a los resultados intermedios. De igual modo que en las matemáticas tradicionales o en otros lenguajes de programación, es posible hacer esto introduciendo *variables* con un nombre específico.

Maxima

Esto inicializa el valor de la variable x con 6.

```
(%i1) x:6;
(%o1) 6
```

Maxima

Donde quiera que aparezca x, *Maxima* la reemplaza por su valor 6.

```
(%i2) x^3-25
(%o2) 191
```

Maxima

Esto asigna un nuevo valor para x.

```
(%i3) x:11+5
(%o3) 16
```

Maxima

pi es inicializada con el valor numérico de π con 20 dígitos de exactitud.

```
(%i4) fpprec:20$
(%i5) pi:%pi,bfloat;
(%o5) 3.1415926535897932385b0
```

Maxima

Aquí esta el valor de **sqrt**(**pi**).

```
(%i6) sqrt(pi)
(%o6) 1.7724538509055160273b0
```

x : *valor*	asigna un valor a la variable x
x : y : *valor*	asigna un valor a las variable x e y
`kill(`x`)`	quita cualquier valor asignado a x
`kill(`x`,`y`)`	quita cualquier valor asignado a x e y
`values`	muestra las variables a las que se les ha asignado un valor

Manipulación de variables.

Es muy importante recordar que los valores asignados a las variables son permanentes. Una vez que el usuario ha asignado un valor a una variable en particular, el valor será almacenado hasta que éste lo remueva explícitamente. El valor, claro está, desaparecerá si el usuario inicia una nueva sesión con *Maxima*.

Olvidarse de las definiciones hechas es la causa más común de errores al usar *Maxima*. Si el usuario pusiera `x:5`, *Maxima* asume que éste siempre quiere que `x` tenga el valor `5`, hasta o a menos que se le indique explícitamente otra cosa. Para evitar los errores, deben quitarse los valores definidos en cuanto se haya terminado de usarlos.

- Quite valores que asigne a las variables en cuanto termine de usarlos.

Un principio útil al usar *Maxima*.

Maxima

Inicializando el valor de la variable y con 9.

```
(%i7) y:9;
(%o7) 9
```

Maxima

Aquí se muestran todas las variables que tienen un valor asignado.

(%i8) `values;`

(%o8) $[x, \pi, y]$

Maxima

Sentencia para quitar el valor asignado a la variable x.

```
(%i9) kill(x);
(%o9) done
```

Maxima

Sentencia para quitar el valor asignado a todas las variables.

```
(%i10) kill(values);
(%o10) done
```

Las variables que el usuario define pueden tener cualquier nombre. No hay límite para la longitud de sus nombres. Un inconveniente, sin embargo, es que los nombres de las variables nunca pueden empezar con números. Por ejemplo, `x3` puede ser una variable, pero `3x` corresponde a una sintaxis incorrecta en *Maxima*.

Maxima

He aquí cuatro variables diferentes.

(%i11) `EstoEsUnaVariable:4/3;`

(%o11) $\frac{4}{3}$

(%i12) `Esto_Es_Una_Variable:5^3;`

(%o12) 125

(%i13) `estoesunavariable:8;`

(%o13) 8

(%i14) `esto_es_una_variable:sqrt(7);`

(%o14) $\sqrt{7}$

Maxima

Con `values` visualizamos las nuevas variables a las que se les ha asignado un valor.

(%i15) `values;`

(%o15) [EstoEsUnaVariable, Esto_Es_Una_Variable, estoesunavariable, esto_es_una_variable]

Maxima

Es posible realizar operaciones diversas con estas variables.

```
(%i16) (Esto_Es_Una_Variable^2+esto_es_una_variable*
       estoesunavariable)/EstoEsUnaVariable;
```

(%o16) $\dfrac{3(8\sqrt{7}+15625)}{4}$

Maxima

Con `kill(all)` también se quita el valor asignado a todas las variables.

```
(%i17) kill(all);
```

(%o17) *done*

Maxima

Esta vez `values` no encuantra ninguan variable con valor asignado.

```
(%i18) values;
```

(%o18) [ ]

5.3 Secuencia de operaciones

Al realizar cálculos con *Maxima*, usualmente se lo hace mediante una secuencia de pasos. Si el usuario desea puede realizar cada paso en una línea separada. A veces, sin embargo, es conveniente ingresar varios pasos en una misma línea. Es posible hacer esto simplemente separando cada una de las partes ingresadas con punto y coma (si quiere verse las salidas en pantalla) o con signo de dolar (si no quiere verse salida alguna).

$expr_1; expr_2; \ldots; expr_n;$	hace varias operaciones y da el resultado de todas ellas
$expr_1$ \$ $expr_2$ \$ ... \$ $expr_n$ \$	hace varias operaciones y no muestra ningún resultado
$expr_1$ \$ $expr_2$ \$ ... \$ $expr_n;$	hace varias operaciones y da el resultado de la última línea

Formas de hacer secuencias de operaciones en *Maxima*.

Maxima

Esto realiza tres operaciones en una misma línea y muestra todos los resultados.

```
(%i1) x:3; y:4; z:x+y;
(%o1) 3
(%o2) 4
(%o3) 7
```

Maxima

Esto realiza tres operaciones en una misma línea sin mostrar resultados.

```
(%i4) x:3$ y:4$ z:x+y$
```

Maxima

Esto realiza tres operaciones en una misma línea y muestra el resultado de la última operación.

```
(%i8) x:3$ y:4$ z:x+y;
(%o10) 7
```

Si el usuario finaliza su entrada con un signo de dolar, esto es interpretado por *Maxima* como si estuviera ingresando una secuencia de operaciones con un signo de dolar al final; así que tiene el efecto de hacer que *Maxima* calcule las operaciones especificadas, pero no muestre la salida.

$expr$ \$	realiza una operación, pero no muestra la salida

Inhibiendo la salida.

Maxima

Añadiendo un signo de dolar al final de la línea se le indica a *Maxima* que no muestre la salida.

```
(%i1) x:47+5$
```

Maxima

Usando % se puede visualizar la salida anterior.

```
(%i12) %
(%o12) 52
```

5.4 Impresión de expresiones sin evaluar

El operador comilla simple evita la evaluación. Aplicado a un símbolo, la comilla simple evita la evaluación del símbolo. Aplicado a la invocación de una función, la comilla simple evita la evaluación de la función invocada, aunque los argumentos de la función son evaluados (siempre y cuando la evaluación no se evite de otra manera). Aplicado a una expresión con paréntesis, la comilla simple evita la evaluación de todos los símbolos y llamadas a funciones que hubiesen en la expresión.

Maxima

Esto asigna valores a las variables a y b.

```
(%i1) a:45$ b:56$
```

'a	evita la evaluación del símbolo a
'$f(x)$	evita la evaluación de la función f, pero no de sus argumentos
'(*expr*)	evita la evaluación de todos los símbolos y llamadas a funciones que hayan en la expresión *expr*

Evitando la evaluación.

Maxima

El operador comilla simple (') aplicado a la variable a evita la evaluación de ésta.

```
(%i3) 'a
```
(%o3) a

Maxima

El operador ' aplicado a la función `integrate` evita la evaluación de ésta.

```
(%i4) 'integrate(x^3,x);
```
(%o4) $\int x^3 dx$

Maxima

Un uso interesante del operador '.

```
(%i5) 'integrate(x^3,x)=integrate(x^3,x);
```
(%o5) $\int x^3 dx = \frac{x^4}{4}$

Maxima

El operador ' no evita la evaluación de los argumentos de una función.

```
(%i6) 'integrate((2+3)*x^3,x);
```
(%o6) $5 \int x^3 dx$

Maxima

El operador ' aplicado a una expresión.

```
(%i7) '(2*sqrt(a)+b*integrate(x^3,x));
```
(%o7) $b \int x^3 dx + 2\sqrt{a}$

Maxima

Otro uso interesante del operador '.

```
(%i8) '(2*sqrt(a)+b*integrate(x^3,x))=(2*sqrt(a)+
      b*integrate(x^3,x));
```

(%o8) $b\int x^3 dx + 2\sqrt{a} = 14x^4 + 6\sqrt{5}$

CAPÍTULO 6

Cálculos algebraicos

6.1 Cálculo simbólico

Una de las características importantes de *Maxima* es que puede hacer cálculos simbólicos y numéricos. Esto significa que puede manejar fórmulas algebraicas así como números.

Maxima

He aquí un típico cálculo numérico.

```
(%i1) 4+36-1;
(%o1) 39
```

Maxima

Este es un cálculo simbólico.

```
(%i2) 7*x-3*x+6;
(%o2) 4 x + 6
```

Cálculo numérico	$4 + 36 - 1 \to 39$
Cálculo simbólico	$7x - 3x + 6 \to 4x + 6$

Cálculo simbólico y numérico.

Maxima

El usuario puede digitar cualquier expresión algebraica en *Maxima*.

```
(%i3) x^3+2*x-1;
```

(%o3) $x^3 + 2\,x - 1$

Maxima

Maxima realiza automáticamente simplificaciones algebraicas básicas. Aquí combina a x^2 y a $-4x^2$ para dar $-3x^2$.

```
(%i4) x^2+x-4*x^2;
```

(%o4) $x - 3\,x^2$

Es posible digitar cualquier expresión algebraica usando los operadores enumerados en la sección 4.1. No debe olvidarse ingresar el asterisco para el producto, por ejemplo: `x * y`, de lo contrario *Maxima* asumirá que se trata de una sola variable.

Maxima

Maxima reordena y combina términos usando las reglas estándares del álgebra.

```
(%i5) x*y+2*x^2*y+y^2*x^2-2*y*x;
```

(%o5) $x^2\,y^2 + 2\,x^2\,y - x\,y$

Maxima

He aquí otra expresión algebraica.

```
(%i6) (x+2*y+1)*(x-2)^2;
```

(%o6) $(x-2)^2\,(2\,y + x + 1)$

Maxima

La función **expand** amplía productos y potencias.

```
(%i7) expand(%);
```

(%o7) $2x^2y - 8xy + 8y + x^3 - 3x^2 + 4$

Maxima

`factor` hace lo inverso de `expand`.

```
(%i8) factor(%);
```
(%o8) $(x-2)^2\,(2y+x+1)$

Cuando se digita expresiones complicadas, es importante poner los paréntesis en los lugares correctos. Así, por ejemplo, debe dar la expresión x^{4y} en la forma `x^(4*y)`. Si no se colocan los paréntesis, se interpretará como x^4y.

Maxima

He aquí una fórmula más complicada, que requiere de varios paréntesis.

```
(%i9) sqrt(2)/9801*(4*n)!*(1103+26390*n)/(n!^4+1);
```
(%o9) $\frac{\sqrt{2}\,(26390\,n+1103)\,(4\,n)!}{9801\,(n!^4+1)}$

Cuando el usuario digita una expresión, *Maxima* aplica automáticamente su gran repertorio de reglas para transformar las expresiones. Estas reglas incluyen las reglas estándares del álgebra, tales como $x-x=0$, junto con reglas mucho más sofisticadas.

Maxima

Maxima utiliza reglas estándares del álgebra para sustituir $(\sqrt{x+1})^4$ por $(x+1)^2$.

```
(%i10) sqrt(x+1)^4;
```
(%o10) $(x+1)^2$

Maxima

Maxima no conoce ninguna regla para esta expresión, así que deja expresión en la forma original que usted le dio.

```
(%i11) log(cos(x)+1);
```
(%o11) $\log(\cos x+1)$

6.2 Valores para símbolos

Cuando *Maxima* transforma una expresión por ejemplo $\mathtt{x} + \mathtt{x}$ en $2\,\mathtt{x}$, está tratando la variable x en forma puramente simbólica o formal. En tales casos, x es un símbolo que puede representar cualquier expresión.

A menudo, sin embargo, se necesita sustituir un símbolo como x por un "valor" determinado. Algunas veces este valor será un número; aunque puede que sea una expresión.

Para sustituir el símbolo x, que aparece en la expresión $1 + 2\mathtt{x}$, con un valor determinado; el usuario puede utilizar la función `ev` o una sintaxis alternativa de la misma.

`ev(`*expr*, *x*=*valor*`)`	reemplaza *x* por *valor* en la expresión *exp*
`ev(`*expr*, *x*=*valor*, *y*=*valor*`)`	realiza varios reemplazos
expr, *x*=*valor*	reemplaza *x* por *valor* en la expresión *exp*
expr, *x*=*valor*, *y*=*valor*	realiza varios reemplazos

Sustitución de símbolos por valores en expresiones.

Maxima

Esto realiza la regla de sustitución $\mathtt{x} = 3$ en la expresión $1 + 2\mathtt{x}$.

```
(%i1) 1+2*x,x=3;
(%o1) 7
```

Maxima

También es posible sustituir x por cualquier expresión. Aquí cada ocurrencia de x es sustituida por $2 - \mathtt{y}$.

```
(%i2) 1+x+x^2,x=2-y;
```

(%o2) $-y + (2 - y)^2 + 3$

Maxima

Maxima trata las reglas de sustitución como cualquier otra expresión simbólica.

```
(%i3) x=y+3;
```

(%o3) $x = y + 3$

Maxima

Esto aplica la regla de sustitución última a la expresión $\mathtt{x}^2 - 9$.

```
(%i4) x^2-9, %;
```

(%o4) $(y + 3)^2 - 9$

Maxima

Es posible aplicar varias reglas de sustitución juntas.

```
(%i5) (x+y)*(x-y)^2,x=3,y=1-a;
```

(%o5) $(4 - a)\ (a + 2)^2$

La función `ev` o su sintaxis alternativa, permiten aplicar reglas de sustitución a una expresión particular. A veces, sin embargo, se querrá definir reglas de sustitución que se apliquen siempre. Por ejemplo, puede ser que se desee sustituir `x` por `3` siempre que aparezca `x`. Según lo discutido en la sección 5.2, puede hacerse esto asignando el valor `3` a `x`, usando `x : 3`. Una vez que se haya hecho la asignación `x : 3`, `x` siempre será sustituido por `3`, cuando aparezca.

Maxima

Esto asigna el valor de `3` a `x`.

```
(%i6) x:3;
```

(%o6) 3

Maxima

Ahora `x` será sustituido automáticamente por `3` dondequiera que aparezca.

```
(%i7) x^2-1;
```

(%o7) 8

Maxima

Esto asigna la expresión `1 + a` a `x`.

```
(%i8) x:a+1;
```

(%o8) $a+1$

Maxima

Ahora `x` es reemplazado por `a + 1`.

(%i9) `x^2-1;`

(%o9) $(a+1)^2-1$

Es posible definir como valor de un símbolo a cualquier expresión, no solamente a un número. Debe recordarse que una vez que se haya dado tal definición, ésta continuará siendo utilizada siempre que aparezca el símbolo, hasta que el usuario la cambie o quite explícitamente. Para la mayoría de usuarios, el olvidarse quitar valores que han asignado a los símbolos es la causa más común de errores al usar *Maxima*.

x:valor	define un valor para x que será utilizado siempre
`kill(`x`)`	remueve cualquier valor definido para x

Asignando valores a símbolos.

Maxima

El símbolo `x` todavía tiene el valor que le asignó arriba.

(%i10) `x+5-2*x;`

(%o10) $-2\,(a+1)+a+6$

Maxima

Esto quita el valor que asignó a `x`.

(%i11) `kill(x)`

(%o11) *done*

Maxima

Ahora x no tiene ningún valor definido, así que puede ser utilizado como variable puramente simbólica.

```
(%i12) x+5-2*x;
```

(%o12) $5-x$

Los lenguajes de programación tradicionales que no soportan el cálculo simbólico permiten que las variables sean utilizadas solamente como nombres para objetos, típicamente números, que se han asignado como valores para ellos. En *Maxima*, sin embargo, `x` se puede también tratar como variable puramente formal, a la cual se le puede aplicar varias reglas de transformación. Por supuesto, si el usuario da explícitamente una definición, por ejemplo `x : 3`, entonces `x` será sustituida siempre por `3`, y no sirve más como variable formal.

Debe recordarse que las definiciones explícitas por ejemplo `x : 3` tienen un efecto global. Por otra parte, un reemplazo tal como

$$expr, \mathtt{x} = \mathtt{3}$$

afecta solamente a la expresión específica *expr*.

Es posible mezclar siempre reemplazos con asignaciones. Con asignaciones, se puede dar nombres a las expresiones en las cuales se desea hacer reemplazos, o a las reglas que se desea utilizar para hacer los reemplazos.

Maxima

Esto asigna un valor al símbolo t.

```
(%i13) t:x^2+1;
```

(%o13) x^2+1

Maxima

Esto asigna un valor al símbolo x.

```
(%i14) t,x=2;
```

(%o14) 5

Maxima

Esto encuentra el valor de `t` para un valor diferente de `x`.

```
(%i15) t,x=5*a;
```
(%o15) $25a^2 + 1$

Maxima

Esto encuentra el valor de `t` cuando `x` es sustituido por `%pi`, y luego evalúa el resultado numéricamente.

```
(%i16) t,x=%pi,numer;
```
(%o16) 10.86960440108936

Maxima

No obstante, el símbolo `t` preserva la definición original.

```
(%i17) t;
```
(%o17) $x^2 + 1$

6.3 Transformación de expresiones algebraicas

A menudo hay muchas formas diferentes de escribir la misma expresión algebraica. Como un ejemplo, la expresión $(x + 1)^2$ puede ser escrita como $x^2 + 2x + 1$. *Maxima* proporciona una gran colección de funciones para hacer conversiones entre las diferentes formas de expresiones algebraicas.

Maxima

`expand` da la "forma expandida" de una expresión, con los productos y las potencias desarrolladas.

```
(%i1) expand((x+1)^2);
```
(%o1) $x^2 + 2\,x + 1$

`expand(`*expr*`)`	desarrolla productos y potencias, escribiendo el resultado como suma de términos
expr`,expand`	equivale a `expand(`*expr*`)`
`factor(`*expr*`)`	escribe *expr* como un producto de factores mínimos
expr`,factor`	equivale a `factor(`*expr*`)`

Dos funciones comunes para transformar expresiones algebraicas.

Maxima

`factor` recupera la forma original.

(%i2) `factor(%);`

(%o2) $(x+1)^2$

Maxima

Es fácil generar expresiones complicadas con `expand`.

(%i3) `(x+3*y+1)^4,expand;`

(%o3) $81\,y^4+108\,x\,y^3+108\,y^3+54\,x^2\,y^2+108\,x\,y^2+54\,y^2+12\,x^3\,y+36\,x^2\,y+36\,x\,y+12\,y+x^4+4\,x^3+6\,x^2+4\,x+1$

Maxima

`factor` a menudo le da expresiones más simples.

(%i4) `%,factor;`

(%o4) $(3\,y+x+1)^4$

Maxima

Hay algunos casos donde `factor` puede dar expresiones más complicadas.

(%i5) `x^8-1,factor;`

(%o5) $(x-1)\,(x+1)\,\left(x^2+1\right)\,\left(x^4+1\right)$

Maxima

En este caso, **expand** da la forma "más simple".

```
(%i6) %,expand;
```

(%o6) $x^8 - 1$

6.4 Simplificación de expresiones algebraicas

En muchas situaciones el usuario desea escribir una expresión algebraica en la forma más simple posible. Aunque sea difícil saber exactamente lo que se entiende por la "forma más simple", un procedimiento práctico que vale la pena seguir es analizar varias formas diferentes de una expresión, y elegir la de menor número de partes

`rat(`*expr*`)`	convierte *expr* al formato canónico racional
`ratsimp(`*expr*`)`	simplifica la expresión *expr* y todas sus subexpresiones, incluyendo los argumentos de funciones no racionales
expr`,ratsimp`	equivale a `ratsimp(`*expr*`)`
`fullratsimp(`*expr*`)`	aplica repetidamente `ratsimp` a una expresión, seguida de simplificaciones no racionales, hasta que no se obtienen más transformaciones; entonces devuelve el resultado
expr`,fullratsimp`	equivale a `fullratsimp(`*expr*`)`

Simplificación de expresiones algebraicas.

Se puede utilizar, a menudo, `ratsimp` para "mejorar" expresiones complicadas que se obtienen como resultado de cálculos.

Maxima

He aquí la integral de $\dfrac{1}{x^4 - 1}$.

```
(%i1) integrate(1/(x^4-1),x);
```

(%o1) $-\frac{\log(x+1)}{4} - \frac{\arctan x}{2} + \frac{\log(x-1)}{4}$

Maxima

Al derivar el último resultado debería volverse a la expresión original. En este caso, como es común, se obtiene una versión más complicada de la expresión original.

```
(%i2) diff(%,x);
```

(%o2) $-\frac{1}{2\,(x^2+1)} - \frac{1}{4\,(x+1)} + \frac{1}{4\,(x-1)}$

Maxima

`ratsimp` permite volver a la forma original de la expresión.

```
(%i3) ratsimp(%);
```

(%o3) $\frac{1}{x^4-1}$

Las expresiones pueden incluir funciones no racionales y los argumentos de tales funciones son también racionalmente simplificados.

Maxima

He aquí una expresión que incluye funciones no racionales cuyos argumentos admiten ser racionalmente simplificados.

```
(%i4) sin(x/(x^2+x))=exp((log(x)+1)^2-log(x)^2);
```

(%o4) $\operatorname{sen}\left(\frac{x}{x^2+x}\right) = \%e^{(\log x+1)^2-\log^2 x}$

Maxima

`ratsimp` simplifica los argumentos de tales funciones.

```
(%i5) %,ratsimp;
```

(%o5) $\operatorname{sen}\left(\frac{1}{x+1}\right) = \%e\, x^2$

Ante expresiones no racionales, una llamada a `ratsimp` puede no ser suficiente para conseguir un resultado simplificado. En ocasiones serán necesarias más de una llamada a `ratsimp`, que es lo que hace precisamente `fullratsimp`.

Maxima

Esto define la variable `expresion`.

```
(%i6) expresion:(x^(a/2)+1)^2*(x^(a/2)-1)^2/(x^a-1);
```

(%o6) $\frac{\left(x^{a/2}-1\right)^2\left(x^{a/2}+1\right)^2}{x^a-1}$

Maxima

En general, `rat` no simplifica otras funciones que no sean la suma, resta, multiplicación, división y exponenciación de exponente entero.

```
(%i7) rat(expresion);
```

(%o7) $\frac{\left(x^{a/2}\right)^4-2\left(x^{a/2}\right)^2+1}{x^a-1}$

Maxima

Con `ratsimp` se consigue una "mejor" simplificación.

```
(%i8) ratsimp(expresion);
```

(%o8) $\frac{x^{2\,a}-2\,x^a+1}{x^a-1}$

Maxima

Con `fullratsimp` se consigue simplificar la expresión al máximo.

```
(%i9) fullratsimp(expresion);
```

(%o9) x^a-1

6.5 Expresiones puestas en diferentes formas

Las expresiones algebraicas complicadas se pueden escribir generalmente en varias maneras. *Maxima* proporciona una variedad de funciones para convertir expresiones de una forma a otra. Los más comunes de estas funciones son `expand`, `factor` y `ratsimp`. Sin embargo, cuando se tiene expresiones racionales que contienen cocientes, puede ser necesario utilizar otras funciones.

`expandwrt(`*expr*,*var*$_1$,...,*var*$_n$`)`	expande la expresión *expr* con respecto a las variables *var*$_1$,...,*var*$_n$ y por defecto no expande los denominadores
`expand(`*expr*`)`	expande la expresión *expr*
`factor(`*expr*`)`	factoriza la expresión *expr*
`partfrac(`*expr*,*var*`)`	expande la expresión *expr* en fracciones parciales respecto de la variable principal *var*

Comandos para transformar expresiones algebraicas.

Maxima

He aquí una expresión racional, la cual puede ser escrita en varias formas diferentes.

```
(%i1) e:(x-1)^2*(2+x)/((1+x)*(x-3)^2);
```

(%o1) $\frac{(x-1)^2\,(x+2)}{(x-3)^2\,(x+1)}$

Maxima

expandwrt expande el numerador de la expresión, pero deja el denominador en forma factorizada.

```
(%i2) expandwrt(e,x);
```

(%o2) $\frac{x^3}{(x-3)^2\,(x+1)} - \frac{3\,x}{(x-3)^2\,(x+1)} + \frac{2}{(x-3)^2\,(x+1)}$

Maxima

expand expande todo, incluyendo el denominador.

```
(%i3) expand(e);
```

(%o3) $\frac{x^3}{x^3-5\,x^2+3\,x+9} - \frac{3\,x}{x^3-5\,x^2+3\,x+9} + \frac{2}{x^3-5\,x^2+3\,x+9}$

Maxima

`partfrac` separa la expresión en términos con denominadores simples.

(%i4) `partfrac(%,x);`

(%o4) $\frac{1}{4\,(x+1)}+\frac{19}{4\,(x-3)}+\frac{5}{(x-3)^2}+1$

Maxima

`factor` factoriza todo, en este caso reproduce la forma original.

(%i5) `factor(%);`

(%o5) $\frac{(x-1)^2\,(x+2)}{(x-3)^2\,(x+1)}$

`collectterms(`*expr*,*var*`)`	agrupa juntas todos las potencias de *var*

Reordenamiento de expresiones en varias variables.

Maxima

He aquí una expresión algebraica en dos variables.

(%i6) `v:expand((3+2*x)^2*(x+2*y)^2);`

(%o6) $16\,x^2\,y^2+48\,x\,y^2+36\,y^2+16\,x^3\,y+48\,x^2\,y+36\,x\,y+4\,x^4+12\,x^3+9\,x^2$

Maxima

Esto agrupa los términos de `v` afectados por la misma potencia de `x`.

(%i7) `collectterms(v,x);`

(%o7) $x\,\left(48\,y^2+36\,y\right)+x^2\,\left(16\,y^2+48\,y+9\right)+6\,y^2+x^3\,\left(16\,y+12\right)+4\,x^4$

Maxima

Esto agrupa juntas las potencias de `y`.

(%i8) `collectterms(v,y);`

(%o8) $\left(16\,x^2+48\,x+36\right)\,y^2+\left(16\,x^3+48\,x^2+36\,x\right)\,y+4\,x^4+12\,x^3+9\,x^2$

Como acaba de verse, cuando el usuario se limita a expresiones polinómicas racionales, hay muchas formas de escribir cualquier expresión particular. Si éste considera expresiones más complicadas, que incluyan, por ejemplo, funciones matemáticas trascendentes, la variedad de formas posibles llega a ser mayor. Por consiguiente, es imposible tener una función incorporada específico en *Maxima* para producir cada forma posible. Más bien, *Maxima* le permite construir sistemas arbitrarios de reglas de transformación para hacer diversas conversiones[1]. Sin embargo, hay algunas funciones incorporadas adicionales de *Maxima* para transformar expresiones.

`trigexpand(`*expr*`)`	expande funciones trigonométricas e hiperbólicas de sumas de ángulos y de múltiplos de ángulos presentes en la expresión *expr*
expr`,trigexpand`	equivale a `trigexpand(`*expr*`)`
`trigsimp(`*expr*`)`	utiliza las identidades $\operatorname{sen}(x)^2+\cos(x)^2=1$ y $\cosh(x)^2-\operatorname{senh}(x)^2=1$ para simplificar expresiones que contienen tan, sec, etc.
`trigreduce(`*expr,var*`)`	combina productos y potencias de senos y cosenos trigonométricos e hiperbólicos de *var*, transformándolos en otros que son múltiplos de *var*
`trigreduce(`*expr*`)`	si no se introduce el argumento *var*, entonces se utilizan todas las variables de *expr*
expr`,trigreduce`	equivale a `trigreduce(`*expr*`)`

Algunas funciones más para transformar expresiones.

[1]Para más detalle al respecto consulte sobre las funciones `scsimp` y `defrule` en la ayuda de *Maxima*.

`trigrat(`*expr*`)`	devuelve una forma canónica simplificada cuasi-lineal de una expresión trigonométrica
`exponentialize(`*expr*`)`	convierte las funciones trigonométricas e hiperbólicas de *expr* a exponenciales
expr`,exponentialize`	equivale a `exponentialize(`*expr*`)`
`demoivre(`*expr*`)`	convierte exponenciales complejos en expresiones equivalentes pero en términos de las funciones trigonométricas
expr`,demoivre`	equivale a `demoivre(`*expr*`)`
`rectform(`*expr*`)`	devuelve una expresión de la forma $a+b\,\%i$ equivalente a *expr*, con a y b reales
expr`,rectform`	equivale a `rectform(`*expr*`)`
`polarform(`*expr*`)`	devuelve una expresión de la forma $r\,\%e^{\%i\,\theta}$ equivalente a *expr*, con r y θ reales
expr`,polarform`	equivale a `polarform(`*expr*`)`
`radcan(`*expr*`)`	simplifica la expresión *expr*, que puede contener logaritmos, exponenciales y radicales, convirtiéndola a una forma canónica
expr`,radcan`	equivale a `radcan(`*expr*`)`
prod`,radexpand:all`	las raíces n-ésimas de los factores del producto *prod*, que sean potencias de n, se extraen del símbolo radical (p. ej., $\sqrt{4x^2} \to 2x$)
`logcontract(`*expr*`)`	analiza la expresión *expr* recursivamente, transformando subexpresiones de la forma `a1*log(b1)+a2*log(b2)+c` en expresiones de la forma `log(ratsimp(b1^a1*b2^a2))+c`
expr`,logcontract`	equivale a `logcontract(`*expr*`)`

Algunas funciones más para transformar expresiones.

Maxima

Esto expande la expresión trigonométrica, escribiéndola de modo que todas las funciones tengan argumento x.

```
(%i9) tan(x)*cos(3*x),trigexpand;
```

(%o9) $\left(\cos^3 x - 3\,\cos x\,\sin^2 x\right)\,\tan x$

Maxima

Esto reduce la expresión usando ángulos múltiples.

```
(%i10) tan(x)*cos(2*x),trigreduce;
```

(%o10) $\tan x\,\cos\left(3\,x\right)$

Maxima

Esto expande el seno asumiendo que `x` e `y` son reales.

```
(%i11) sin(x+%i*y),rectform;
```

(%o11) $\%i\,\cos x\,\sinh y + \sin x\,\cosh y$

Maxima

Con `logcontract` se "contrae" una expresión logarítimica.

```
(%i12) 2*(a*log(x)+2*a*log(y)),logcontract;
```

(%o12) $a\,\log\left(x^2\,y^4\right)$

Las transformaciones hechas por funciones como `expand` y `factor` siempre son correctas, independientemente del valor que puedan tener las variables simbólicas en las expresiones. A veces, sin embargo, es útil realizar transformaciones que sólo son correctas para algunos posibles valores de las variables simbólicas. Transformaciones como éstas las realizan `radcan` y `radexpand:all`.

Maxima

Maxima no expande automáticamente potencias no enteras de productos y cocientes.

```
(%i13) sqrt(x^5*y/w^3);
```

(%o13) $\sqrt{\frac{x^5\,y}{w^3}}$

Maxima

radcan hace la expansión.

```
(%i14) %,radcan;
```

(%o14) $\frac{x^{\frac{5}{2}}\,\sqrt{y}}{w^{\frac{3}{2}}}$

Maxima

En este caso *Maxima* aplica una equivalencia matemática.

```
(%i15) sqrt(x^6*y^2/w^10);
```

(%o15) $\frac{|x|^3\,|y|}{|w|^5}$

Maxima

Utilizando la variable opcional **radexpand** con el valor asignado **all**, *Maxima* pasa por alto la equivalencia anterior.

```
(%i16) sqrt(x^6*y^2/w^10),radexpand:all;
```

(%o16) $\frac{x^3\,y}{w^5}$

6.6 Simplificación con asunciones

$assume(pred_1, \ldots, pred_n)$	añade los predicados $pred_1, \ldots, pred_n$ al contexto actual
$forget(pred_1, \ldots, pred_n)$	borra los predicados establecidos por `assume`
$facts()$	devuelve una lista con los predicados asociados al contexto actual

Asunción de predicados.

Maxima

`assume` devuelve una lista con los predicados que han sido añadidos al contexto.

```
(%i1) assume(x>0,y<0);
```

(%o1) $[x > 0, y < 0]$

Maxima

Maxima realiza simplificaciones asumiendo los predicados ingresados.

```
(%i2) [sqrt(x^2),sqrt(y)];
```

(%o2) $[x, \sqrt{y}]$

Maxima

Otra simplificación asumiendo los predicados ingresados.

```
(%i3) sqrt(x^2*y^2);
```

(%o3) $-x\,y$

Maxima

`facts` muestra los predicados asociadas al contexto actual.

```
(%i4) facts();
```

(%o4) $[x > 0, y < 0]$

Maxima

`forget` borra los predicados previamente establecidos.

```
(%i5) forget(x>0,y<0);
```

(%o5) $[x > 0, y < 0]$

Maxima

Después de borrar los predicados con `forget`, la llamada `facts()` devuelve una lista vacía.

```
(%i6) facts();
```

(%o6) $[\,]$

6.7 Selección de partes de expresiones algebraicas

`coeff(`*expr*,*x*,*n*`)`	devuelve el coeficiente de x^n en *expr* (el argumento n puede omitirse si es igual a la unidad)
`hipow(`*expr*,*x*`)`	devuelve el mayor exponente explícito de x en *expr* (si x no aparece en *expr*, `hipow` devuelve 0)
`part(`*expr*,$n_1, \ldots, n_k$`)`	devuelve la parte de *expr* que se especifica por los índices $n_1, \ldots, n_k$ (primero se obtiene la parte n_1 de *expr*, después la parte n_2 del resultado anterior, y así sucesivamente)

Comandos para seleccionar partes de polinomios.

Maxima

He aquí una expresión algebraica.

```
(%i1) e:expand((1+3*x+4*y^2)^2);
```

(%o1) $16\,y^4 + 24\,x\,y^2 + 8\,y^2 + 9\,x^2 + 6\,x + 1$

Maxima

Esto da el coeficiente de `x` en `e`.

```
(%i2) coeff(e,x);
```

(%o2) $24\,y^2 + 6$

Maxima

`hipow(`*expr*`,y)` da la mayor potencia de `y` que aparece en *expr*.

```
(%i3) hipow(e,y);
```

(%o3) 4

Maxima

Esto da el cuarto término en `e`.

```
(%i4) part(e,4);
```

(%o4) $9\,x^2$

`num(`*expr*`)`	devuelve el numerador de *expr* (si *expr* no es una fracción, devuelve *expr*)
`denom(`*expr*`)`	devuelve el denominador de *expr* (si *expr* no es una fracción, devuelve 1)

Comandos para seleccionar partes de expresiones racionales.

Maxima

He aquí una expresión racional.

```
(%i5) r:(1+x)/(2*(2-y));
```

(%o5) $\frac{x+1}{2\,(2-y)}$

Maxima

denom selecciona el denominador.

```
(%i6) denom(r);
```

(%o6) $2\,(2-y)$

Maxima

denom da 1 para las expresiones que no son cocientes.

```
(%i7) denom(1/x+1/y);
```

(%o7) 1

CAPÍTULO 7

Matemáticas simbólicas

La capacidad de *Maxima* de tratar con expresiones simbólicas, así como numéricas, le permite usarlo para muchas áreas de la matemática, siendo la más común el cálculo.

7.1 Límites

limit(f,x,x_0)	el límite $\lim\limits_{x\to x_0} f$
limit(f,x,x_0,plus)	el límite $\lim\limits_{x\to x_0^+} f$
limit(f,x,x_0,minus)	el límite $\lim\limits_{x\to x_0^-} f$

Límites.

Maxima

He aquí la expresión $\dfrac{\operatorname{sen} x}{x}$.

```
(%i1) f:sin(x)/x;
```

(%o1) $\dfrac{\sin(x)}{x}$

Maxima

Si se sustituye x por 0, la expresión se hace $0/0$, y se obtiene un mensaje de error.

```
(%i2) f,x=0;
```

```
(%o2) Division by 0
      --an error. To debug this try: debugmode(true);
```

Maxima

Si se evalúa $\frac{\mathrm{sen}(x)}{x}$ para un x próximo a 0, se consigue un resultado próximo a 1.

```
(%i3) f,x=0.01;
(%o3) 0.99998333341667
```

Maxima

Esto encuentra el límite de $\frac{\mathrm{sen}(x)}{x}$ cuando x tiende a 0.

```
(%i4) limit(f,x,0);
(%o4) 1
```

`inf`	$+\infty$
`minf`	$-\infty$
`und`	indefinido
`ind`	indefinido pero acotado
`zeroa`	infinitesimal mayor que cero
`zerob`	infinitesimal menor que cero
`infinity`	infinito complejo

Símbolos especiales para límites.

La función `limit` con un solo argumento se utiliza frecuentemente para simplificar expresiones en las que aparecen los símbolos especiales para límites.

Maxima

Esto da un resultado para $1 - (-\infty)$.

```
(%i5) limit(1-minf);
(%o5) ∞
```

Maxima

He aquí la simplificación de una expresión que incluye un infinitesimal mayor que cero.

```
(%i6) limit(x+zeroa);
```
(%o6) x

7.2 Diferenciación

`diff(`f,x`)`	devuelve la primera derivada de f respecto de la variable x
`diff(`f,x,n`)`	devuelve la n-esima derivada de f respecto de x
`diff(`f,$x_1, n_1, \ldots, x_m, n_m$`)`	devuelve la derivada parcial de f con respecto de $x_1, \ldots, x_m$ y equivale a `diff(`...`(diff(`f,x_m, n_m ...`)`, x_1, n_1`)`
`diff(`f`)`	devuelve el diferencial total de f

Diferenciación con *Maxima*.

Maxima

He aquí la derivada x^n con respecto a x.

```
(%i1) diff(x^n,x);
```
(%o1) $n\,x^{n-1}$

Maxima

Maxima conoce las derivadas de todas las funciones matemáticas estándar.

```
(%i2) diff(atan(x),x);
```
(%o2) $\frac{1}{x^2+1}$

Maxima

La tercera derivada con respecto a x.

```
(%i3) diff(x^n,x,3);
```

(%o3) $(n-2)\,(n-1)\,n\,x^{n-3}$

Si no se indica la variable, *Maxima* asume que se quiere calcular la diferencial total. En notación matemática, `diff(`f,x`)` es como $\frac{d}{dx}f$, mientras `diff(`f`)` es como df.

Maxima

Esto da la diferencial total $d(x^n)$. `del`x y `del`n son los diferenciales dx y dn, respectivamente.

(%i4) `diff(x^n);`

(%o4) $n\,x^{n-1}\,\mathtt{del}\,(x)+x^n\,\log x\,\mathtt{del}\,(n)$

Así como se trata variables simbólicamente, también es posible tratar funciones simbólicamente en *Maxima*. Así, por ejemplo, puede encontrarse fórmulas para las derivadas de `f(x)`, sin especificar una forma explícita para la función `f`. Para esto hay que indicar a *Maxima* la dependencia de la función, lo que se consigue con la función `depends`.

Maxima

Esto declara la dependencia $f(x^2)$.

(%i5) `depends(f,x^2);`

(%o5) $\left[f\,\left(x^2\right)\right]$

Maxima

Ahora *Maxima* puede utilizar la regla de cadena para simplificar la derivada.

(%i6) `diff(2*x*f,x);`

(%o6) $4\,\left(\frac{d}{d\,x^2}\,f\right)\,x^2+2\,f$

`depends`(ϕ,φ)	declara la dependencia funcional $\phi(\varphi)$
`depends`$(\phi_1,\varphi_1,\ldots,\phi_n,\varphi_n)$	declara la dependencia funcional $\phi_1(\varphi_1),\ldots,\phi_n(\varphi_n)$
`depends`$([\phi_1,\ldots,\phi_n],\varphi)$	declara la dependencia funcional $\phi_1(\varphi),\ldots,\phi_n(\varphi)$
`depends`$(\phi,[\varphi_1\ldots,\varphi_n])$	declara la dependencia funcional $\phi(\varphi_1,\ldots,\varphi_n)$
`depends`$([\phi_1,\ldots,\phi_n],[\varphi_1,\ldots,\varphi_m])$	declara la dependencia funcional $\phi_1(\varphi_1,\ldots,\varphi_m),\ldots,\phi_n(\varphi_1,\ldots,\varphi_m)$
`dependencies`	lista de átomos que tienen algún tipo de dependencia funcional
`remove`$(\phi,$`dependency`$)$	elimina la dependencia funcional asociada con ϕ
`remove`$([\phi_1,\ldots,\phi_n],$`dependency`$)$	elimina la dependencia funcional asociada con $\phi_1,\ldots,\phi_n$
`remove(all,dependency)`	elimina la dependencia funcional de todos los átomos que la tengan

Declaración y remonición de dependencia funcional.

Maxima

Esto declara las dependencias $u(x)$ y $v(x)$.

```
(%i7) depends([u,v],x);
```

(%o7) $[u(x), v(x)]$

Maxima

Aquí se obtiene una fórmula para $\frac{d}{dx}(u^v)$, donde $u=u(x)$ y $v=v(x)$.

```
(%i8) diff(u^v,x),expand;
```

(%o8) $u^v \log u \left(\frac{d}{d\,x} v\right) + u^{v-1}\left(\frac{d}{d\,x} u\right) v$

Maxima

Esto permite apreciar todas las dependencias declaradas hasta el momento.

```
(%i9) dependencies;
```

(%o9) $\left[f\left(x^2\right), u\left(x\right), v\left(x\right)\right]$

Maxima

Con esta sentencia se borran todas las dependencias.

```
(%i10) remove(all,dependency);
(%o10) done
```

7.3 Integración

`integrate`(f,x)	la integral indefinida $\int f dx$
`integrate`(f,x,a,b)	la integral definida $\int_a^b f dx$
`integrate`(f=g,x)	la integral definida de una ecuación, equivale a $\int f dx = \int g dx$
`changevar('`*expr*,$\phi(x,y)$, y,x`)`	hace el cambio de variable dado por $\phi(x,y) = 0$ en la expresión *expr* que depende de x (la nueva variable será y)

Integración.

`integration_constant`	variable del sistema cuyo valor por defecto es `%c`
`integration_constant_counter`	variable del sistema cuyo valor por defecto es 0

Constantes y contadores de constantes para integrar ecuaciones.

Maxima

Para calcular la integral $\displaystyle\int x^n dx$, *Maxima*, pregunta si $n+1=0$ o $n+1 \neq 0$. En este caso se le ha indicado la elección de la opción $n+1=0$, es decir, $n \neq -1$.

```
(%i1) integrate(x^n,x);
Is n+1 zero or nonzero? n;
```

(%o1) $\frac{x^{n+1}}{n+1}$

Maxima

He aquí la integral $\int x^n dx$, cuando $n = -1$.

```
(%i2) integrate(x^n,x);
Is n+1 zero or nonzero? z;
```

(%o2) $\log(x)$

Maxima

Este es un ejemplo ligeramente más complicado.

```
(%i3) integrate(1/(x^4-a^4),x);
```

(%o3) $-\frac{\log(x+a)}{4\,a^3} + \frac{\log(x-a)}{4\,a^3} - \frac{\arctan\left(\frac{x}{a}\right)}{2\,a^3}$

Maxima

Recuérdese que `logcontract` "contrae" los logaritmos.

```
(%i4) integrate(1/(x^4-a^4),x),logcontract;
```

(%o4) $-\frac{\log\left(\frac{x+a}{x-a}\right)+2\,\arctan\left(\frac{x}{a}\right)}{4\,a^3}$

Maxima resuelve casi cualquier integral que puede ser expresada en términos de funciones matemáticas estándares. Pero debe comprenderse que aun cuando un integrando pueda contener sólo funciones simples, su integral puede implicar funciones mucho más complicadas, o puede no ser expresable en absoluto en términos de funciones matemáticas estándares.

Maxima

He aquí una integral simple.

```
(%i5) integrate(log(1-x^2),x),logcontract;
```

(%o5) $x\,\left(\log\left(1-x^2\right)-2\right)+\log\left(\frac{x+1}{x-1}\right)$

Maxima

Esta integral puede ser expresada sólo en términos de una función dilogarítmica[1].

```
(%i6) integrate(log(1-x^2)/x,x);
```

(%o6) $\log(x)\,\log\left(1-x^2\right)+\frac{li_2(1-x^2)}{2}$

Maxima

Esta integral involucra la función **erf**[2].

```
(%i7) integrate(exp(1-x^2),x);
```

(%o7) $\frac{\sqrt{\pi}\,e\,\mathrm{erf}(x)}{2}$

Maxima

Esta integral simplemente no puede ser expresada en términos de funciones matemáticas estándares. Por consiguiente, *Maxima* la deja como está.

```
(%i8) integrate(x^ x,x);
```

(%o8) $\int x^x\,dx$

Maxima

He aquí la integral definida $\int_a^b \operatorname{sen}^2(x)\,dx$.

```
(%i9) integrate(sin(x)^2,x,a,b);
```

(%o9) $\frac{\sin(2\,a)-2\,a}{4}-\frac{\sin(2\,b)-2\,b}{4}$

Maxima

He aquí otra integral definida.

```
(%i10) integrate(exp(-x^2),x,0,inf);
```

(%o10) $\frac{\sqrt{\pi}}{2}$

Maxima

Maxima no puede darle una fórmula para esta integral definida.

```
(%i11) integrate(x^x,x,0,1);
```

(%o11) $\int_0^1 x^x\,dx$

Maxima

Esto evalúa la integral múltiple $\int_0^1 \int_0^x (x^2 + y^2)\,dy\,dx$.

```
(%i12) integrate(integrate(x^2+y^2,y,0,x),x,0,1);
```

(%o12) $\frac{1}{3}$

Cuando una constante de integración se crea durante la integración definida de una ecuación, el nombre de la constante se construye concatenando las variables (del sistema) `integration_constant` e `integration_constant_counter`.

Maxima

Esto calcula la integral definida de una ecuación.

```
(%i13) integrate(x^2=sin(x),x);
```

(%o13) $\frac{x^3}{3} = \%c_1 - \cos(x)$

Maxima

A `integration_constant` se le puede asignar un símbolo cualquiera.

```
(%i14) integration_constant: 'K;
```

(%o14) K

Maxima

Esto calcula la integral definida de una ecuación a la que se le ha reiniciado la constante de integración por defecto.

```
(%i15) integrate(x^2=sin(x),x);
```

(%o15) $\frac{x^3}{3} = K_2 - \cos x$

Maxima

Es posible reiniciar el contador de `integration_constant_counter`.

```
(%i16) reset(integration_constant_counter);
```

(%o16) $[integration\_constant\_counter]$

Maxima

Aquí se aprecia que el contador ha sido reiniciado.

```
(%i17) integrate (x^2=sin(x),x);
```

(%o17) $\frac{x^3}{3} = K_1 - \cos x$

Es posible realizar un cambio de variable en una integral indefinida o definida usando la función `changevar`. Para que *Maxima* pueda realizar esto debe mantenerse la integral sin evaluar (sección 5.4).

Maxima

He aquí una integral clásica sin evaluar.

```
(%i18) 'integrate(%e^sqrt(y),y);
```

(%o18) $\int \%e^{\sqrt{y}}\, dy$

Maxima

Esto realiza un cambio de variable en una integral indefinida.

```
(%i19) assume(z>0)$ changevar(%o34,y-z^2,z,y);
```

(%o19) $\int \%e^{\sqrt{y}}\, dy$

Maxima

Esto realiza un cambio de variable en una integral definida.

```
(%i20) 'changevar(integrate(%e^sqrt(y),y,0,4),
       y-z^2,z,y);
```

(%o20) $-2\int_{-2}^{0} z\,\%e^{z}\,dz$

7.4 Sumas y Productos

`sum(`f`,`i`,`i_{min}`,`i_{max}`)`	la suma $\sum_{i_{min}}^{i_{max}} f$
`lsum(`f`,`i`,`L`)`	representa la suma de f para cada elemento i en L

Sumas.

Maxima

Esto construye la suma $\sum_{i=1}^{7} \frac{x^i}{i}$.

```
(%i1) sum(x^i/i,i,1,7);
```

(%o1) $\frac{x^7}{7}+\frac{x^6}{6}+\frac{x^5}{5}+\frac{x^4}{4}+\frac{x^3}{3}+\frac{x^2}{2}+x$

Maxima

Esto devuelve la suma $\sum_{i=1}^{n} i^2$ sin realizar ningún cambio.

```
(%i2) sum(i^2,i,1,n);
```

(%o2) $\sum_{i=1}^{n} i^2$

Maxima

Agregando **simpsum** la suma es calculada simbólicamente como una función de **n**.

```
(%i3) sum(i^2,i,1,n),simpsum;
```

(%o3) $\frac{2\,n^3+3\,n^2+n}{6}$

Maxima

Combinando **simpsum** con **factor** se obtiene un resultado factorizado.

```
(%i4) sum(i^2,i,1,n),simpsum,factor;
```

(%o4) $\frac{n\,(n+1)\,(2\,n+1)}{6}$

Maxima

Maxima también puede dar un resultado exacto para esta suma infinita.

```
(%i5) sum(1/i^4,i,1,inf),simpsum;
```

(%o5) $\frac{\pi^4}{90}$

Maxima

Esta es la suma múltiple $\sum_{i=1}^{3}\sum_{j=1}^{i} x^i y^j$.

```
(%i6) sum(sum(x^i*y^j,j,1,i),i,1,3);
```

(%o6) $x^3\,y^3+x^3\,y^2+x^2\,y^2+x^3\,y+x^2\,y+x\,y$

Maxima

Esta es una suma para la cual los valores del índice de variación no están equi-incrementados.

```
(%i7) lsum(x^i,i,[1,2,7]);
```

(%o7) x^7+x^2+x

product($f(i)$,i,i_{min}, i_{max}) el producto $\prod_{i_{min}}^{i_{max}} f(i)$

Productos.

Maxima

Los productos se obtienen en forma similar a las sumas.

```
(%i8) prod(x+i,i,1,4);
```

(%o8) $(x+1)\ (x+2)\ (x+3)\ (x+4)$

Maxima

Agregando `simpproduct` la suma es calculada simbólicamente como una función de `n`.

```
(%i9) product(k,k,1,n),simpproduct;
```

(%o9) $n!$

Maxima

Este es un producto que no puede ser resuelto.

```
(%i10) product(integrate(x^k,x,0,1),k,1,n);
```

(%o10) $$\prod_{k=1}^{n} \frac{1}{k+1}$$

Cabe mencionar que la función `changevar` también se puede utilizar para cambiar los índices de una suma o producto. Sin embargo, debe tenerse en cuenta que cuando se realiza un cambio en una suma o producto, el mismo debe expresarse en términos de sumas, como $i = j + \ldots$, no como una función de mayor grado.

Maxima

He aquí una suma finita con índice j.

```
(%i11) sum(f(j+2)*x^j,j,-2,n);
```

(%o11) $$\sum_{j=-2}^{n} f\,(j+2)\ x^j$$

Maxima

Esto realiza el cambio $j = i - 2$ en la suma anterior.

```
(%i12) changevar(%,j-i+2,i,j);
```

(%o12) $\sum_{i=0}^{n+2} f(i)\, x^{i-2}$

Maxima

Aquí se hace el cambio $i = k - 2$ en un producto infinito.

```
(%i13) product(f(i+2)*x^(i+2),i,-2,inf);
```

(%o13) $\prod_{i=-2}^{\infty} f(i+2)\, x^{i+2}$

```
(%i14) changevar(%,i-k+2,k,i);
```

(%o14) $\prod_{k=0}^{\infty} f(k)\, x^{k}$

7.5 Operadores relacionales y lógicos

$=$	igual (por sintaxis)
$\#$	desigual (por sintaxis)
$>$	mayor que
$>=$	mayor o igual que
$<$	menor que
$<=$	menor o igual que
`equal`	igual (por valor)
`notequal`	desigual (por valor)

Operadores relacionales.

`compare(`x,y`)`	compara x e y y devuelve $<$, $<=$, $>$, $>=$, $=$, $\#$, `notcomparable` ó `unknown`, según sea el caso

Función para obtener operadores relacionales.

Maxima

Esto prueba si 10 es menor que 7.

```
(%i1) 10<7,pred;
(%o1) false
```

Maxima

Esto prueba si 10 es diferente que 7.

```
(%i2) 10#7,pred;
(%o2) true
```

Maxima

Maxima puede determinar que esto es verdadero.

```
(%i3) %pi^%e<%e^%pi,pred;
(%o3) true
```

Maxima

Maxima no sabe si esto es verdadero o falso.

```
(%i4) x>y,pred;
(%o4) x > y
```

Maxima

He aquí dos comparaciones.

```
(%i5) compare(1,2);
(%o5) <
(%i6) compare(1/x,0);
(%o6) #
```

Maxima

Una diferencia notable entre = y `equal`.

```
(%i7) (x+1)^2=x^2+2*x+1,pred;
(%o7) false
```

```
(%i8) equal((x+1)^2,x^2+2*x+1),pred;
```
(%o8) *true*

El usuario debe tener presente que, los operadores relacionales son todos operadores binarios. *Maxima* no reconoce expresiones del estilo $a < b < c$.

Maxima

Al pretender evaluar una desigualdad como la siguiente, *Maxima* devuelve un mensaje de error.

```
(%i9) 2<3<4,pred;
```
(%o9) incorrect syntax: Found logical expression where algebraic expression expected
incorrect syntax: Premature termination of input at ;.

`not`	negación
`and`	conjunción
`or`	disyunción
`if` *cond* `then` $expr_1$ `else` $expr_2$	devuelve $expr_1$ si *cond* es `true`, y $expr_2$ si *cond* es `false`

Operadores lógicos.

Maxima

Una forma de ingresar expresiones del estilo $a < b < c$, en *Maxima*, es usando el operador lógico **and**.

```
(%i10) 2<3 and 3<4;
```
(%o10) *true*

Maxima

Mientras no se hayan hecho asignaciones adecuadas a p y q, *Maxima* no sabe si esto es verdadero o falso.

```
(%i11) p and q;
```

(%o11) $p \wedge q$

7.6 Ecuaciones

En *Maxima*, una ecuación consiste de dos expresiones vinculadas con el símbolo $=$. La expresión $a = b$ representa una ecuación sin evaluar, la cual puede verificarse o no.

$expr_1 = expr_2$	representa una ecuación
`lhs`$(expr_1 = expr_2)$	devuelve $expr_1$
`rhs`$(expr_1 = expr_2)$	devuelve $expr_2$

Ecuaciones en *Maxima*.

Maxima

Una ecuación por sí misma no es evaluada.

```
(%i1) x^4-5*x^2-3=x;
```

(%o1) $x = x^4 - 5\,x^2 - 3$

Es muy importante que el usuario no confunda `x:y` con `x=y` (ver sección 5.2). Mientras que `x:y` es una declaración imperativa que origina una asignación, `x=y` no causa ninguna acción explícita.

Maxima

Con `lhs`[3] se selecciona el miembro izquierdo de la ecuación.

```
(%i2) lhs(x=x^4-5*x^2-3);
```

(%o2) x

Maxima

Con `rhs` se selecciona el miembro derecho de la ecuación.

```
(%i3) rhs(x=x^4-5*x^2-3);
```

(%o3) $x^4 - 5\,x^2 - 3$

7.7 Solución de Ecuaciones Algebraicas

`solve(`*ecu*,*x*`)`	resuelve la ecuación algebraica *ecu* de incógnita x

Solución de ecuaciones.

`solve` siempre trata de dar fórmulas explícitas para las soluciones de ecuaciones. Sin embargo, para ecuaciones suficientemente complicadas, no pueden darse fórmulas algebraicas explícitas. Si se tiene una ecuación algebraica en una variable, y la potencia más alta de la variable es menor que cinco, entonces *Maxima* siempre puede dar fórmulas para las soluciones. Sin embargo, si la potencia más alta es cinco o más, puede ser matemáticamente imposible dar fórmulas algebraicas explícitas para todas las soluciones.

Maxima

Maxima siempre puede solucionar ecuaciones algebraicas en una variable cuando la potencia más alta es menor que cinco.

```
(%i1) solve(x^4-5*x^2-3=0,x);
```

(%o1) $$\left[x=-\frac{\sqrt{\sqrt{37}+5}}{\sqrt{2}}, x=\frac{\sqrt{\sqrt{37}+5}}{\sqrt{2}}, x=-\frac{\sqrt{\sqrt{37}-5}\,\%i}{\sqrt{2}}, x=\frac{\sqrt{\sqrt{37}-5}\,\%i}{\sqrt{2}}\right]$$

Maxima

También puede solucionar algunas ecuaciones que involucran potencias más altas.

```
(%i2) solve(x^6=1,x);
```

(%o2) $$\left[x=\frac{\sqrt{3}\,\%i+1}{2}, x=\frac{\sqrt{3}\,\%i-1}{2}, x=-1, x=-\frac{\sqrt{3}\,\%i+1}{2}, x=-\frac{\sqrt{3}\,\%i-1}{2}, x=1\right]$$

Maxima

Hay algunas ecuaciones, sin embargo, para las cuales es matemáticamente imposible encontrar fórmulas explícitas para las soluciones.

```
(%i3) solve(2-4*x+x^5=0,x);
```

(%o3) $[0 = x^5 - 4\,x + 2]$

También puede usarse *Maxima* para solucionar sistemas de ecuaciones simultáneas.

`solve`$([ecu_1, \ldots, ecu_n]\ ,\ [x_1, \ldots, x_n])$	resuelve un sistema de ecuaciones polinómicas simultáneas (lineales o no)

Solución de sistemas ecuaciones.

Maxima

He aquí una lista de dos ecuaciones simultáneas, para que sean resueltas en las variables x e y.

```
(%i4) solve([a*x+y=0,2*x+(1-a)*y=1],[x,y]);
```

(%o4) $$\left[\left[x = \frac{1}{a^2-a+2}, y = -\frac{a}{a^2-a+2}\right]\right]$$

Maxima

He aquí algunas ecuaciones simultáneas más complicadas.

```
(%i5) solve([x^2+x*y+y^2=4,x+x*y+y=2],[x,y]);
```

(%o5) $$\left[[x = 2, y = 0]\,, \left[x = -\frac{\sqrt{11}\,\%i+3}{2}, y = -\frac{\sqrt{11}\,\%i+7}{\sqrt{11}\,\%i+1}\right], \left[x = \frac{\sqrt{11}\,\%i-3}{2}, y = -\frac{\sqrt{11}\,\%i-7}{\sqrt{11}\,\%i-1}\right], [x = 0, y = 2]\right]$$

Maxima

Con `rectform` se obtiene una mejor presentación.

```
(%i6) %,rectform;
```

(%o6) $$\left[[x = 2, y = 0]\,, \left[x = -\frac{\sqrt{11}\,\%i}{2} - \frac{3}{2}, y = \frac{\sqrt{11}\,\%i}{2} - \frac{3}{2}\right], \left[x = \frac{\sqrt{11}\,\%i}{2} - \frac{3}{2}, y = -\frac{\sqrt{11}\,\%i}{2} - \frac{3}{2}\right], [x = 0, y = 2]\right]$$

Una variable para `solve` puede ser también una expresión. Esto se aprecia, por ejemplo, al aplicar esta función para obtener la derivada de una función definida en forma implícita.

Maxima

Esto asigna la ecuación $y^3 - 3\,x\,y + x^3 = 0$ a la variable e.

```
(%i7) e:x^3-3*x*y+y^3=0;
```

(%o7) $y^3 - 3\,x\,y + x^3 = 0$

Maxima

Aquí se define la dependencia $y(x)$.

```
(%i8) depends(y,x);
```

(%o8) $[y\,(x)]$

Maxima

Ahora se deriva e

```
(%i9) diff(e,x);
```

(%o9) $3\,y^2\,\left(\frac{d}{d\,x}\,y\right) - 3\,x\,\left(\frac{d}{d\,x}\,y\right) - 3\,y + 3\,x^2 = 0$

Maxima

Finalmente se resuelve la ecuación para la variable $\frac{d}{dx}\,y$.

```
(%i10) solve(%,diff(y,x));
```

(%o10) $\left[\frac{d}{d\,x}\,y = \frac{y-x^2}{y^2-x}\right]$

7.8 Solución de Ecuaciones Trascendentales

El paquete `to_poly_solver`[4] escrito por Barton Willis de la Universidad de Nebraska amplia la capacidad de *Maxima* para solucionar algunas ecuaciones que implican funciones trascendentales.

[4]El paquete `to_poly_solver` es experimental, siendo posible que las especificaciones de sus funciones puedan cambiar en el futuro, o que algunas de estas funciones puedan ser incorporadas a otras partes de *Maxima*.

`load(to_poly_solver)$`	inicializa el paquete `to_poly_solver`
`to_poly_solve`$(e, l,$ $[options])$	intenta resolver las ecuaciones e de incógnitas l

Comando para solucionar ecuaciones trascendentales.

`simpfuncs`	permite insertar funciones para transformar las soluciones (por ejemplo, **`expand`**, **`dfloat`**, **`nicedummies`**, etc. Incluso pueden ser funciones anónimas, definida por el usuario)
`parameters`	intenta encontrar una solución válida para parámetros especificados por el usuario
`use_grobner`	se aplica bases de grobner a las ecuaciones antes de intentar resolverlas

Algunas opciones de `to_poly_solver`.

Maxima

Inicialización del paquete `to_poly_solver`.

```
(%i1) load(to_poly_solver)$
```

Maxima

Aquí se soluciona una ecuación algebraica.

```
(%i2) to_poly_solve(2*x^2-3*x+5=0,x);
```

(%o2) $\%\text{union}\left([x=-\frac{\sqrt{31}\,i-3}{4}],[x=\frac{\sqrt{31}\,i+3}{4}]\right)$

Maxima

Esto muestra la solución de un sistema de ecuaciones lineales.

```
(%i3) to_poly_solve(
        [2*x-y=5,x+3*y=1],
        [x,y]);
```

(%o3) $\%\text{union}\left([x=\frac{16}{7},y=-\frac{3}{7}]\right)$

Maxima

Esto muestra la solución de un sistema de ecuaciones polinomiales.

```
(%i4) to_poly_solve(
        [(x^2+1)*(y^2+1)=10,(x+y)*(x*y-1)=3],
        [x,y]);
```

(%o4) $\%\mathrm{union}\left([x=-3,y=0],[x=-2,y=1],[x=0,y=-3],[x=1,y=-2],[x=1,y=2],[x=2,y=1]\right)$

Maxima

Este sistema de ecuaciones, en un inicio, no puede ser resuelto.

```
(%i5) to_poly_solve(
        [x^2+y^2=2^2,(x-1)^2+(y-1)^2=2^2],
        [x,y]);
```

(%o5) $\%\mathrm{union}\left(\right)$

Maxima

Al asignar el valor **true** a la opción **use_grobner**, se obtiene la solución.

```
(%i6) to_poly_solve(
        [x^2+y^2=2^2,(x-1)^2+(y-1)^2=2^2],
        [x,y],use_grobner=true);
```

(%o6) $\%\mathrm{union}\left(\left[x=-\frac{\sqrt{7}-1}{2},y=\frac{\sqrt{7}+1}{2}\right],\left[x=\frac{\sqrt{7}+1}{2},y=-\frac{\sqrt{7}-1}{2}\right]\right)$

Maxima

Esta solución no restringe el parámetro.

```
(%i7) to_poly_solve(a*x=x,x);
```

(%o7) $\%\mathrm{union}\left([x=0]\right)$

Maxima

Al indicar los parámetros con la opción **parameters** la solución es formal.

```
(%i8) to_poly_solve(a*x=x,x,parameters=[a]);
```

(%o8) $\%\mathrm{union}\left(\%\mathrm{if}\left(a-1=0,[x=\%c50],\%\mathrm{union}\left(\right)\right),\ \%\mathrm{if}\left(a-1\#0,[x=0],\%\mathrm{union}\left(\right)\right)\right)$

Maxima

Esto muestra la solución de la ecuación $\sqrt{x+\sqrt{x}}-\sqrt{x-\sqrt{x}}=\frac{3}{2}\sqrt{\frac{x}{x+\sqrt{x}}}$. Note que siendo $x=0$ el segundo miembro de la ecuación pierde el sentido (falla debida al algoritmo utilizado por el paquete).

```
(%i9) to_poly_solve(
        sqrt(x+sqrt(x))-sqrt(x-sqrt(x))=
        (3/2)*sqrt(x/(x+sqrt(x))),x);
```

(%o9) $\%\mathrm{union}\left([x=0],[x=\frac{25}{16}]\right)$

Maxima

Aquí se muestra la solución de la ecuación $\frac{3\log(\sqrt{x+1}+1)}{\log(x-40)}=3$, que involucra la función trascendental logaritmo.

```
(%i10) to_poly_solve(
        log(sqrt(x+1)+1)/log((x-40)^(1/3))=3,x);
```

(%o10) $\%\mathrm{union}\left([x=48],\ [x=36.02856987347005-5.91047671912819\,i],\ [x=5.910476719128112\,i+36.02856987347006]\right)$

Maxima

Para convertir cada solución a real de doble precisión hágase uso de **simpfunc=[dfloat]**.

```
(%i11) to_poly_solve(x^2+x+1=0,x,simpfuncs=[dfloat]);
```

(%o11) $\%\mathrm{union}\left([x=-0.86602540378444\,i-0.5],\ [x=0.86602540378444\,i-0.5]\right)$

Maxima

Solución de la ecuación $\frac{\cos(x)}{\sin(x)+1}+\frac{\sin(x)+1}{\cos(x)}=4$ que involucra las funciones trascendentales seno y coseno.

```
(%i12) to_poly_solve(
        cos(x)/(sin(x)+1)+(sin(x)+1)/cos(x)=4,x);
```

(%o12) $\%\mathrm{union}\left([x = 2\,\pi\,\%z60 - \frac{\pi}{3}], [x = 2\,\pi\,\%z62 + \frac{\pi}{3}]\right)$

Maxima

Aquí se sustituyen las constantes de la salida previa (vea la sección 6.2).

```
(%i13) %, %z60=k, %z62=n;
```

(%o13) $\%\mathrm{union}\left([x = 2\,\pi\,k - \frac{\pi}{3}], [x = 2\,\pi\,n + \frac{\pi}{3}]\right)$

Maxima

No obstante, la función **nicedummies** inicializa los índices (compare (%o14) con (%o12)).

```
(%i14) nicedummies(%o80);
```

(%o14) $\%\mathrm{union}\left([x = 2\,\pi\,\%z0 - \frac{\pi}{3}], [x = 2\,\pi\,\%z1 + \frac{\pi}{3}]\right)$

Maxima

He aquí la solución de la ecuación trascendental $\sin\left(x + \frac{\pi}{6}\right) = \cos\left(x - \frac{\pi}{4}\right)$.

```
(%i15) to_poly_solve(
         sin(x+%pi/6)=cos(x-%pi/4),x,
         simpfuncs=[expand]);
```

(%o15) $\%\mathrm{union}\left([x = \pi\,\%z65 + \frac{7\,\pi}{24}]\right)$

Maxima

En este caso se resuelve la ecuación trascendental $\tan x = \cot x$.

```
(%i16) to_poly_solve(tan(x)=cot(x),x,
         simpfuncs=[expand,nicedummies]);
```

(%o16) $\%\mathrm{union}\left([x = \frac{\pi\,\%z0}{2} + \frac{\pi}{4}]\right)$

Maxima

Esta es la solución de la ecuación trascendental $\sin\left(x^2 + 1\right) = \cos\left(x\right)$.

```
(%i17) to_poly_solve(sin(x^2+1)=cos(x),x,
         simpfuncs=[expand,nicedummies]);
```

(%o17) %union(
$$[x = \frac{1}{2} - \frac{\sqrt{-8\,\pi\,\%z0-2\,\pi-3}}{2}], [x = \frac{\sqrt{-8\,\pi\,\%z0-2\,\pi-3}}{2} + \frac{1}{2}],$$
$$[x = -\frac{\sqrt{8\,\pi\,\%z0+2\,\pi-3}}{2} - \frac{1}{2}], [x = \frac{\sqrt{8\,\pi\,\%z0+2\,\pi-3}}{2} - \frac{1}{2}])$$

Maxima

Esta es la solución de una ecuación exponencial.

```
(%i18) to_poly_solve(exp(x^2-1)-1=0,x,
         simpfuncs=[expand,nicedummies]);
```

(%o18) $\%\mathrm{union}\left([x = -\sqrt{2\,i\,\pi\,\%z0+1}], [x = \sqrt{2\,i\,\pi\,\%z0+1}]\right)$

Maxima

Esta es una ecuación que no puede ser resulta por `to_poly_solve`.

```
(%i19) to_poly_solve(exp(x^2-1)-x=0,x);
```

Unable to solve
Unable to solve
Unable to solve

(%o19) $\%\mathrm{solve}\left([e^{x^2-1} - x = 0], [x]\right)$

7.9 Sistemas de Inecuaciones Lineales

Además de la función `to_poly_solve` el paquete `to_poly_solver` incluye la función `fourier_elim` que permite resolver sistemas de inecuaciones lineales y algunas inecuaciones racionales.

`load(to_poly_solver)$`	inicializa el paquete `to_poly_solver`
`fourier_elim`$([ineq_1, ineq_2, \ldots], [var_1, var_2, \ldots])$	aplica el algoritmo de eliminación de Fourier para resolver el sistema de inecuaciones lineales $[ineq_1, ineq_2, \ldots]$ respecto de las variables $[var_1, var_2, \ldots]$

Comando para solucionar inecuaciones lineales.

Maxima

Esto permite resolver el sistema $\begin{cases} -2 < x < \pi, \\ e < x < 4\,. \end{cases}$ La solución aquí obtenida equivale a $e < x < \pi$.

```
(%i1) fourier_elim([-2<x,x< %pi and %e<x,x<4],[x]);
```

(%o1) $[e < x, x < \pi]$

Maxima

Eliminando primero respecto de x y luego respecto de y, se obtienen límites inferior y superior para x que dependen de y, y límites numéricos para y. Si se eliminan en orden inverso, se obtienen los límites de y en función de x, y los de x son números.

```
(%i2) fourier_elim([x+y<1,y>x and x>0],[x,y]);
```

(%o2) $[0 < x, x < \min(1-y, y), 0 < y, y < 1]$

```
(%i3) fourier_elim([x+y<1,y>x and x>0],[y,x]);
```

(%o3) $[x < y, y < 1-x, 0 < x, x < \frac{1}{2}]$

Maxima

Estas desigualdades no admiten solución.

```
(%i4) fourier_elim([x+y<1,y>x and x>1],[x,y]);
```

(%o4) *emptyset*

```
(%i5) fourier_elim([x+y<1,y>x and x>1],[y,x]);
```

(%o5) *emptyset*

Maxima

Aquí se resuelve la inecuación $\frac{(x-2)\,(x-1)^2\,(x+4)}{x+3} < 0.$

```
(%i6) fourier_elim([(x-1)^2*(x+4)*(x-2)/(x+3)<0],[x]);
```

(%o6) $[-3 < x, x < 1] \vee [1 < x, x < 2] \vee [x < -4]$

Maxima

Esto resuelve la inecuación $x - \frac{1}{x} \geq 0$

```
(%i7) fourier_elim([x-1/x>=0],[x]);
```

(%o7) $[x = -1] \vee [x = 1] \vee [1 < x] \vee [-1 < x, x < 0]$

Maxima

En este caso se resuelve la inecuación $\frac{|x+2|}{|x-1|} < 1$.

```
(%i8) fourier_elim([abs((x+2)/(x-1))<1],[x]);
```

(%o8) $[x < -2] \vee [x = -2] \vee [-2 < x, x < -\frac{1}{2}]$

Maxima

En caso de que uno de los factores no sea lineal, `fourier_elim` devuelve una lista de inecuaciones simplificadas.

```
(%i9) fourier_elim([(x^3-1)*(x+3)<0],[x]);
```

(%o9) $[-3 < x, x < 1, x^2 + x + 1 > 0]$
$\vee[1 < x, -\left(x^2 + x + 1\right) > 0]$
$\vee[x < -3, -\left(x^2 + x + 1\right) > 0]$

7.10 Inecuaciones racionales

Un paquete especializado en resolver inecuaciones racionales, escrito por Volker van Nek, es `solve_rat_ineq`.

`load(solve_rat_ineq)$`	inicializa el paquete `solve_rat_ineq`
`solve_rat_ineq(`*ineq*`)`	resuelve la inecuación racional *ineq*

Comando para solucionar inecuaciones racionales.

Maxima

Aquí se resuelve la inecuación $(x^3 - 1)(x + 3) < 0$.

```
(%i1) solve_rat_ineq((x^3-1)*(x+3)<0);
```

(%o1) $[[x > -3, x < 1]]$

Maxima

En este ejemplo se resuelve la inecuación $(y^3 - 1)(y + 3)^2 \geq 0$.

```
(%i2) solve_rat_ineq((y^3-1)*(y+3)^2>=0);
```

(%o2) $[[y = -3], [y >= 1]]$

Maxima

En este caso se resuelve la inecuación $\frac{(x-1)^2(x+4)(x-2)}{x+3} < 0$.

```
(%i3) solve_rat_ineq((x-1)^2*(x+4)*(x-2)/(x+3)<0);
```

(%o3) $[[x < -4], [x > -3, x < 1], [x > 1, x < 2]]$

Maxima

Aquí se resuelve la inecuación $x - 1 > \frac{1}{x}$.

```
(%i4) solve_rat_ineq(x-1>1/x);
```

(%o4) $[[x > -\frac{\sqrt{5}-1}{2}, x < 0], [x > \frac{\sqrt{5}+1}{2}]]$

7.11 Ecuaciones diferenciales ordinarias

`ode2(`*ecu,dvar,ivar*`)`	resuelve la ecuación diferencial ordinaria *ecu*, de variable dependiente *dvar* y variable independiente *ivar*, de primer y segundo orden
`ic1(`*sol*,$x = x_0, y = y_0$`)`	resuelve el problema del valor inicial en ecuaciones diferenciales ordinarias de primer orden
`ic2(`*sol*,$x = x_0, y = y_0$, `'`$diff(y, x) = dy_0$`)`	resuelve el problema del valor inicial en ecuaciones diferenciales ordinarias de segundo orden
`bc2(`*sol*,$x = x_0, y = y_0, x = x_1, y = y_1$`)`	resuelve el problema del valor en la frontera para ecuaciones diferenciales ordinarias de segundo orden

Solución de ecuaciones diferenciales ordinarias.

Maxima

He aquí la solución de la ecuación diferencial $y'(x) = ay(x) + 1$. En esta solución $\%c$ es una constante que debe ser determinada a partir de valores iniciales.

```
(%i1) ode2('diff(y,x)=a*y+1,y,x);
```

(%o1) $y = \left(\%c - \frac{e^{-a\,x}}{a}\right) e^{a\,x}$

Maxima

Si se incluye una condición inicial apropiada, entonces no hay ninguna constante en la solución.

```
(%i2) ic1(%,x=0,y=0);
```

(%o2) $y = \frac{e^{a\,x}-1}{a}$

Maxima

He aquí la solución de la ecuación diferencial $y''(x) + y\,y'(x) = 0$. En esta solución $\%k_1$ y $\%k_2$ son constantes a ser determinadas a partir de valores iniciales o valores de frontera.

```
(%i3) ode2('diff(y,x,2)+y*'diff(y,x)^3=0,y,x);
```

(%o3) $\frac{y^3+6\ \%k_1\,y}{6} = x + \%k_2$

Maxima

Si se incluyen las condiciones iniciales apropiadas desaparecen las constantes.

```
(%i4) ic2(%,x=0,y=0,'diff(y,x)=1);
```

(%o4) $\frac{y^3-3\,y\left(y^2-1\right)}{6} = x$

Maxima

Si se incluyen los valores de frontera apropiados también desaparecen las constantes.

```
(%i5) bc2(%o96,x=0,y=0,x=1,y=1);
```

(%o5) $\frac{y^3+5\,y}{6} = x$

7.12 Sistemas de ecuaciones diferenciales ordinarias lineales

La función `desolve` resuelve sistemas de ecuaciones diferenciales ordinarias lineales utilizando la transformada de Laplace. La dependencia funcional respecto de una variable independiente debe indicarse explícitamente, tanto en las variables como en las derivadas. Por ejemplo, para la primera derivada, a diferencia de la forma `'diff(y,x)` (usada en la sección 7.11), debe usarse la forma `'diff(y(x),x)`.

`desolve(`*ecu*,*y*(*x*)`)`	resuelve una ecuación diferencial para $y(x)$, tomando a x como variable independiente
`desolve(`$[ecu_1, \ldots, ecu_n]$, $[y_1(x), \ldots, y_n(x)]$`)`	resuelve sistemas de ecuaciones diferenciales ordinarias lineales
`atvalue(`$\phi(x), x = x_0, val$`)`	permite añadir la condición inicial $\phi(x_0) = val$ a un determinado sistema de ecuaciones diferenciales ordinarias lineales

Solución de sistemas de ecuaciones diferenciales ordinarias lineales.

Maxima

He aquí la solución de la ecuación diferencial $y''(x) + y(x) = 2x$.

```
(%i1) desolve(diff(y(x),x,2)+y(x)=2*x,y(x));
```

(%o1) $$y(x) = \sin(x)\left(\left.\frac{d}{d\,x}\,y(x)\right|_{x=0} - 2\right) + y(0)\cos(x) + 2\,x$$

Maxima

Esto resuelve el sistema $\begin{cases} x'(t) = y(t), \\ y'(t) = x(t)\,. \end{cases}$

```
(%i2) desolve([diff(x(t),t)=y(t),diff(y(t),t)=x(t)],
      [x(t),y(t)]);
```

(%o2) $$\left[x(t) = \frac{(y(0)+x(0))\,\%e^{t}}{2} - \frac{(y(0)-x(0))\,\%e^{-t}}{2},\; y(t) = \frac{(y(0)+x(0))\,\%e^{t}}{2} + \frac{(y(0)-x(0))\,\%e^{-t}}{2}\right]$$

Maxima

De esta manera se añaden las las condiciones iniciales $x(0) = 1$, $y(0) = 0$ al sistema anterior.

```
(%i3) atvalue(x(t),t=0,1);
(%o3) 1
(%i4) atvalue(y(t),t=0,0);
(%o4) 0
```

Maxima

Al volver a resolver el sistema se obtienen las soluciones con las condiciones iniciales dadas.

```
(%i5) desolve([diff(x(t),t)=y(t),diff(y(t),t)=x(t)],
       [x(t),y(t)]);
```

(%o5) $\left[x\left(t\right)=\frac{\%e^{t}}{2}+\frac{\%e^{-t}}{2}, y\left(t\right)=\frac{\%e^{t}}{2}-\frac{\%e^{-t}}{2}\right]$

Maxima

En este otro ejemplo se resuelve la ecuación diferencial $\frac{d^3y}{dx^3} - 3\frac{d^2y}{dx2} + 3\frac{dy}{dx} - y = e^x x^2$, con las condiciones iniciales $\begin{cases} y(0) & = 1 \\ y'(0) & = 0 \\ y''(0) & = -2 \end{cases}$.

```
(%i6) eq:diff(y(x),x,3)-3*diff(y(x),x,2)+3*diff(y(x),x)
       -y(x)=%e^x*x^2;
```

(%o6) $\frac{d^3}{d\,x^3}\,\mathrm{y}\left(x\right)-3\,\left(\frac{d^2}{d\,x^2}\,\mathrm{y}\left(x\right)\right)+3\,\left(\frac{d}{d\,x}\,\mathrm{y}\left(x\right)\right)-\mathrm{y}\left(x\right)=x^2\,e^x$

```
(%i7) atvalue(diff(y(x),x,2),x=0,-2)$
        atvalue(diff(y(x),x),x=0,0)$
        atvalue(y(x),x=0,1)$
(%i10) desolve([eq],[y(x)]);
```

(%o10) $\mathrm{y}\left(x\right)=\frac{x^5\,e^x}{60}-\frac{x^2\,e^x}{2}-x\,e^x+e^x$

7.13 Series de potencias

`taylor`$(expr, x, a, n)$	expande la expresión *expr* en un desarrollo de Taylor o de Laurent respecto de la variable x alrededor del punto a, con términos hasta $(x-a)^n$
`taylor`$(expr, [x_1, x_2, \ldots], a, n)$	devuelve la serie en potencias truncada de grado n en todas las variables $x_1, x_2, \ldots$ alrededor del punto $(a, a, \ldots)$
`taylor`$(expr, [x_1, a_1, n_1], [x_2, a_2, n_2], \ldots)$	devuelve la serie en potencias truncada en las variables $x_1, x_2, \ldots$ alrededor del punto $(a_1, a_2, \ldots)$; el truncamiento se realiza, respectivamente, en los grados $n_1, n_2, \ldots$
`taylor`$(expr, [x_1, x_2, \ldots], [a_1, a_2, \ldots], [n_1, n_2, \ldots], \ldots)$	equivale a `taylor`$(expr, [x_1, a_1, n_1], [x_2, a_2, n_2], \ldots)$
`taylor`$(expr, [x, a, n,$ `'asymp`$])$	desarrolla *expr* en potencias negativas de $x-a$ (el término de mayor orden es $(x-a)^{-n}$)

Obtención de series de potencias.

Las operaciones matemáticas de las que se ha hablado hasta ahora son exactas. Considerando la entrada exacta, sus resultados son fórmulas exactas. En muchas situaciones, sin embargo, no se necesita un resultado exacto. Puede ser suficiente, por ejemplo, encontrar una fórmula aproximada que es válida, digamos, cuando la cantidad x es pequeña.

Maxima

Esto da una aproximación en serie de potencias para alrededor de 0, hasta los términos de orden 3.

```
(%i1) taylor((1+x)^n,x,0,3);
```

(%o1) $1 + n\,x + \frac{(n^2-n)\,x^2}{2} + \frac{(n^3-3\,n^2+2\,n)\,x^3}{6} + \cdots$

Maxima

Maxima conoce las expansiones en serie de potencias para una gran cantidad funciones matemáticas.

```
(%i2) taylor(exp(-a*t)*(1+sin(2*t)),t,0,4);
```

(%o2) $$1+(-a+2)\,t+\frac{\left(a^2-4\,a\right)\,t^2}{2}-\frac{\left(a^3-6\,a^2+8\right)\,t^3}{6}+\frac{\left(a^4-8\,a^3+32\,a\right)\,t^4}{24}+\cdots$$

Maxima

Si se da una función no conocida, `taylor` escribe la serie de potencias en términos de derivadas.

```
(%i3) taylor(1+f(t),t,0,3);
```

(%o3) $$1+f\,(0)+\left(\frac{d}{d\,t}\,f\,(t)\Big|_{t=0}\right)\,t+\frac{\left(\frac{d^2}{d\,t^2}\,f(t)\Big|_{t=0}\right)\,t^2}{2}+\frac{\left(\frac{d^3}{d\,t^3}\,f(t)\Big|_{t=0}\right)\,t^3}{6}+\cdots$$

Las series de potencias son fórmulas aproximadas que juegan el mismo papel con respecto a las expresiones algebraicas como los números aproximados con las expresiones numéricas. *Maxima* permite realizar operaciones en series de potencias y en todos los casos mantiene el orden apropiado o el "grado de precisión" para las series de potencias resultantes.

Maxima

He aquí una serie de potencias simple, de orden 3.

```
(%i4) taylor(exp(x),x,0,3);
```

(%o4) $$1+x+\frac{x^2}{2}+\frac{x^3}{6}+\cdots$$

Maxima

Cuando se hacen operaciones en una serie de potencias, el resultado es calculado sólo en el orden apropiado en x.

```
(%i5) %^2*(1+%);
```

(%o5) $2+5\,x+\frac{13\,x^2}{2}+\frac{35\,x^3}{6}+\cdots$

Maxima

Al copiar los tres primeros términos del desarrollo anterior se tiene una expresión ordinaria.

```
(%i6) 2+5*x+(13*x^2)/2+(35*x^3)/6;
```

(%o6) $\frac{35\,x^3}{6}+\frac{13\,x^2}{2}+5\,x+2$

Maxima

Ahora el cuadrado es calculado exactamente.

```
(%i7) %^2;
```

(%o7) $\left(\frac{35\,x^3}{6}+\frac{13\,x^2}{2}+5\,x+2\right)^2$

Maxima

Al aplicar `expand` se obtiene un resultado con once términos.

```
(%i8) %,expand;
```

(%o8) $\frac{1225\,x^6}{36}+\frac{455\,x^5}{6}+\frac{1207\,x^4}{12}+\frac{265\,x^3}{3}+51\,x^2+20\,x+4$

Maxima

He aquí una serie de potencias doble de orden 3.

```
(%i9) taylor(sin(y+x),[x,0,3],[y,0,3]);
```

(%o9) $y-\frac{y^3}{6}+\cdots+\left(1-\frac{y^2}{2}+\cdots\right)\,x+\left(-\frac{y}{2}+\frac{y^3}{12}+\cdots\right)\,x^2+$
$\left(-\frac{1}{6}+\frac{y^2}{12}+\cdots\right)\,x^3+\cdots$

Un detalle interesante para mencionar es que *Maxima*, en los casos que sea posible, permite obtener la fórmula general del desarrollo en serie de potencias de una función.

`powerseries`$(expr, x, a)$	devuelve la forma general del desarrollo en serie de potencias de $expr$ para la variable x alrededor del punto a

Obtención de series de potencias.

Maxima

He aquí una conocida fórmula.

```
(%i10) powerseries(sin(x),x,0);
```

(%o10) $\sum_{i_1=0}^{\infty} \frac{(-1)^{i_1}\, x^{2\, i_1+1}}{(2\, i_1+1)!}$

Maxima

Esta es una fórmula más elaborada.

```
(%i11) powerseries(cos(x^2-1),x,0);
```

(%o11) $\sin(1)\, \sum_{i_2=0}^{\infty} \frac{(-1)^{i_2}\, x^{2\,(2\, i_2+1)}}{(2\, i_2+1)!} + \cos(1)\, \sum_{i_2=0}^{\infty} \frac{(-1)^{i_2}\, x^{4\, i_2}}{(2\, i_2)!}$

Maxima

Una forma de eliminar los *subíndices* de los *índices*, en las sumas, es usando la función `niceindices`.

```
(%i12) niceindices(powerseries(cos(x^2-1),x,0));
```

(%o12) $\sin(1)\, \sum_{i=0}^{\infty} \frac{(-1)^{i}\, x^{2\,(2\, i+1)}}{(2\, i+1)!} + \cos(1)\, \sum_{i=0}^{\infty} \frac{(-1)^{i}\, x^{4\, i}}{(2\, i)!}$

7.14 Transformada de Laplace

`laplace`$(expr, t, s)$	calcula la transformada de Laplace de $expr$ con respecto de la variable t y parámetro de transformación s
`ilt`$(expr, s, t)$	calcula la transformada inversa de Laplace de $expr$ con respecto de s y parámetro t.

Transformadas de Laplace.

Maxima

Esto calcula la transformada de Laplace.

```
(%i1) laplace(t^3*exp(a*t),t,s);
```

(%o1) $\frac{6}{(s-a)^4}$

Maxima

He aquí la transformada inversa.

```
(%i2) ilt(%,s,t);
```

(%o2) $t^3\,\%e^{a\,t}$

7.15 Ecuaciones recurrentes

El paquete `solve_rec` resuelve expresiones recurrentes lineales con coeficientes polinomiales.

Maxima

Inicialización del paquete `solve_rec`.

```
(%i1) load(solve_rec)$
```

Maxima

Esto resuelve una ecuación recurrente simple.

```
(%i2) solve_rec(a[n]=3*a[n-1]+1,a[n],a[1]=1);
```

(%o2) $a_n = \frac{3^n}{2} - \frac{1}{2}$

Maxima

He aquí una solución más complicada de otra ecuación recurrente.

```
(%i3) solve_rec(a[n+1]=(a[n]+1)/(n+1),a[n],a[1]=0);
```

(%o3) $a_n = \frac{\sum_{\%j=1}^{n-1} \frac{(\%j+1)!}{\%j+1}}{n!}$

Maxima

He aquí la solución de una ecuación recurrente con dos condiciones iniciales.

```
(%i4) solve_rec(
        a[n+2]=(3*a[n+1]-a[n])/2,
        a[n],a[1]=0,a[2]=1);
```

(%o4) $a_n = 2 - 2^{2-n}$

Maxima

Ejemplo de recurrencia lineal con coeficientes polinomiales.

```
(%i5) solve_rec(
        2*x*(x+1)*y[x]-(x^2+3*x-2)*y[x+1]+(x-1)*y[x+2],
        y[x],y[1]=1,y[3]=3);
```

(%o5) $y_x = 3\,2^{x-2} - \frac{x!}{2}$

Maxima

Cálculo de las soluciones racionales de una expresión recurrente lineal.

```
(%i6) solve_rec_rat(
        (x+4)*a[x+3]+(x+3)*a[x+2]-x*a[x+1]+
          (x^2-1)*a[x]=(x+2)/(x+1),
        a[x]);
```

(%o6) $a_x = \frac{1}{(x-1)\,(x+1)}$

CAPÍTULO 8

Matemáticas numéricas

8.1 Solución numérica de ecuaciones

algsys($[ecu_1, \ldots, ecu_m]$, $[x_1, \ldots, x_n]$)	resuelve el sistema de ecuaciones polinómicas $ecu_1, \ldots, ecu_m$ para las variables $x_1, \ldots, x_n$
allroots(ecu, x)	calcula aproximaciones numéricas de las raíces reales y complejas de la ecuación polinómica ecu para la variable x
find_root(ecu, x, a, b)	calcula una raíz de la ecuación ecu en el intervalo de aislamiento $[a, b]$

Búsqueda de raíces numéricas.

Maxima

solve devuelve la ecuación ingresada.

```
(%i1) solve(2-4*x+x^5=0,x);
```

(%o1) $\left[0 = x^5 - 4\,x + 2\right]$

Maxima

algsys proporciona una lista con soluciones aproximadas.

```
(%i2) algsys([2-4*x+x^5=0],[x]);
```

(%o2) $[[x = 1.243596445373759],$
$[x = -1.438447695329177\,\%i - 0.11679186122298],$
$[x = 1.438447695329177\,\%i - 0.11679186122298],$
$[x = -1.518512140520062], [x = 0.50849947534103]]$

Maxima

La opción `realonly:true` proporciona únicamente las aproximaciones reales.

```
(%i3) algsys([2-4*x+x^5=0],[x]),realonly:true;
```

(%o3) $[[x = 1.243596445373759], [x = 0.50849947534103],$
$[x = -1.518512140520062]]$

Si las ecuaciones involucran sólo funciones lineales o polinómicas, entonces puede usarse `algsys` para obtener aproximaciones numéricas de todas las soluciones. Sin embargo, cuando las ecuaciones involucran funciones más complicadas, no hay en general ningún procedimiento sistemático para obtener todas las soluciones, aún numéricamente. En tales casos, puede usarse `find_root` para buscar soluciones.

Téngase presente que `find_root` espera que la función en cuestión tenga signos diferentes en los extremos del intervalo de aislamiento.

Maxima

Aquí *Maxima* devuelve la misma sentencia de entrada, pues `find_root` evalúa los extremos del intervalo de aislamiento: $[0, 2]$; y la función log que, en este caso, forma parte de la ecuación no está definida en cero.

```
(%i4) find_root(3*cos(x)=log(x),x,0,2);
```

(%o4) $\text{find\_root}\,(3\cos(x) = \log(x), x, 0.0, 2.0)$

Maxima

Variando el extremo izquierdo del intervalo de aislamiento se obtiene una solución aproximada.

```
(%i5) find_root(3*cos(x)=log(x),x,0.00001,2);
```

(%o5) 1.447258617277903

Maxima

La ecuación tiene varias soluciones. Si se da un intervalo de aislamiento diferente, find_root puede devolver una solución diferente.

```
(%i6) find_root(3*cos(x)=log(x),x,12,15);
(%o6) 13.10638768062491
```

8.2 Integrales numéricas

quad_qags(f, x, a, b)	calcula numéricamente la integral $\int_a^b f\,dx$
quad_qagi(f, x, a, inf)	calcula numéricamente la integral $\int_a^\infty f\,dx$
quad_qagi$(f, x, minf, b)$	calcula numéricamente la integral $\int_{-\infty}^b f\,dx$
quad_qagi$(f, x, minf, inf)$	calcula numéricamente la integral $\int_{-\infty}^\infty f\,dx$

Integrales numéricas.

Las funciones quad_qags y quad_qagi devuelven una lista de cuatro elementos:

1. la aproximación a la integral,
2. el error absoluto estimado de la aproximación,
3. el número de evaluaciones del integrando,
4. un *código de error*.

El *código de error* puede tener los siguientes valores:

0 si no ha habido problemas;

1 si se utilizaron demasiados intervalos;

2 si se encontró un número excesivo de errores de redondeo;

3 si el integrando ha tenido un comportamiento extraño frente a la integración;

4 fallo de convergencia;

5 la integral es probablemente divergente o de convergencia lenta;

6 si los argumentos de entrada no son válidos.

Maxima

`quad_qags` puede manejar singularidades en los puntos finales de la región de integración.

```
(%i1) quad_qags(1/sqrt(x*(1-x)),x,0,1);
```

(%o1) $\left[3.141592653589849, 6.2063554295832546 \times 10^{-10}, 567, 0\right]$

Maxima

Para resolver integrales numéricas sobre regiones infinitas se usa `quad_qagi`.

```
(%i2) quad_qagi(exp(-x^2),x,minf,inf);
```

(%o2) $\left[1.772453850905516, 1.420263678183091 \times 10^{-8}, 270, 0\right]$

CAPÍTULO 9

Funciones y programas

9.1 Definición de funciones

Hasta aquí, se ha visto muchos ejemplos de funciones incorporadas en *Maxima*. En esta sección, se mostrará la forma en que el usuario puede añadir sus propias funciones a *Maxima*.

Maxima

Esto define la función `f`.

(%i1) `f(x):=x^2;`

(%o1) $f(x) := x^2$

Maxima

`f` eleva al cuadrado su argumento.

(%i2) `f(a+1);`

(%o2) $(a+1)^2$

Maxima

El argumento puede ser un número.

(%i3) `f(4);`

(%o3) 16

Maxima

O puede ser una expresión más complicada.

```
(%i4) f(x^2+3*x);
```

(%o4) $\left(x^2+3\,x\right)^2$

Maxima

Puede usarse `f` en un cálculo.

```
(%i5) expand(f(x+y+1));
```

(%o5) $y^2+2\,x\,y+2\,y+x^2+2\,x+1$

Maxima

Esto muestra la definición hecha para `f`.

```
(%i6) dispfun(f);
```

(%t6) $f(x) := x^2$

(%o6) $[\%t6]$

`f(x) := x^2`	define la función `f`
`dispfun(f)`	muestra la definición de `f`
`remfunction(f)`	borra todas las definiciones de `f`
`functions`	es una variable que contiene los nombres de las funciones definidas por el usuario

Definición de una función en *Maxima*.

Maxima

Las funciones en *Maxima* pueden tener cualquier número de argumentos

```
(%i7) hump(x,xmax):=(x-xmax)^2/xmax;
```

(%o7) $\mathrm{hump}\left(x, xmax\right) := \frac{(x-xmax)^2}{xmax}$

Maxima

Puede usarse la función **hump** tal como cualquiera de las funciones predefinidas.

```
(%i8) 2+hump(x,3.5);
```

(%o8) $0.28571428571429\,(x-3.5)^2+2$

Maxima

Esto da una nueva definición para **hump**, que sobrescribe la anterior.

```
(%i9) hump(x,xmax):=(x-xmax)^4;
```

(%o9) $hump(x, xmax) := (x - xmax)^4$

Maxima

Sólo es mostrada la nueva definición.

```
(%i10) dispfun(hump);
```

(%t10) $\mathrm{hump}\,(x, xmax) := \frac{(x-xmax)^2}{xmax}$

(%o10) $[\%t10]$

Maxima

Esto limpia todas las definiciones para **hump**.

```
(%i11) remfunction(hump);
```

(%o11) $[hump]$

`dispfun`$(f_1, \ldots, f_n)$	muestra las definiciones de f_i
`remfunction`$(f_1, \ldots, f_n)$	borra todas las definiciones de f_i
`dispfun`(all)	muestra las definiciones de todas las funciones definidas por el usuario
`remfunction`(all)	borra todas las definiciones de todas la funciones definidas por el usuario

Vista y borrado de varias funciones

Maxima

Ahora se define la función `g`.

(%i12) `g(x):=sqrt(1-x);`

(%o12) $g(x) := \sqrt{1-x}$

Maxima

Estro muestra la definición de todas las funciones (en este caso `f` y `g`).

(%i13) `dispfun(all);`

(%t13) $f(x) := x^2$

(%t13) $g(x) := \sqrt{1-x}$

(%o13) $[\%t13, \%t14]$

Maxima

Esto borra la definición de todas las funciones (en este caso `f` y `g`).

(%i14) `remfunction(all);`

(%o14) $[f, g]$

Maxima

Ya no hay funciones definidas por el usuario.

(%i15) `dispfun(all);`

(%o15) $[\ ]$

Cuando se ha terminado con una función particular, es bueno limpiar las definiciones que se haya hecho para ella. Si no, podría incurrirse en un conflicto al usar la misma función para un propósito diferente en la sesión de *Maxima*.

Maxima también permite definir funciones con una cantidad variable de argumentos (funciones de argumento variable) y funciones que se aplican directamente a listas (funciones *array*).

`F` $([L]) := \phi(L)$	define la función `F` cuyo número de argumentos es variable
`F` $[x_1, \ldots, x_n] := \phi(x_1, \ldots, x_n)$	define la función array `F` cuyo argumento es una lista
`F` $[x_1, \ldots, x_n] := [\phi_1(x_1, \ldots, x_n), \ldots, \phi_m(x_1, \ldots, x_n)]$	define la función array `F` cuyo argumento es una lista y que devuelve una lista

Más definiciones de funciones en *Maxima*.

Maxima

Aquí se define la función `F` que admite un número variable de argumentos.

```
(%i16) F([x]):=x^2+4;
```

(%o16) $F([x]) := x^2 + 4$

Maxima

Esto evalúa `F` en un solo argumento.

```
(%i17) F(2);
```

(%o17) 8

Maxima

Esto evalúa `F` en dos argumentos.

```
(%i18) F(2,3);
```

(%o18) $[8, 13]$

Maxima

He aquí la definición de la función array `G`.

```
(%i19) G[x,y]:=x^2+y^2;
```

(%o19) $G_{x,y} := x^2 + y^2$

Maxima

Aquí se evalúa `G` en la lista $[2, 3]$

```
(%i20) G[2,3];
```

(%o20) 13

Maxima

Esto define la función array H que devuelve una lista.

```
(%i21) H[x,y]:=[2*x,3-y,x*y];
```

(%o21) $H_{x,y} := [2 * x, 3 - y, x * y]$

Maxima

Aquí se evalúa la función H en la lista $[4, 3]$.

```
(%i22) H[4,3];
```

(%o22) $[8, 0, 12]$

Para visualizar la definición de este tipo de funciones puede usarse la función `dispfun`, pero para borrar las respectivas definiciones debe usarse la función `remarray`.

`remarray`(F)	borra la función array F
`remarray`$(F_1, \ldots, F_n)$	borra las funciones array $F_1, \ldots, F_n$
`remarray`(all)	borra todas las funciones array definidas por el usuario

Borrado de funciones array

Maxima

Esto la definición de todas las funciones array, hasta ahora, definidas.

```
(%i23) dispfun(all);
```

(%t23) $F([x]) := x^2 + 4$

(%t23) $G_{x,y} := x^2 + y^2$

(%t23) $H_{x,y} := [2\, x, 3 - y, x\, y]$

(%o23) $[\%t24, \%t25, \%t26]$

Maxima

Esto borra la definición de la función F definida en (%i17) .

```
(%i24) remfunction(F);
```

(%o24) $[F]$

Maxima

Esto borra la definición de todas las funciones array definidas por el usuario.

```
(%i25) remarray(all);
```

(%o25) $[G, H]$

Además, *Maxima* permite definir las llamadas funciones *anónimas* que son de mucha utilidad en contextos vinculados con la programación. La función para definirlas es *lambda*.

`lambda`$([x], expr)$	define una función anónima
`lambda`$([x_1, \ldots, x_m], expr_1, \ldots, expr_n)$	define una función anónima de argumentos múltiples

Función anónima

Maxima

Una función anónima puede asignarse a una variable y ser evaluada como si fuese una función ordinaria.

```
(%i26) f:lambda([x], x^2);
```

(%o26) $lambda([x], x^2)$

```
(%i27) f(a);
```

(%o27) a^2

Maxima

O puede aplicarse directamente.

```
(%i28) lambda([x], x^2)(a);
```

(%o28) a^2

Maxima

No obstante, no es reconocida por `dispfun`.

```
(%i29) dispfun(all);
```

(%o29) [ ]

En muchas clases de cálculos, el usuario puede encontrarse digitando la misma entrada a *Maxima* muchas veces. Es posible ahorrarse mucha digitación definiendo una función que contiene todas las sentencias de entrada.

Maxima

Esto construye un producto de tres términos, y expande el resultado.

(%i30) `expand(product(x+i,i,1,3));`

(%o30) $x^3 + 6\,x^2 + 11\,x + 6$

Maxima

Esto hace lo mismo, pero con cuatro términos.

(%i31) `expand(product(x+i,i,1,4));`

(%o31) $x^4 + 10\,x^3 + 35\,x^2 + 50\,x + 24$

Maxima

Esto define una función `exprod` que construye un producto de n términos, luego lo expande.

(%i32) `exprod(n):=expand(product(x+i,i,1,n)) $`

Maxima

Siempre que usa la función, ejecutará las operaciones `product` y `expand`.

(%i33) `exprod(5);`

(%o33) $x^5 + 15\,x^4 + 85\,x^3 + 225\,x^2 + 274\,x + 120$

Las funciones que se definen en *Maxima* son esencialmente procedimientos que ejecutan las sentencias dadas por el usuario. Es posible incluir varios pasos en los procedimientos, separados por comas.

Maxima

El resultado que se obtiene de la función es simplemente la última expresión en el procedimiento. Note que debe ponerse paréntesis alrededor del procedimiento cuando se define.

```
(%i34) cex(n,i):=(
         t:exprod(n),coeff(t,x^i)
       )$
```

Maxima

Esto "corre" el procedimiento.

```
(%i35) cex(5,3);
(%o35) 85
```

$(expr_1, expr_2, \ldots)$	una secuencia de expresiones para evaluar (procedimiento)
`block`$([a, b, \ldots], proc)$	un procedimiento con variables locales $a, b, \ldots$

Construcción de procedimientos.

Una buena idea es declarar, como locales, las variables que se usan dentro de los procedimientos, de modo que no interfieran con cálculos fuera de éstos. Puede hacerse esto estableciendo los procedimientos como *bloques*, en los cuales se da una lista de variables para que sean tratadas como locales.

Maxima

La función `cex` definida en `(%i37)` no es un bloque, así que el valor de `t` "escapa", y existe incluso después de la evaluación de la función.

```
(%i36) t;
```
(%o36) $x^5 + 15\,x^4 + 85\,x^3 + 225\,x^2 + 274\,x + 120$

Maxima

Esta función es definida como un bloque con variable local u.

```
(%i37) ncex(n,i):=block([u],
         u:exprod(n),
         coeff(u,x^i) ) $
```

Maxima

La función devuelve el mismo resultado que la anteriormente definida.

```
(%i38) ncex(5,3);
(%o38) 85
```

Maxima

Ahora, sin embargo, el valor de u no se escapa de la función.

```
(%i39) u
(%o39) u
```

Es posible asignar un valor inicial a una o más variables locales dentro de la lista de las mismas. También se puede definir una función como bloque sin declarar variables locales, para lo cual debe omitirse dicha lista.

Un bloque puede aparecer dentro de otro bloque. Las variables locales se inicializan cada vez que se entra dentro de un nuevo bloque. Las variables locales de un bloque se consideran globales dentro de otro anidado dentro del primero. Si una variable es no local dentro de un bloque, su valor es el que le corresponde en el bloque superior. Este criterio se conoce con el nombre de “alcance dinámico”.

9.2 Reglas de transformación para funciones

Maxima permite al usuario definir sus propias reglas de transformación y patrones de comparación.

Maxima

Definición de la regla de transformación que reemplaza x por 3.

```
(%i1) defrule(regtran1,x,3);
```

(%o1) $regtran1 : x \to 3$

Maxima

Aplicación de la regla de transformación `regtran1`

```
(%i2) apply1(1+f(x)+f(y),regtran1);
```

(%o2) $f(y) + f(3) + 1$

Maxima

Puede definirse una regla de transformación para `f(x)`. Ésta no afecta a `f(y)`.

```
(%i3) defrule(regtran2,f(x),p);
```

$regtran2 : \mathrm{f}(x) \to p$

```
(%i4) apply1(1+f(x)+f(y),regtran2);
```

(%o4) $f(y) + p + 1$

Maxima

Esto define un patrón `f(t)` que admite cualquier argumento para f.

```
(%i5) matchdeclare(t,true);
```

$done$

```
(%i6) defrule(patron,f(t),t^2);
```

(%o6) $patron : \mathrm{f}(t) \to t^2$

Maxima

Aquí se aplica el patrón previamente definido.

```
(%i7) apply1(1+f(x)+f(y),patron);
```

(%o7) $y^2 + x^2 + 1$

Maxima

Esto muestra todas las reglas, hasta aquí, definidas.

```
(%i8) disprule (all);
```

(%t8) $regtran_1 : x \to 3$

(%t8) $regtran_2 : f(x) \rightarrow p$
(%t8) $patron : f(t) \rightarrow t^2$
(%o8) $[\%t51, \%t52, \%t53]$

Maxima

Con esta sentencia se borran todas las definiciones de las reglas **regtran1**, **regtran2** y **patron**.

```
(%i9) clear_rules();
```
(%o9) $false$

Maxima

Esto indica que todas las definiciones de las reglas han sido borradas.

```
(%i10) disprule (all);
```
(%o10) [ ]

Probablemente, el aspecto más potente de las reglas de transformación en *Maxima* es que ellas pueden involucrar expresiones no sólo literales, sino también patrones. Un patrón es una expresión genérica `f(t)` para la cual se ha declarado el tipo de variable `t` con `matchdeclare`. Así, una regla de transformación para `f(t)` especifica cómo debería ser transformada la función `f` con el tipo de argumento especificado. Nótese que, en contraste, una regla de transformación para `f(x)` sin realizar la declaración de la variable, especifica sólo la transformación de la expresión literal `f(x)`, y por ejemplo, no dice nada sobre la transformación de `f(y)`.

Siempre que se cumpla con la declaración especificada, es posible establecer reglas de transformación para expresiones de cualquier forma.

Maxima

Esto define una función de predicado que define el patrón de producto para un argumento.

```
(%i11) prodp(expr):= not atom(expr) and op(expr)=“*”;
```
(%o11) $prodp(expr) := \neg\, atom(expr)\ \wedge\ op(expr) = “*”$

```
(%i12) matchdeclare(p,prodp);
```
(%o12) *done*

Maxima

Ahora se define una regla que transforme una función aplicada a un producto como la suma de los factores a los cuales se les ha aplicado la función.

```
(%i13) defrule( reg,f(p),apply( "+", map(f,args(p)) )
       );
```
(%o13) $reg : \mathrm{f}\,(p) \rightarrow \mathrm{apply}\,(+, \mathrm{map}\,(f, \mathrm{args}\,(p)))$

Maxima

Luego, al aplicar la regla recién definida se aprecia el efecto producido.

```
(%i14) apply1(f(a*b)+f(c*d),reg);
```
(%o14) $f\,(d) + f\,(c) + f\,(b) + f\,(a)$

Maxima

Esto aplica la regla sin limitar el número de los factores en el argumento de `f`.

```
(%i15) apply1(f(a)+f(a*b*c)+f(c),reg);
```
(%o15) $2\,f\,(c) + f\,(b) + 2\,f\,(a)$

Maxima

Esto combina la regla definida por el usuario con la función `expand` de *Maxima*.

```
(%i16) apply1(f(a)*f(a*b*c)*f(c),reg),expand;
```
(%o16) $f\,(a)\;f^2\,(c) + f\,(a)\;f\,(b)\;f\,(c) + f^2\,(a)\;f\,(c)$

9.3 Funciones definidas a partir de expresiones

En muchos casos, sobre todo en programación, resulta bastante útil poder definir una función a partir de una expresión dada para luego poder invocarla con facilidad y así evitar el uso reiterado de la función **ev** (ver sección 6.2).

define$(f(x_1,\ldots,x_n),$ $expr)$	define una función de nombre f con argumentos $x_1,\ldots,x_n$ y cuerpo $expr$
define$(f[x_1,\ldots,x_n],$ $expr)$	define una función array de nombre f con argumentos $x_1,\ldots,x_n$ y cuerpo $expr$

Definición de funciones a partir de expresiones.

Maxima

Esto almacena la expresión x^2+1 en la variable ex.

```
(%i1) ex:x^2+1;
```

(%o1) x^2+1

Maxima

Aquí se pretende definir la función f a partir de ex. No obstante, al realizar una evaluación no se obtiene el resultado esperado.

```
(%i2) f(x):=ex $
(%i3) f(8);
```

(%o3) x^2+1

Maxima

Por otra parte, al utilizar **define** para definir la función g a partir de ex si se obtiene un resultado satisfactorio.

```
(%i4) define(g(x),ex) $
(%i5) g(8);
```

(%o5) 65

Una gran utilidad de **define** se aprecia al definir funciones a partir de las derivadas de otras, las cuales serán evaluadas en valores numéricos.

Maxima

He aquí la definición de la función $f = x^2 + 1$.

```
(%i6) f(x):=x^2+1 $
```

Maxima

Ahora, a partir de f, se define "f'" como "f'" $= \frac{df}{dx}$.

```
(%i7) "f'"(x):=diff(f(x),x) $
```

Maxima

La evaluación simbólica de "f'" no presenta problema alguno.

```
(%i8) "f'"(t);
```

(%o8) $2t$

Maxima

No obstante, la evaluación numérica no esta permitida para dicha función.

```
(%i9) "f'"(8);
```

(%o9) diff: second argument must be a variable; found 8
#0: f'(x=8)
-- an error. To debug this try: debugmode(true);

Maxima

Aquí se vuelve a definir "f'", pero en este caso se utiliza la función `define` (recuerde que esta nueva definición anula a la anterior).

```
(%i10) define("f'"(x),diff(f(x),x)) $
```

Maxima

La evaluación simbólica es idéntica.

```
(%i11) "f'"(t);
```

(%o11) $2t$

Maxima

Sin embargo, ya no hay dificultad en la evaluación numérica.

```
(%i12) ''f'''(8);
(%o12) 16
```

9.4 Funciones definidas a trozos

Maxima no cuenta con una función específica para manipular adecuadamente las funciones definidas a trozos. Una forma, bastante limitada, de superar esta carencia es utilizando el condicional `if`.

Maxima

Esto define la función $f(x) = \begin{cases} x^2, & x < 2\,, \\ \sqrt{x}, & x \geq 2\,. \end{cases}$

```
(%i1) f(x):=block([],
        if (x<2) then return(x^2),
        if (x>=2) then return(sqrt(x))
      )$
```

Maxima

Aquí se hacen dos evaluaciones numéricas de la función.

```
(%i2) f(-2);
(%o2) 4
(%i3) f(9);
(%o3) 3
```

Maxima

No obstante, una evaluación simbólica deja mucho que desear.

(%i4) `f(t);`

(%o4) if $t >= 2$ then $return\ \left(\sqrt{t}\right)$

Maxima

Y ni que decir de un intento de derivar o integrar.

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

(%o5) $\frac{d}{d\,x}\left(\text{if } x >= 2 \text{ then } return\ \left(\sqrt{x}\right)\right)$

```
(%i6) integrate(f(x),x);
```

(%o6) $\int \text{if } x >= 2 \text{ then } return\ \left(\sqrt{x}\right)\ dx$

Maxima

Definamos, ahora, la función $g(x) = \begin{cases} x^2, & -2 < x < 2\,, \\ \sqrt{x}, & 2 \leq x < 3\,, \\ 1 - x, & 3 \leq x < 5. \end{cases}$

```
(%i7) g(x):=block([],
        if (-2<x and x<2) then return(x^2),
        if (2<=x and x<3) then return(sqrt(x)),
        if (3<=x and x<5) then return(1-x) )$
```

Maxima

El álgebra de funciones tampoco puede efectuarse.

```
(%i8) f(x)+g(x);
```

(%o8) $\left(\text{if } x >= 2 \text{ then } return\ \left(\sqrt{x}\right)\right) + \left(\text{if } 3 <= x\ and\ x < 5 \text{ then } return\ (1 - x)\right)$

Por otra parte, se tiene el paquete `pw` elaborado por Richard Hennessy[1], el cual está en proceso de desarrollo, que puede resultar útil en ciertos casos.

Maxima

Esto carga el paquete `pw`.

```
(%i9) load(pw)$
```

[1] El paquete `pw` (cuya versión en la actualidad es 6.4) se puede descargar desde `http://sourceforge.net/projects/piecewisefunc/`

Maxima

Esto define la función $f(x) = \begin{cases} x^2, & x < 2\,, \\ \sqrt{x}, & x \geq 2\,. \end{cases}$

```
(%i10) f(x):=piecewise(
          [minf,x^2,2,sqrt(x),2,sqrt(x),inf],
          x,open)$
```

Maxima

Aquí se hacen tres evaluaciones numéricas de la función.

```
(%i11) f(-2);
```

(%o11) 4

```
(%i12) f(2);
```

(%o12) $\sqrt{2}$

```
(%i13) f(9);
```

(%o13) 3

Maxima

Una evaluación simbólica nos deja ver las funciones utilizadas por el autor para manipular las funciones definidas a trozos.

```
(%i14) f(t);
```

(%o14) $$\frac{\sqrt{t}\,(\mathrm{signum}(t-2)-\mathrm{unit\_spike}(t-2)+1)}{2}+ \frac{t^2\,(-\mathrm{signum}(t-2)-\mathrm{unit\_spike}(t-2)+1)}{2}+\sqrt{t}\,\mathrm{unit\_spike}\,(t-2)$$

Maxima

Con la función **pwsimp** se obtiene una presentación matricial.

```
(%i15) pwsimp(f(x),x,array);
```

$$\begin{pmatrix} If & x & in & (& -\infty & , & 2 &) & then & x^2 \\ If & x & in & [& 2 & , & 2 &] & then & \sqrt{2} \\ If & x & in & (& 2 & , & \infty &) & then & \sqrt{x} \end{pmatrix}$$

(%o15) $Done$

Maxima

No obstante, la derivación no es correcta. Note que, de acuerdo con la definición, no existe $f'(2)$; sin embargo, aquí se obtiene $\frac{1}{2^{\frac{3}{2}}}$.

```
(%i16) pwsimp( diff(f(x),x), x,array);
```

$$\begin{pmatrix} If & x & in & (& -\infty & , & 2 &) & then & 2\,x \\ If & x & in & [& 2 & , & 2 &] & then & \frac{1}{2^{\frac{3}{2}}} \\ If & x & in & (& 2 & , & \infty &) & then & \frac{1}{2\,\sqrt{x}} \end{pmatrix}$$

(%o16) *Done*

Maxima

He aquí integración.

```
(%i17) pwsimp(pwint(f(x),x),x,array),expand;
```

$$\begin{pmatrix} If & x & in & (& -\infty & , & 2 &) & then & \frac{x^3}{3}+\frac{2^{\frac{3}{2}}}{3}-\frac{4}{3} \\ If & x & in & [& 2 & , & 2 &] & then & \frac{2^{\frac{3}{2}}}{3}+\frac{4}{3} \\ If & x & in & (& 2 & , & \infty &) & then & \frac{2\,x^{\frac{3}{2}}}{3}-\frac{2^{\frac{3}{2}}}{3}+\frac{4}{3} \end{pmatrix}$$

(%o17) *Done*

Maxima

Definamos, ahora, la función $g(x) = \begin{cases} x^2, & -2 < x < 2\,, \\ \sqrt{x}, & 2 \leq x < 3\,, \\ 1-x, & 3 \leq x < 5. \end{cases}$

```
(%i18) g(x):=piecewise(
         [-2,x^2,2,sqrt(x),2,sqrt(x),3,1-x,3,1-x,5],
         x,open)$
```

Maxima

Observe esta operación.

```
(%i19) f(3)+g(3);
```

(%o19) $\sqrt{3}-2$

Maxima

No obstante, al calcular $f(x)+g(x)$ se aprecia que $f(3)+g(3) = -\frac{4-2\sqrt{3}}{2}$.

```
(%i20) pwsimp(f(x)+g(x),x,array);
```

$$\begin{pmatrix} If & x & in & (& -\infty & , & -2 &) & then & x^2 \\ If & x & in & [& -2 & , & -2 &] & then & 4 \\ If & x & in & (& -2 & , & 2 &) & then & 2\,x^2 \\ If & x & in & [& 2 & , & 2 &] & then & 2^{\frac{3}{2}} \\ If & x & in & (& 2 & , & 3 &) & then & 2\sqrt{x} \\ If & x & in & [& 3 & , & 3 &] & then & -\frac{4-2\sqrt{3}}{2} \\ If & x & in & (& 3 & , & 5 &) & then & -x+\sqrt{x}+1 \\ If & x & in & [& 5 & , & 5 &] & then & \sqrt{5} \\ If & x & in & (& 5 & , & \infty &) & then & \sqrt{x} \end{pmatrix}$$

(%o20) *Done*

Es preciso anotar que el objetivo no ha sido criticar negativamente el paquete `pw`, sino, mostrar al lector la existencia de un trabajo que en un futuro cercano puede ser una buena alternativa para la carencia que adolece *Maxima* en el tema de las funciones definidas a trozos.

CAPÍTULO 10

Listas

10.1 Juntar objetos

Las listas son bloques de construcción básica para *Maxima* y *Lisp*[1]. Todos los tipos de datos diferentes a los arreglos, tablas mixtas o números son representados como listas en *Lisp*. Puede decirse que, en un nivel básico, lo que esencialmente hace una lista en *Maxima* es proporcionar una forma de agrupar varias expresiones de cualquier clase.

Maxima

He aquí una lista de números.

```
(%i1) [2,3,4];
```

(%o1) $[2, 3, 4]$

Maxima

Esto da una lista de expresiones simbólicas.

```
(%i2) x^%-1;
```

(%o2) $\left[x^2 - 1, x^3 - 1, x^4 - 1\right]$

[1]Lisp (*List Processing*) es el lenguaje de programación de alto nivel en el que está programado *Maxima*.

Maxima

Es posible derivar estas expresiones.

(%i3) diff(%,x);

(%o3) $[2\,x, 3\,x^2, 4\,x^3]$

Maxima

Y luego encontrar los valores cuando x es reemplazado por 3.

(%i4) %,x=3;

(%o4) $[6, 27, 108]$

La mayoría de funciones matemáticas incorporadas en *Maxima* han sido implementadas "listables" de modo que actúen separadamente sobre cada elemento de una lista.

10.2 Generación de listas

Maxima permite crear listas de acuerdo a una cierta regla definida por el usuario. Para ello cuenta con la función incorporada `create_list`.

`create_list`(f, i, i_{min}, i_{max})	da una lista de valores con i variando de i_{min} a i_{max}
`create_list`$(f, i, list)$	da una lista de valores con i variando de acuerdo a cada elemento de $list$
`create_list`$(f, i, i_{min}, i_{max}, j, j_{min}, j_{max}, \ldots)$	genera una lista con cada índice $(i, j, \ldots)$ variando del valor mínimo al valor máximo que le han sido asociados
`create_list`$(f, i, list_1, j, list_2, \ldots)$	genera una lista con cada índice $(i, j, \ldots)$ variando de acuerdo a cada elemento de la lista que se le ha asociado

Funciones para generar listas.

Maxima

Esto da una lista de los valores de i^2, con i variando de 1 a 6.

```
(%i1) create_list(i^2,i,1,6);
```

(%o1) $[1, 4, 9, 16, 25, 36]$

Maxima

He aquí una lista de los valores de sen $\left(\frac{n}{5}\right)$ para n variando de 0 a 4.

```
(%i2) create_list(sin(n/5),n,0,4);
```

(%o2) $\left[0, \sin\left(\frac{1}{5}\right), \sin\left(\frac{2}{5}\right), \sin\left(\frac{3}{5}\right), \sin\left(\frac{4}{5}\right)\right]$

Maxima

Esto da los valores numéricos de la tabla generada en (%i6) .

```
(%i3) %,numer;
```

(%o3) $[0, 0.19866933079506, 0.38941834230865, 0.56464247339504, 0.71735609089952]$

Maxima

También puede hacerse tablas de fórmulas.

```
(%i4) create_list(x^i+2*i,i,1,5);
```

(%o4) $\left[x+2, x^2+4, x^3+6, x^4+8, x^5+10\right]$

Maxima

`create_list` usa exactamente la misma notación para el iterador que las funciones `sum` y `product` que fueron mencionadas en la sección 7.4.

```
(%i5) product(x^i+2*i,i,1,5);
```

(%o5) $(x+2)\left(x^2+4\right)\left(x^3+6\right)\left(x^4+8\right)\left(x^5+10\right)$

Maxima

Esto hace una lista con valores de x variando desde 0 hasta 1 en pasos de 0.25

```
(%i6) create_list(0.25*i,i,1,4)$
```

```
(%i7) create_list(sqrt(x),x,%);
```
(%o7) $[0.5, 0.70710678118655, 0.86602540378444, 1.0]$

Maxima

Es posible realizar otras operaciones en las listas que se obtienen con `create_list`.

```
(%i8) %^2+3;
```
(%o8) $[3.25, 3.5, 3.75, 4]$

Maxima

Esto hace una lista de $x^i + y^j$ con i variando desde 1 hasta 3 y j variando desde 1 hasta 2.

```
(%i9) create_list(x^i+y^j,i,1,3,j,1,2);
```
(%o9) $\left[y+x, y^2+x, y+x^2, y^2+x^2, y+x^3, y^2+x^3\right]$

Maxima

Esto crea una lista conteniendo cuatro copias del símbolo x.

```
(%i10) create_list(x,i,1,4);
```
(%o10) $[x, x, x, x]$

Maxima

Esto da una lista de cuatro números seudo aleatorios.

```
(%i11) create_list(random(1.0),i,1,4);
```
(%o11) $[0.25223887668538, 0.41556272272148,$
$0.61257388925382, 0.84181265533592]$

10.3 Elección de elementos de una lista

`part`($list, i$) ó $list\,[\,i\,]$	el i-ésimo elemento de $list$ (el primer elemento es $list\,[\,1\,]$)
`part`($list, [\,i, j, \ldots\,]$)	una lista formada por los elementos $i, j, \ldots$ de $list$
`part`($list, i_1, \ldots, i_k$)	obtiene la parte de $list$ que se especifica por los índices $i_1, \ldots, i_k$ (primero se obtiene la parte i_1 de $list$, después la parte i_2 del resultado anterior, y así sucesivamente)
`first`($list$), `second`($list$), ..., `tenth`($list$)	devuelve el primer, segundo, ..., décimo elemento de $list$
`last`($list$)	devuelve el último elemento de $list$

Operaciones con elementos de una lista.

Maxima

Esto extrae el tercer elemento de la lista.

```
(%i1) part([4,-1,8,-6],3);
```

(%o1) 8

Maxima

Esto extrae una lista de elementos.

```
(%i2) part([4,-1,8,-6],[2,3,1,2,3,4,1]);
```

(%o2) $[-1, 8, 4, -1, 8, -6, 4]$

Maxima

Esto asigna una lista en la variable `u`.

```
(%i3) u:[7,2,4,6];
```

(%o3) $[7, 2, 4, 6]$

Maxima

Puede extraerse elementos de `u`.

```
(%i4) u[3];
```
(%o4) 4

Maxima

Pueden realizarse operaciones diversas con `u`.

```
(%i5) (u+1)/(u-6);
```
(%o5) $\left[-\frac{3}{4}, -\frac{1}{6}, -6, 8\right]$

Es posible crear listas multidimensionales mediante la anidación de listas en listas.

Maxima

Esto crea una lista 2×2, y le da el nombre `m`.

```
(%i6) m:create_list(create_list(i-j,j,1,2),
        i,1,2);
```
(%o6) $[[0, -1], [1, 0]]$

Maxima

Esto extrae la primera sublista de la lista de listas, `m`

```
(%i7) m[1];
```
(%o7) $[0, -1]$

Maxima

Esto extrae el segundo elemento de aquella sublista.

```
(%i8) %[2];
```
(%o8) -1

Maxima

Esto hace las dos operaciones juntas.

```
(%i9) part(m,1,2);
```

(%o9) -1

Maxima

Esto genera una lista tridimensional de $2 \times 2 \times 2$. Es una lista de listas de listas.

```
(%i10) create_list(create_list(create_list(i*j^2*k^3,
         k,1,2),j,1,2),i,1,2);
```

(%o10) $[[[1,8],[4,32]],[[2,16],[8,64]]]$

Maxima

Si no se anida **create_list** se obtiene una lista unidimensional.

```
(%i11) create_list(i*j^2*k^3,i,1,2,j,1,2,k,1,2);
```

(%o11) $[1,8,4,32,2,16,8,64]$

Al asignar una lista a una variable, puede usarse las listas en *Maxima* de modo similar a los "arrays" en otros lenguajes de programación. De esta manera, por ejemplo, es posible resetear un elemento de una lista asignando un valor a u[i].

part(u, i) ó u[i]	extrae el i-ésimo elemento de la lista u
u[i] : *valor*	resetea el i-ésimo elemento de la lista u

Operando listas como arrays.

Maxima

Aquí hay una lista.

```
(%i12) u:[2,1,5,7];
```

(%o12) $[2,1,5,7]$

Maxima

Esto resetea el segundo elemento de la lista.

```
(%i13) u[2]:0;
```

(%o13) 0

Maxima

Ahora la lista asignada a `u` se ha modificado.

```
(%i14) u;
```

(%o14) $[2,0,5,7]$

10.4 Prueba y búsqueda de elementos de una lista

`sublist`(L,P)	devuelve la lista de elementos x de la lista L para los cuales el predicado $P(x)$ es verdadero
`sublist_indices`(L,P)	devuelve los índices de los elementos x de la lista L para la cual el predicado $P(x)$ es verdadero
`?position`$(elem,L)$	devuelve el índice que indica la posición que ocupa $elem$ en la lista L
`member`$(form,list)$	prueba si $form$ es un elemento de $list$
`freeof`$(form,list)$	prueba si $form$ no ocurre en ninguna parte de $list$
`freeof`$(form_1,\ldots,$ $form_n,list)$	equivale a `freeof`$(form_1,list)$ `and` ... `and` `freeof`$(form_n,list)$

Funciones para la prueba y búsqueda de elementos de una lista.

Maxima

Esto da una lista de todos los elementos de $[a,1,3,b,7]$ que son de tipo simbólico.

```
(%i1) sublist([a,1,3,b,7],symbolp);
```

(%o1) $[a,b]$

Maxima

Esto da una lista de todas posiciones, de la lista $[a,1,3,b,7]$, en las que se encuentran elementos de tipo simbólico.

```
(%i2) sublist_indices([a,1,3,b,7],symbolp);
```

(%o2) $[1,4]$

Maxima

En este caso se emplea una función anónima para definir un predicado, el cual indica que se quiere obtener una lista de todos los elementos, de la lista $[1,2,4,7,6,2]$, que son mayores que 2.

```
(%i3) sublist([1,2,4,7,6,2],lambda([h],h>2));
```
(%o3) $[4,7,6]$

Maxima

Esto muestra que 4 es un elemento de $[11,1,2,4,5,5]$.

```
(%i4) member(4,[11,1,2,4,5,5]);
```
(%o4) $true$

Maxima

Por otra parte 3, no lo es.

```
(%i5) member(3,[11,1,2,4,5,5]);
```
(%o5) $false$

Maxima

Esto muestra que 0 ocurre en alguna parte de $[[1,2],[3,0]]$.

```
(%i6) freeof(0,[[1,2],[3,0]]);
```
(%o6) $false$

10.5 Combinación de listas

join($list_1, list_2$)	crea una nueva lista con los elementos de las listas $list_1$ y $list_2$ alternados

Combinar listas.

Maxima

He aquí dos listas, de igual longitud, combinadas.

```
(%i1) join([1,2],[a,b]);
```

(%o1) $[1, a, 2, b]$

Maxima

He aquí la combinación de dos listas de diferente longitud (`join` ignora los elementos sobrantes de la lista de mayor longitud).

```
(%i2) join([1,2],[a,b,c,d]);
```

(%o2) $[1, a, 2, b]$

10.6 Reordenamiento de listas

`sort`(*list*)	ordena los elementos de *list* en forma ascendente
`sort`(*list*, P)	ordena los elementos de *list* de acuerdo con el predicado P
`reverse`(*list*)	invierte el orden de los elementos de *list*
`permutations`(*list*)	devuelve un conjunto con todas las permutaciones distintas de los miembros de *list*
`unique`(*list*)	devuelve *list* sin redundancias, es decir, sin elementos repetidos

Ordenamiento de listas.

Maxima

Esto ordena los elementos de una lista en forma estándar. En casos simples como éste el ordenamiento es alfabético o numérico.

```
(%i1) sort([d,b,c,a]);
```

(%o1) $[a, b, c, d]$

Maxima

Esto ordenada los elementos de la lista en forma descendente.

```
(%i2) sort([11,1,2,4,5,5],">");
```

(%o2) $[11, 5, 5, 4, 2, 1]$

Maxima

Aquí se ordenada los elementos de la lista según sea la norma de cierto elemento menor que la norma del posterior.

```
(%i3) sort([3+5*%i,%i-8,4+%i,1+2*%i],
        lambda([a,b],abs(a)<abs(b))
      );
```

(%o3) $[2\,i + 1, i + 4, 5\,i + 3, i - 8]$

Maxima

Esto invierte el orden de los elementos de la lista.

```
(%i4) reverse([d,b,c,a]);
```

(%o4) $[a, c, b, d]$

Maxima

Aquí se tienen todas las permutaciones de $[a, b, c]$.

```
(%i5) permutations([a,b,c]);
```

(%o5) $\{[a, b, c]\,, [a, c, b]\,, [b, a, c]\,, [b, c, a]\,, [c, a, b]\,, [c, b, a]\}$

Maxima

De esta forma se eliminan los elementos repetidos en una lista.

```
(%i6) unique([11,1,2,2,4,5,5]);
```

(%o6) $[1, 2, 4, 5, 11]$

10.7 Agregar y quitar elementos de una lista

`append`$(list_1, list_2, \ldots, list_n)$	devuelve una lista cuyos elementos son los de la lista $list_1$ seguidos de los de $list_2, \ldots, list_n$
`cons`$(elem, list)$	agrega *elem* al inicio de la lista *list*
`endcons`$(elem, list)$	agrega *elem* al final de la lista *list*
`delete`$(elem, list)$	borra *elem* de la lista *list*
`rest`$(list, n)$	borra los primeros n elementos $(n > 0)$ o los últimos $-n$ elementos $(n < 0)$ de la lista *list*
`deleten`$(list, n)$	borra el n-ésimo elemento de la lista *list*

Agregar y quitar elementos de una lista.

Maxima

Aquí se genera una lista con los elementos de dos listas dadas.

```
(%i1) append([1,2,3],[a,b,c]);
```

(%o1) $[1, 2, 3, a, b, c]$

Maxima

Aquí se inserta `x` al inicio de la lista dada.

```
(%i2) cons(x,[a,b,c]);
```

(%o2) $[x, a, b, c]$

Maxima

En este caso se inserta `x` al final de la lista dada.

```
(%i3) endcons(x,[a,b,c]);
```

(%o3) $[a, b, c, x]$

Maxima

Esto devuelve una lista dada con los dos primeros elementos borrados.

```
(%i4) rest([a,g,f,c,x],2);
```

(%o4) $[f, c, x]$

Maxima

Esto devuelve una lista dada con los dos últimos elementos borrados.

```
(%i5) rest([a,g,f,c,x],-2);
```

(%o5) $[a, g, f]$

10.8 Reorganización de listas

`flatten`$(list)$	elimina todos los niveles que pueda tener la lista *list*
`partition`$(list, x)$	particiona la lista *list* en dos listas; una que contiene los elementos que no dependen de x, y otra, los que si dependen de x

Funciones para reorganizar listas anidadas.

Maxima

He aquí una lista a la que se le eliminan todos los niveles.

```
(%i1) flatten([7,2,[3,[5]],4]);
```

(%o1) $[7, 2, 3, 5, 4]$

Maxima

Esto particiona la lista $[a, b, a^2 + 1, c]$.

```
(%i2) partition([a,b,a^2+1,c],a);
```

(%o2) $[[b, c], [a, a^2 + 1]]$

10.9 Funciones adicionales para listas

`listp`($expr$)	verifica si $expr$ es una lista
`emptyp`($list$)	verifica si $list$ es una lista vacía
`length`($list$)	devuelve la longitud de $list$
$list_1 . list_2$	devuelve el producto escalar de $list_1$ y $list_2$
`apply`($f, [x_1, \ldots, x_n]$)	evalúa $f(x_1, \ldots, x_n)$
`map`($f, [a_1, \ldots, a_n]$)	devuelve la lista $[f(a_1), \ldots, f(a_n)]$
`map`($f, [a_1, \ldots, a_n], [b_1, \ldots, b_n], \ldots$)	devuelve la lista $[f(a_1, b_1, \ldots), \ldots, f(a_n, b_n, \ldots)]$
`outermap`($f, list_1, \ldots, list_n$)	aplica la función f a cada uno de los elementos del producto cartesiano $list_1 \times list_2 \ldots \times list_n$

Algunas funciones adicionales para listas.

Maxima

Con la función `length` es fácil obtener la longitud de una lista.

```
(%i1) length([7,9,0]);
(%o1) 3
```

Maxima

Aquí se usa la función `apply` para encontrar el elemento de mayor valor.

```
(%i2) apply(max,[11,1,2,4,5,5]);
(%o2) 11
```

Maxima

He aquí un ejemplo en el que se usa la función `map`.

```
(%i3) map(first,[[a,b],[c,d]]);
(%o3) [a, c]
```

Maxima

Este es otro ejemplo en el que se usa la función **map**. Para usos como este debe recordarse que las listas deben tener la misma cantidad de elementos.

```
(%i4) map("+",[1,2,9],[-1,3,7]);
```

(%o4) $[0, 5, 16]$

Maxima

He aquí un ejemplo en el que se usa la función **outermap**.

```
(%i5) outermap("^",[a,b],[c,d,e]);
```

(%o5) $[[a^c, a^d, a^e], [b^c, b^d, b^e]]$

Maxima

En este ejemplo se combina el uso de la función **map** con una función anónima (véase (%i26) y (%i28) de la sección 9.1) para obtener una lista que contiene los productos escalares de p con cada una de las sublistas de q.

```
(%i6) p:[x,y,z]$
      q:[[a1,b1,c1],[a2,b2,c2],[a3,b3,c3]]$
(%i8) map(lambda([h],p.h),q);
```

(%o8) $[c1\,z + b1\,y + a1\,x, c2\,z + b2\,y + a2\,x, c3\,z + b3\,y + a3\,x]$

CAPÍTULO 11

Arrays

`array`$(nombre, dim_1, \ldots, dim_n)$	crea un array de dimensión n, que debe ser menor o igual que 5
`array`$(nombre, tipo, dim_1, \ldots, dim_n)$	crea un array con sus elementos del tipo especificado (el tipo puede ser `fixnum` o `flonum`)

Creación de arrays.

`listarray`(A)	devuelve una lista con los elementos del array A
`arrayinfo`(A)	devuelve información acerca del array A
`fillarray`(A, B)	rellena el array A con los valores de B, que puede ser una lista u otro array

Algunas funciones para arrays.

Maxima

Esto define el array `u` de dimensión 2.

```
(%i9) array(u,2);
(%o9) u
```

Maxima

Esto indica que el array no tiene valores asignados.

```
(%i10) listarray(u);
```

(%o10) $[\#\#\#\#\#, \#\#\#\#\#, \#\#\#\#\#]$

Maxima

Esto asigna valores, a partir de una lista, al array.

```
(%i11) fillarray(u,[1,-7,8]);
```

(%o11) u

Maxima

Esto permite visualizar los valores asignados.

```
(%i12) listarray(u);
```

(%o12) $[1, -7, 8]$

Maxima

Aquí se define el array ϕ.

```
(%i13) array(phi,1,2);
```

(%o13) ϕ

Maxima

Esto asigna valores a ϕ.

```
(%i14) phi[0,0]:-1$ phi[0,1]:1$ phi[0,2]:4$
       phi[1,0]:4$ phi[1,1]:7$ phi[1,2]:5$
```

Maxima

Aquí se tiene una lista con los valores asignados a ϕ.

```
(%i22) listarray(phi);
```

(%o22) $[-1, 1, 4, 4, 7, 5]$

Maxima

Esto da información acerca de ϕ

```
(%i23) arrayinfo(phi);
```
(%o23) $[declared, 2, [1, 2]]$

Si el usuario asigna un valor a una variable subindicada antes de declarar el array correspondiente, se construye un array no declarado.

Los arrays no declarados, también conocidos por el nombre de "arrays de claves" (hashed arrays), son más generales que los arrays declarados. El usuario no necesita declarar su tamaño máximo y pueden ir creciendo de forma dinámica. Los subíndices de los arrays no declarados no necesariamente deben ser números.

Maxima

Esto asigna un valor a la variable `t` subindicada con `hola`.

```
(%i24) t[hola]:a;
```
(%o24) a

Maxima

Esto asigna un valor a la variable `t` subindicada con `adios`.

```
(%i25) t[adios]:b;
```
(%o25) b

Maxima

Ahora se muestran los valores asignados al array no declarado `t`.

```
(%i26) listarray(t);
```
(%o26) $[b, a]$

Maxima

Y también se muestra información con respecto a `t`.

```
(%i27) arrayinfo(t);
```
(%o27) $[hashed, 1, [adios], [hola]]$

CAPÍTULO 12

Matrices

12.1 Generación de Matrices

Las matrices en *Maxima* son expresiones que comprenden el operador `matrix` y cuyos argumentos son listas (ver sección 19.2). Esto indica que *Maxima* tiene establecida una diferencia entre una matriz y una lista de listas.

`matrix`$([a_{11}, a_{12}, \ldots, a_{1n}],$ $\ldots, [a_{m1}, a_{m2}, \ldots, a_{mn}])$	devuelve una matriz de orden $m \times n$
`genmatrix`(f, m, n)	devuelve una matriz $m \times n$ generada a partir de f
`entermatrix`(m, n)	devuelve una matriz de orden $m \times n$, cuyos elementos son leidos de forma interactiva
`ident`(n)	devuelve la matriz identidad de orden n
`zeromatrix`(m, n)	devuelve una matriz rectangular $m \times n$ con todos sus elementos iguales a cero
`diagmatrix`$(n, elem)$	devuelve una matriz diagonal de orden n con los elementos de la diagonal todos ellos iguales a $elem$

Funciones para generar matrices.

`diag_matrix`$(d_1, d_2, \ldots, d_n)$	devuelve una matriz diagonal con los elementos de la diagonal iguales a $d_1, d_2, \ldots, d_n$
`vandermonde_matrix` $([x_1, \ldots, x_n])$	devuelve una matriz n por n, cuya i-ésima fila es $[1, x_i, x_i^2, \ldots, x_i^{(n-1)}]$
`hilbert_matrix` (n)	devuelve la matriz de Hilbert n por n (si n no es un entero positivo, emite un mensaje de error)

Funciones para generar matrices.

Maxima

He aquí la definición de una matriz.

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

$$(\%o1)\quad \begin{bmatrix} 3 & 1 & 0 \\ 2 & 4 & 1 \\ 1 & 7 & 2 \end{bmatrix}$$

Maxima

Una lista de listas no es interpretada como matriz.

```
(%i2) [[3,1,0],[2,4,1],[1,7,2]];
```

$$(\%o2)\quad [[3, 1, 0], [2, 4, 1], [1, 7, 2]]$$

Maxima

Con `apply` puede convertirse una lista de listas a matriz.

```
(%i3) apply(matrix,[[3,1,0],[2,4,1],[1,7,2]]);
```

$$(\%o3)\quad \begin{bmatrix} 3 & 1 & 0 \\ 2 & 4 & 1 \\ 1 & 7 & 2 \end{bmatrix}$$

Maxima

Aquí se genera una matriz $(a_{i,j}) \in \mathbb{M}_{3\times 3} / a_{i,j} = i - 2j$.

```
(%i4) a[i,j]:=i-2*j $
```

```
(%i5) genmatrix(a,3,3);
```

(%o5) $$\begin{bmatrix} -1 & -3 & -5 \\ 0 & -2 & -4 \\ 1 & -1 & 3 \end{bmatrix}$$

Maxima

Esto genera una matriz de Vandermonde simbólica.

```
(%i6) vandermonde_matrix([x[0],x[1],x[2]]);
```

(%o6) $$\begin{bmatrix} 1 & x_0 & x_0^2 \\ 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \end{bmatrix}$$

Maxima

Esto genera una matriz de Vandermonde numérica.

```
(%i7) vandermonde_matrix([2,3,5]);
```

(%o7) $$\begin{bmatrix} 1 & 2 & 4 \\ 1 & 3 & 9 \\ 1 & 5 & 25 \end{bmatrix}$$

Maxima

He aquí una matriz de Hilbert de orden 3.

```
(%i8) hilbert_matrix(3);
```

(%o8) $$\begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \end{bmatrix}$$

12.2 Elegir elementos de matrices

$M[\,i\,]$ ó part(M,i)	devuelve la i-ésima fila de la matriz M
$M[\,i,j\,]$ ó part(M,i,j)	devuelve el elemento $M_{i,j}$ de M

Funciones para elegir elementos de matrices.

Maxima

Esto genera la matriz A mediante una función anónima.

```
(%i1) A:genmatrix(lambda([i,j],i-j),3,3);
```

(%o1) $$\begin{bmatrix} 0 & -1 & -2 \\ 1 & 0 & -1 \\ 2 & 1 & 0 \end{bmatrix}$$

Maxima

Aquí se selecciona el elemento $A_{1,2}$.

```
(%i2) A[1,2];
```

(%o2) -1

Maxima

Esto devuelve una matriz identidad de orden 3.

```
(%i3) ident(3);
```

(%o3) $$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Maxima

Esto selecciona la segunda fila de la matriz identidad.

```
(%i4) %[2];
```

(%o4) $[0,1,0]$

12.3 Operaciones matriciales

$A.B$	producto matricial
`invert`(M)	devuelve la inversa de la matriz M
M^^p	exponenciación matricial no conmutativa
`determinant`(M)	devuelve el determinante de la matriz M
`mat_trace`(M)	devuelve la traza de la matriz cuadrada M
`transpose`(M)	devuelve la transpuesta de la matriz M
`minor`(M, i, j)	devuelve el menor (i, j) de la matriz M
`adjoint`(M)	devuelve la matriz adjunta de la matriz M
`addcol`$(M, list_1, \ldots, list_n)$	añade la(s) columna(s) dada(s) por la(s) lista(s) (o matrices) a la matriz M
`addrow`$(M, list_1, \ldots, list_n)$	añade la(s) fila(s) dada(s) por la(s) lista(s) (o matrices) a la matriz M
`col`(M, i)	devuelve la i-ésima columna de la matriz M. El resultado es una matriz de una sola columna
`row`(M, i)	devuelve la i-ésima fila de la matriz M. El resultado es una matriz de una sola fila
`rank`(M, i, j)	calcula el rango de la matriz M
`setelmx`(x, i, j, M)	asigna el valor x al (i,j)-ésimo elemento de la matriz M y devuelve la matriz actualizada
`submatrix`$(i_1, \ldots, i_m, M, j_1, \ldots, j_n)$ `submatrix`$(i_1, \ldots, i_m, M)$ `submatrix`$(M, j_1, \ldots, j_n)$	devuelve una nueva matriz formada a partir de la matriz M pero cuyas filas $i_1, \ldots, 1_m$ y columnas $j_1, \ldots, j_n$ han sido eliminadas

Algunas operaciones matriciales.

`echelon`(M)	devuelve la forma escalonada de la matriz M, obtenida por eliminación gaussiana
`triangularize`(M)	devuelve la forma triangular superior de la matriz M, obtenida por eliminación gaussiana
`rowop`(M, i, j, λ)	devuelve la matriz que resulta de relizar la transformación $F_i \leftarrow F_i - \lambda * F_j$ con las filas F_i y F_j de la matriz M
`rowswap`(M, i, j)	intercambia las filas i y j de la matriz M
`columnop`(M, i, j, λ)	devuelve la matriz que resulta de relizar la transformación $C_i \leftarrow C_i - \lambda * C_j$ con las columnas C_i y C_j de la matriz M
`columnswap`(M, i, j)	intercambia las columnas i y j de la matriz M
`kronecker_product`(A, B)	devuelve el producto de Kronecker de las matrices A y B

Algunas operaciones matriciales.

Maxima

He aquí una matriz 2×2 de variables simbólicas.

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

(%o1) $$\begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

Maxima

He aquí la transpuesta de A.

```
(%i2) transpose(A);
```

(%o2) $$\begin{bmatrix} a & c \\ b & d \end{bmatrix}$$

Maxima

Esto da la inversa de A en forma simbólica.

```
(%i3) invert(A);
```

(%o3) $$\begin{bmatrix} \frac{d}{ad-bc} & -\frac{b}{ad-bc} \\ -\frac{c}{ad-bc} & \frac{a}{ad-bc} \end{bmatrix}$$

Maxima

Esto da el determinante de A.

```
(%i4) determinant(A);
```

(%o4) $a\,d - b\,c$

Maxima

He aquí una matriz racional 3×3.

```
(%i5) B:genmatrix(lambda([i,j],1/(i+j-1)),3,3);
```

(%o5) $$\begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \end{bmatrix}$$

Maxima

Esto da su inversa.

```
(%i6) invert(B);
```

(%o6) $$\begin{bmatrix} 9 & -36 & 30 \\ -36 & 192 & -180 \\ 30 & -180 & 180 \end{bmatrix}$$

Maxima

Multiplicando la inversa con la matriz original se obtiene la matriz identidad.

```
(%i7) %.B;
```

(%o7) $$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Maxima

Aquí se define una matriz M de orden 4×5.

```
(%i8) M:matrix([2,-1,2,-1,-8],[2,-2,3,-3,-20],
        [1,1,1,0,-2],[1,-1,4,3,4]);
```

$$(\%o8) \begin{bmatrix} 2 & -1 & 2 & -1 & -8 \\ 2 & -2 & 3 & -3 & -20 \\ 1 & 1 & 1 & 0 & -2 \\ 1 & -1 & 4 & 3 & 4 \end{bmatrix}$$

Maxima

Esto efectúa el intercambio de las filas 2 y 1 de la matriz M y devuelve la matriz resultante.

```
(%i9) rowswap(M,2,1);
```

$$(\%o9) \begin{bmatrix} 2 & -2 & 3 & -3 & -20 \\ 2 & -1 & 2 & -1 & -8 \\ 1 & 1 & 1 & 0 & -2 \\ 1 & -1 & 4 & 3 & 4 \end{bmatrix}$$

Maxima

Esto realiza la transformación $F_2 \leftarrow F_2 + F_1$ en la matriz M y devuelve la matriz resultante.

```
(%i10) rowop(M,2,1,-1);
```

$$(\%o10) \begin{bmatrix} 2 & -1 & 2 & -1 & -8 \\ 4 & -3 & 5 & -4 & -28 \\ 1 & 1 & 1 & 0 & -2 \\ 1 & -1 & 4 & 3 & 4 \end{bmatrix}$$

Maxima

He aquí el producto de Kronecker de dos matrices.

```
(%i11) kronecker_product(matrix([a,b],[c,d]),
        matrix([1,2,3],[4,5,6],[7,8,9]));
```

(%o11) $$\begin{bmatrix} a & 2a & 3a & b & 2b & 3b \\ 4a & 5a & 6a & 4b & 5b & 6b \\ 7a & 8a & 9a & 7b & 8b & 9b \\ c & 2c & 3c & d & 2d & 3d \\ 4c & 5a & 6a & 4d & 5d & 6d \\ 7c & 8c & 9c & 7d & 8d & 9d \end{bmatrix}$$

12.4 Funciones adicionales para matrices

`matrix_size`(M)	devuelve una lista con dos elementos que constituyen el número de filas y columnas de la matriz M, respectivamente
`matrixp`$(expr)$	verifica si $expr$ es una matriz
`mat_norm`(M, p)	devuelve la p-norma de la matriz M. Los valores admisibles para p son 1, `inf` y `frobenius`
`addmatrices`$(f, A, B, \ldots)$	devuelve la matriz $(f(a_{i,j}, b_{i,j}, \ldots))_{m\times n}$, donde $m \times n$ es el orden común de las matrices $A, B, \ldots$
`mat_unblocker`(M)	si M es una matriz de bloques, deshace los bloques de un nivel
`outermap`$(f, M_1, \ldots, M_n)$	aplica la función f a cada uno de los elementos del producto cartesiano $M_1 \times M_2 \ldots \times M_n$

Más funciones para matrices.

Maxima

He aquí la matriz D.

```
(%i1) d[i,j]:=1+(-1)^(i+j)$
(%i3) D:genmatrix(d,3,4);
```

(%o3) $$\begin{bmatrix} 2 & 0 & 2 & 0 \\ 0 & 2 & 0 & 2 \\ 2 & 0 & 2 & 0 \end{bmatrix}$$

Maxima

Esto devuelve una lista con el número de filas y columnas de la matriz D.

```
(%i4) matrix_size(D);
```

(%o4) $[3, 4]$

Maxima

Utilizando la función **matrixp** es posible comprobar que B es una expresión matricial.

```
(%i5) matrixp(D);
```

(%o5) $true$

Maxima

He aquí la norma de frobenius de la matriz B.

```
(%i6) mat_norm(D,frobenius);
```

(%o6) $2\sqrt{6}$

Maxima

A continuación se definen las matrices A, B y C con elementos simbólicos indexados.

```
(%i7) A:genmatrix(a,2,3);
```

(%o7) $$\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \end{bmatrix}$$

```
(%i8) B:genmatrix(b,2,3);
```

(%o8) $$\begin{bmatrix} b_{1,1} & b_{1,2} & b_{1,3} \\ b_{2,1} & b_{2,2} & b_{2,3} \end{bmatrix}$$

```
(%i9) C:genmatrix(c,2,3);
```

(%o9) $\begin{bmatrix} c_{1,1} & c_{1,2} & c_{1,3} \\ c_{2,1} & c_{2,2} & c_{2,3} \end{bmatrix}$

Maxima

Este es el resultado obtenido al aplicar f mediante la función **addmatrices** a las matrices A, B y C.

```
(%i10) addmatrices(f,A,B,C);
```

(%o10) $\begin{bmatrix} f(a_{1,1}, b_{1,1}, c_{1,1}) & f(a_{1,2}, b_{1,2}, c_{1,2}) & f(a_{1,3}, b_{1,3}, c_{1,3}) \\ f(a_{2,1}, b_{2,1}, c_{2,1}) & f(a_{2,2}, b_{2,2}, c_{2,2}) & f(a_{2,3}, b_{2,3}, c_{2,3}) \end{bmatrix}$

Maxima

Si f es "+", el resultado es $A + B + C$.

```
(%i11) addmatrices("+",A,B,C);
```

(%o11) $\begin{bmatrix} c_{1,1} + b_{1,1} + a_{1,1} & c_{1,2} + b_{1,2} + a_{1,2} & c_{1,3} + b_{1,3} + a_{1,3} \\ c_{2,1} + b_{2,1} + a_{2,1} & c_{2,2} + b_{2,2} + a_{2,2} & c_{2,3} + b_{2,3} + a_{2,3} \end{bmatrix}$

Maxima

Con **outermap** y **mat_unblocker** también es posible obtener el producto de Kronecker de dos matrices.

```
(%i12) outermap("*",matrix([a,b],[c,d]),
        matrix([1,2,3],[4,5,6],[7,8,9]));
```

(%o12) $\begin{bmatrix} \begin{bmatrix} a & 2a & 3a \\ 4a & 5a & 6a \\ 7a & 8a & 9a \end{bmatrix} & \begin{bmatrix} b & 2b & 3b \\ 4b & 5b & 6b \\ 7b & 8b & 9b \end{bmatrix} \\ \begin{bmatrix} c & 2c & 3c \\ 4c & 5c & 6c \\ 7c & 8c & 9c \end{bmatrix} & \begin{bmatrix} d & 2d & 3d \\ 4d & 5d & 6d \\ 7d & 8d & 9d \end{bmatrix} \end{bmatrix}$

```
(%i13) mat_unblocker(%);
```

(%o13) $\begin{bmatrix} a & 2a & 3a & b & 2b & 3b \\ 4a & 5a & 6a & 4b & 5b & 6b \\ 7a & 8a & 9a & 7b & 8b & 9b \\ c & 2c & 3c & d & 2d & 3d \\ 4c & 5a & 6a & 4d & 5d & 6d \\ 7c & 8c & 9c & 7d & 8d & 9d \end{bmatrix}$

12.5 Matrices asociadas a sistemas de ecuaciones

`coefmatrix`$([eq_1, \ldots, eq_m], [x_1, \ldots, x_n])$	devuelve la matriz de coeficientes para las variables $x_1, \ldots, x_n$ del sistema de ecuaciones lineales $eq_1, \ldots, eq_m$
`augcoefmatrix`$([eq_1, \ldots, eq_m], [x_1, \ldots, x_n])$	devuelve la matriz aumentada de coeficientes para las variables $x_1, \ldots, x_n$ del sistema de ecuaciones lineales $eq_1, \ldots, eq_m$

Funciones para generar matrices asociadas a sistemas de ecuaciones.

Maxima

He aquí la matriz de coeficientes del sistema de ecuaciones lineales, dado.

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

(%o1) $\begin{bmatrix} 2 & -3 \\ 1 & 1 \end{bmatrix}$

Maxima

Esta es la matriz aumentada del mismo sistema anterior.

```
(%i2) augcoefmatrix([2*x-3*y=1,x+y=3],[x,y]);
```

(%o2) $\begin{bmatrix} 2 & -3 & -1 \\ 1 & 1 & -3 \end{bmatrix}$

Maxima

Aquí se hace la transformación $F_2 \leftarrow F_2 - \frac{1}{2}F_1$ en la matriz aumentada anterior. A partir de éste último resultado se deduce que la solución del sistema de ecuaciones lineales, dado, será $y = 1$ y $x = 2$.

```
(%i3) rowop(%,2,1,1/2);
```

(%o3) $\begin{bmatrix} 2 & -3 & -1 \\ 0 & \frac{5}{2} & -\frac{5}{2} \end{bmatrix}$

12.6 Autovalores y autovectores

charpoly(M, x)	calcula el polinomio característico vinculado a la matriz M, en términos de la variable x
eigenvalues(M)	devuelve una lista con dos sublistas, la primera de éstas conteniendo los valores propios de la matriz M y la segunda, conteniendo sus correspondientes multiplicidades
eigenvectors(M)	devuelve una lista con dos elementos; el primero está formado por eigenvalues(M), el segundo es una lista de listas de vectores propios, una por cada valor propio, pudiendo haber más de un vector propio en cada lista

Obtención de autovalores y autovectores asociados a una matriz dada.

Maxima

He aquí una matriz 3×3.

```
(%i1) c[i,j]:=i+j+1$ C:genmatrix(c,3,3);
```

(%o1) $\begin{bmatrix} 3 & 4 & 5 \\ 4 & 5 & 6 \\ 5 & 6 & 7 \end{bmatrix}$

Maxima

Éste es el polinomio característico, en términos de la variable x, asociado a la matriz C.

```
(%i2) charpoly(C,x),expand;
```

(%o2) $-x^3+15\,x^2+6\,x$

Maxima

Aquí se tienen los valores propios, y su respectiva multiplicidad, asociados a la matriz C.

```
(%i3) eigenvalues(C);
```

(%o3) $\left[\left[-\frac{\sqrt{249}-15}{2},\frac{\sqrt{249}+15}{2},0\right],[1,1,1]\right]$

Maxima

Finalmente, se muestran los valores propios, con su respectiva multiplicidad, y vectores propios de la matriz C.

```
(%i4) eigenvectors(C);
```

(%o4) $\left[\left[\left[-\frac{\sqrt{249}-15}{2},\frac{\sqrt{249}+15}{2},0\right],[1,1,1]\right],\right.$
$\left[\left[\left[1,-\frac{\sqrt{249}-19}{28},-\frac{\sqrt{3}\,\sqrt{83}-5}{14}\right]\right],\right.$
$\left.\left.\left[\left[1,\frac{\sqrt{249}+19}{28},\frac{\sqrt{3}\,\sqrt{83}+5}{14}\right]\right],[[1,-2,1]]\right]\right]$

CAPÍTULO 13

Conjuntos

Los conjuntos en *Maxima* son expresiones que comprenden el operador `set` y cuyos argumentos son los elementos del mismo (ver sección 19.2). Esto indica que *Maxima* hace una diferencia entre conjuntos y listas, lo que permite trabajar con conjuntos cuyos elementos puedan ser también conjuntos o listas.

Maxima dispone de funciones para realizar operaciones con conjuntos, como la intersección o la unión. No obstante, debe tenerse en cuenta que los conjuntos deben ser finitos y definidos por enumeración.

13.1 Generación de conjuntos

`set`$(a_1, \ldots, a_n)$	construye un conjunto cuyos elementos son $a_1, \ldots, a_n$
$\{a_1, \ldots, a_n\}$	equivale a `set`$(a_1, \ldots, a_n)$
`set`$()$	genera un conjunto vacío
$\{\}$	equivale a `set`$()$
`makeset`$(f, vars, s)$	genera un conjunto cuyos miembros se generan a partir de f, siendo $vars$ una lista de variables de f y s un conjunto o lista de listas

Funciones para generar conjuntos.

Maxima

He aquí un conjunto formado por los elementos a, b, c, d.

```
(%i1) {a,b,c,d};
```

(%o1) $\{a, b, c, d\}$

Maxima

Al ingresar un conjunto cuyos elementos están desordenados y repetidos, éstos son ordenados y unificados de forma automática.

```
(%i2) {c,e,f,f,a,b,c,d,a};
```

(%o2) $\{a, b, c, d, e, f\}$

Maxima

La función **makeset** permite generar un conjunto. En este caso se considera una función de dos variables, la cual es aplicada a una lista de listas.

```
(%i3) makeset(i/j,[i,j],[[1,a],[2,b],[3,c],[4,d]]);
```

(%o3) $\left\{\frac{1}{a}, \frac{2}{b}, \frac{3}{c}, \frac{4}{d}\right\}$

Maxima

He aquí otro conjunto generado con **makeset**. Ahora se considera una función de una variable, la cual es aplicada a un conjunto de listas.

```
(%i4) makeset(x/2,[x],{[1],[2],[3],[4],[5]});
```

(%o4) $\left\{\frac{1}{2}, 1, \frac{3}{2}, 2, \frac{5}{2}\right\}$

Maxima

Las operaciones aritméticas son "listables".

```
(%i5) x^[1,2,3];
```

(%o5) $\left[x, x^2, x^3\right]$

Maxima

Sin embargo, las operaciones aritméticas no son "conjuntables".

```
(%i6) x^{1,2,3};
```

(%o6) $x^{\{1,2,3\}}$

Maxima

La sustitución si es "conjuntable", aunque siempre respetando la unificación.

```
(%i7) {x,x^2,x^3,x+1},x=1;
```

(%o7) $\{1,2\}$

13.2 Conversiones entre conjuntos y listas

`setify`(*list*)	forma un conjunto a partir de los miembros de la lista *list*
`fullsetify`(*list*)	sustituye los operadores de lista presentes en *conj* por operadores de conjuntos

Conversiones de listas a conjuntos.

`listify`(*conj*)	forma una lista a partir de los miembros del conjunto *conj*
`full_listify`(*conj*)	sustituye los operadores de conjunto presentes en *conj* por operadores de listas

Conversiones de conjuntos a listas.

Maxima

Esto convierte una lista en conjunto.

```
(%i1) setify([a,c,d,e]);
```

(%o1) $\{a,c,d,e\}$

Maxima

Esto convierte un conjunto en lista.

```
(%i2) listify({a,b,c,d});
```

(%o2) $[a, b, c, d]$

Maxima

Esto sustituye todos los operadores de conjuntos a operadores de listas.

```
(%i3) fullsetify([a,[b,c],[e,[f,g]],k]);
```

(%o3) $\{a, \{b, c\}, \{e, \{f, g\}\}, k\}$

Maxima

Esto sustituye todos los operadores de listas a operadores de conjuntos.

```
(%i4) full_listify(%);
```

(%o4) $[a, [b, c], [e, [f, g]], k]$

Considerando que, cualquier lista puede convertirse a conjunto, entonces puede usarse `create_list` para generar una lista y luego convertir la lista obtenida a conjunto. De esta manera se tiene una forma indirecta para generar conjuntos.

13.3 Elección de elementos de un conjunto

`part`(*conj*, *i*)	devuelve el *i*-ésimo elemento del conjunto *conj* respetando el ordenamiento y la unificación internos
`first`(*conj*), `second`(*conj*), ..., `tenth`(*conj*)	devuelve el primer, segundo, ..., décimo elemento de *conj*
`last`(*conj*)	devuelve el último elemento de *conj*

Selección de partes de un conjunto.

Maxima

Aquí se obtiene el primer elemento de un conjunto que ha sido ordenado y unificado internamente.

```
(%i1) part({b,c,c,d,a},1);
```

(%o1) a

Maxima

El quinto elemento no existe.

```
(%i2) part({b,c,c,d,a},5);
part: fell off the end.
-- an error. To debug this try: debugmode(true);
```

13.4 Prueba y búsqueda de elementos de un conjunto

Existe una función específica para averiguar si un elemento pertenece a un conjunto, no obstante hay funciones genéricas alternativas para ello.

`subset`$(conj, P)$	extrae todos los elementos de un conjunto *conj* que satisfacen el predicado P
`subsetp`$(conj_1, conj_2)$	devuelve *true* si y sólo si $con_1 \subseteq conj_2$
`elementp`$(x, conj)$	devuelve *true* si y sólo si x es elemento del conjunto *conj*
`member`$(x, conj)$	devuelve *true* si y sólo si x es miembro del conjunto *conj*
`freeof`$(x, conj)$	prueba si x no ocurre en ninguna parte *conj*
`freeof`$(x_1 \ldots, x_n, conj)$	equivale a $freeof(x_1; conj)$ **and** ... **and** $freeof(x_n; conj)$

Prueba de elementos de un conjunto.

Maxima

En este ejemplo se extrae, del conjunto $\{1, 2, 3, 4, 5, 7\}$, el subconjunto de todos los elementos pares.

```
(%i1) subset({1,2,3,4,5,7},evenp);
```

(%o1) $\{2,4\}$

Maxima

Aquí se extrae, del conjunto $\{1,2,3,4,5,7\}$, el subconjunto de todos los elementos t tales que $t<4$.

```
(%i2) subset({1,2,3,4,5,7},lambda([t],is(t<4)));
```

(%o2) $\{1,2,3\}$

Maxima

Aquí se extrae, del conjunto $\{1,2,3,4,5,7\}$, el subconjunto de todos los elementos t tales que $mod(t,3)=1$.

```
(%i3) subset({1,2,3,4,5,7},
        lambda([t],is(mod(t,3)=1)));
```

(%o3) $\{1,4,7\}$

Maxima

Esto verifica una contención de conjuntos.

```
(%i4) subsetp({1,2},{5,6,7,9,1,2});
```

(%o4) $true$

Maxima

Esto verifica que el elemento 2 pertenece al conjunto $\{2,4,5,7\}$.

```
(%i5) elementp(2,{2,4,5,7});
```

(%o5) $true$

Maxima

La función genérica **member** realiza los mismo.

```
(%i6) member(2,{2,4,5,7});
```

(%o6) $true$

Maxima

La función genérica *freeof* indica que 2 forma parte del conjunto $\{2,4,5,7\}$.

```
(%i7) freeof(2,{2,4,5,7});
(%o7) false
```

Maxima

La función genérica *freeof* indica que 6 y 5 no forman parte del conjunto $\{1,2,3\}$.

```
(%i8) freeof(6,5,{1,2,3});
(%o8) true
```

13.5 Agregar y quitar elementos de un conjunto

Es posible agregar y/o quitar elementos de un conjunto dado mediante dos funciones específicas de *Maxima*.

`adjoin`$(x, conj)$	devuelve el conjunto *conj* con el elemento x insertado
`disjoin`$(x, conj)$	devuelve el conjunto *conj* sin el elemento x

Funciones para agregar y quitar elementos de un conjunto.

Maxima

Aquí se inserta el elemento 2 en el conjunto $\{1,4,7\}$.

```
(%i1) adjoin(2,{1,4,7});
(%o1) {1,2,4,7}
```

Maxima

Esto elimina el elemento 4 de $\{1,4,7\}$.

```
(%i2) disjoin(4,{1,4,7});
(%o2) {1,7}
```

13.6 Reorganización de conjuntos

`flatten`($conj$)	elimina todos los niveles en *conj*
`set_partitions`($conj$)	devuelve el conjunto de todas las particiones de *conj*
`set_partitions`($conj, n$)	devuelve un conjunto con todas las descomposiciones de *conj* en n conjuntos no vacíos disjuntos

Reorganización de conjuntos anidados.

Maxima

He aquí un conjunto al que se le eliminan todos los niveles.

```
(%i1) flatten({4,{1,3},{4,{7,8}},10});
```

(%o1) $\{1, 3, 4, 7, 8, 10\}$

Maxima

Esto da todas las particiones de $\{a, b, c\}$; y, seguidamente las descomposiciones del mismo $\{a, b, c\}$ en dos conjuntos no vacíos disjuntos.

```
(%i2) set_partitions({a,b,c});
```

(%o2) $\{\{\{a\}, \{b\}, \{c\}\}, \{\{a\}, \{b, c\}\}, \{\{a, b\}, \{c\}\}, \{\{a, b, c\}\}, \{\{a, c\}, \{b\}\}\}$

```
(%i3) set_partitions({a,b,c},2);
```

(%o3) $\{\{\{a\}, \{b, c\}\}, \{\{a, b\}, \{c\}\}, \{\{a, c\}, \{b\}\}\}$

13.7 Operaciones con conjuntos

Las más comunes operaciones matemáticas con conjuntos están implementadas en *Maxima*.

`union`$(conj_1, \ldots, conj_n)$	devuelve la unión de los conjuntos $conj_1, \ldots, conj_n$
`intersection`$(conj_1, \ldots, conj_n)$	devuelve la intersección de los conjuntos $conj_1, \ldots, conj_n$
`setdifference`$(conj_1, conj_2)$	devuelve la diferencia de los conjuntos $conj_1, conj_2$
`powerset`$(conj)$	devuelve el conjunto potencia de $conj$
`subsetp`$(conj_1, conj_2)$	devuelve *true* si y sólo si el conjunto $conj_1$ es un subconjunto de $conj_2$
`setequalp`$(conj_1, conj_2)$	devuelve *true* si y sólo los conjuntos $conj_1$ y $conj_2$ son iguales
`symmdifference`$(conj_1, conj_2)$	devuelve la diferencia simétrica de los conjuntos $conj_1, conj_2$
`cartesian_product`$(conj_1, \ldots, conj_n)$	devuelve el producto cartesiano de los conjuntos $conj_1, \ldots, conj_n$ en forma de listas
`disjointp`$(conj_1, conj_2)$	devuelve *true* si y sólo si los conjuntos $conj_1$ y $conj_2$ son disjuntos
`equiv_classes`$(conj, F)$	devuelve el conjunto de las clases de equivalencia del conjunto $conj$ respecto de la relación de equivalencia F

Operaciones con conjuntos.

Maxima

Aquí se definen los conjuntos A y B.

```
(%i1) A:{a,b,c,d}$ B:{c,d,e,f,g}$
```

Maxima

Esto da la unión de A y B.

```
(%i3) union(A,B);
```

(%o3) $\{a, b, c, d, e, f, g\}$

Maxima

Y esto la intersección.

```
(%i4) intersection(A,B);
```

(%o4) $\{c, d\}$

Maxima

La diferencia $A - B$ se calcula así.

```
(%i5) setdifference(A,B);
```

(%o5) $\{a, b\}$

Maxima

En este caso las diferencias simétricas $A \ominus B$ y $B \ominus A$ son iguales.

```
(%i6) symmdifference(A,B);
```

(%o6) $\{a, b, e, f, g\}$

Maxima

He aquí el conjunto potencia del conjunto A.

```
(%i7) powerset(A);
```

(%o7) $\{\{\}, \{a\}, \{a, b\}, \{a, b, c\}, \{a, b, c, d\}, \{a, b, d\}, \{a, c\}, \{a, c, d\}, \{a, d\}, \{b\}, \{b, c\}, \{b, c, d\}, \{b, d\}, \{c\}, \{c, d\}, \{d\}\}$

Maxima

Esto da el producto cartesiano $A \times B$.

```
(%i8) cartesian_product(A,B);
```

(%o8) $\{[a, c], [a, d], [a, e], [a, f], [a, g], [b, c], [b, d], [b, e], [b, f], [b, g], [c, c], [c, d], [c, e], [c, f], [c, g], [d, c], [d, d], [d, e], [d, f], [d, g]\}$

Maxima

Esto indica que los conjuntos A y B no son disjuntos.

```
(%i9) disjointp(A,B);
```
(%o9) $false$

Maxima

En este ejemplo, las clases de equivalencia son números que difieren en un múltiplo de 3.

```
(%i10) equiv_classes({1,2,3,4,5,6,7},lambda([s,t],
       mod(s-t,3)=0));
```
(%o10) $\{\{1,4,7\},\{2,5\},\{3,6\}\}$

13.8 Funciones adicionales para conjuntos

`setp`(*expr*)	devuelve *true* si y sólo si *expr* es un conjunto de *Maxima*
`emptyp`(*conj*)	devuelve *true* si y sólo si *conj* es el conjunto vacío
`cardinality`(*conj*)	devuelve el número de elementos del conjunto *conj*
`map`$(f,\{a_1,\ldots,a_n\})$	devuelve $\{f(a_1),\ldots,f(a_n)\}$
`map`$(f,\{a_1,\ldots,a_n\},\{b_1,\ldots,b_n\},\ldots)$	devuelve el conjunto $\{f(a_1,b_1,\ldots),\ldots,f(a_n,b_n,\ldots)\}$, siempre que los conjuntos estén bien ordenados

Algunas funciones adicionales.

Maxima

He aquí el cardinal de un conjunto.

```
(%i1) cardinality({a,b,c,a});
```
(%o1) 4

Maxima

Para conjuntos bien ordenados, al aplicar la función **map**, se obtiene un resultado acorde con el ordenamiento de tales conjuntos.

```
(%i2) map(f,{a,b,c},{x,y,z});
```

(%o2) $\{f(a,x), f(b,y), f(c,z)\}$

Maxima

En este caso primero son ordenados los conjuntos y luego devuelve un resultado que coincide con el de (%o37) .

```
(%i3) map(f,{c,a,b},{y,x,z});
```

(%o3) $\{f(a,x), f(b,y), f(c,z)\}$

CAPÍTULO 14

Gráficos

Maxima no ha sido programado para elaborar sus propios gráficos, de modo que alternativamente utiliza un programa externo que realiza esta tarea. El usuario se limita a ordenar el tipo de gráfico deseado y *Maxima* se encarga de comunicárselo al programa gráfico que esté activo en ese momento, que por defecto será *Gnuplot*. La otra aplicación gráfica que utiliza *Maxima* es *Openmath*[1].

Las funciones `plot2d` y `plot3d`[2] son funciones para obtener gráficos que están definidas en el propio núcleo de *Maxima* (téngase presente que existen otras funciones similares definidas en un paquete llamado `draw`[3]).

Lo antes mencionado ocasiona que los gráficos generados con las funciones `plot2d` y `plot3d` se muestren en una ventana aparte del cuaderno de trabajo actual. No obstante, en contraparte, *wxMaxima* incorpora funciones alternativas, cuyos nombres se obtienen anteponiendo `wx` a `plot2d` y `plot3d`, respectivamente. La diferencia con las anteriores es que éstas funciones devuelven todos los gráficos en el cuaderno de trabajo actual.

[1]Tanto *Gnuplot* como *Openmath* son softwares libres y una copia de cada uno de ellos es distribuida conjuntamente con *Maxima*.

[2]Con `plot2d` y `plot3d` es posible dar instrucciones tanto a *Gnuplot* como a *Openmath* sin que sea necesario tener conocimiento alguno de la sintaxis de estos programas.

[3]El paquete `draw` (ver el capítulo 16) está orientado exclusivamente a *Gnuplot*. Éste incorpora las funciones `draw2d` y `draw3d`, las cuales dan resultados similares a `plot2d` y `plot3d`. Además incorpora funciones para graficar funciones definidas implícitamente.

14.1 Gráficos básicos

Por comodidad, la mayoría de los ejemplos de esta sección se ejecutarán con las funciones incorporadas en *wxMaxima*; sin embargo, todos estos ejemplos pueden ser ejecutados sin ningún problema con las funciones estándar de *Maxima*.

`plot2d`$(f, [x, x_{min}, x_{max}])$	muestra un gráfico de $y = f(x)$, con $x_{min} \leq x \leq x_{max}$, en una ventana independiente
`plot2d`$([f_1, \ldots, f_n], [x, x_{min}, x_{max}])$	muestra un gráfico de $y = f_1(x), \ldots, y = f_n(x)$, con $x_{min} \leq x \leq x_{max}$, en una ventana independiente
`wxplot2d`$(f, [x, x_{min}, x_{max}])$	muestra un gráfico de $y = f(x)$, con $x_{min} \leq x \leq x_{max}$, en el cuaderno de trabajo actual
`wxplot2d`$([f_1, \ldots, f_n], [x, x_{min}, x_{max}])$	muestra un gráfico de $y = f_1(x), \ldots, y = f_n(x)$, con $x_{min} \leq x \leq x_{max}$, en el cuaderno de trabajo actual

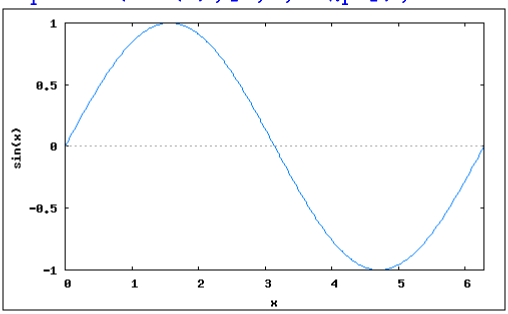
Trazado básico de funciones.

Maxima

Esto traza un gráfico de sen(x) como una función de x. Para x variando desde 0 hasta 2π.

```
(%i1) wxplot2d(sin(x),[x,0,2*%pi]);
```

(%t1)



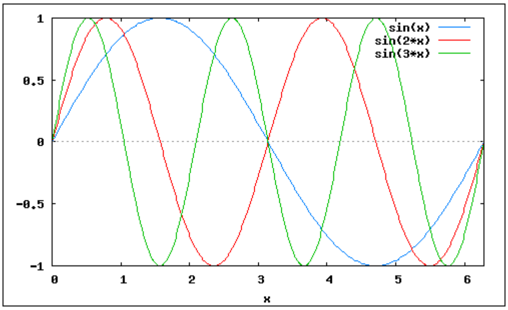
(%o1)

Maxima

Puede darse una lista de funciones para que sean trazadas. Por defecto, a cada función se le asigna un color específico.

```
(%i2) wxplot2d([sin(x),sin(2*x),sin(3*x)],
        [x,0,2*%pi]);
```

(%t2)



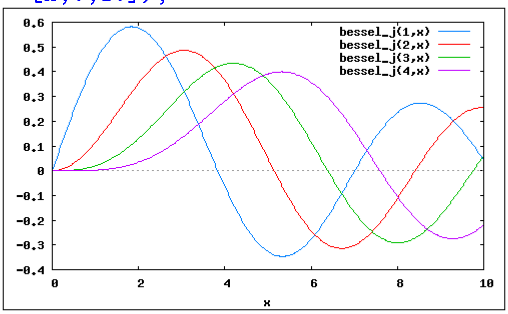
(%o2)

Maxima

Aquí se utiliza la función **create_list** para generar una lista de las funciones de Bessel, $J_n(x)$, con n variando desde 1 hasta 4. Luego se devuelve la gráfica de dichas funciones.

```
(%i3) wxplot2d(create_list(bessel_j(n,x),n,1,4),
        [x,0,10]);
```

(%t3)



(%o3)

14.2 Opciones

Hay muchas opciones para escoger en el trazado de gráficos. La mayoría de las veces, la aplicación gráfica invocada por *Maxima* probablemente tomará opciones bastante buenas. Sin embargo, si se quiere obtener los mejores dibujos posibles para objetivos particulares, debería ayudarse a la aplicación en particular en la elección de algunas de sus opciones.

Hay un mecanismo general para especificar opciones en las funciones de *Maxima*. Cada opción tiene un nombre definido. Como últimos argumentos de una función, como `plot2d` (`wxplot2d`), se puede incluir una secuencia de la forma $[nombre, valor]$ para especificar los valores de varias opciones. A cualquier opción para la cual no se indique un valor explícito se le asigna su valor por defecto.

`plot2d`$(f, [x, x_{min}, x_{max}], [opcion, valor])$	devuelve un gráfico, especificando un valor particular para una opción, en otra ventana

Elección de una opción para el trazado de un gráfico.

Opción	*Val. por defecto*	*Descripción*
`y`	no presenta	rango vertical del gráfico
`nticks`	29	número inicial de puntos utilizados por el procedimiento adaptativo para la representación de funciones
`adapt_depth`	10	número máximo de particiones utilizado por el algoritmo adaptativo de representación gráfica
`xlabel`	x	etiqueta del eje horizontal en gráficos 2d
`ylabel`	nombre de la función f, dada	etiqueta del eje vertical en gráficos 2d

Algunas de las opciones de `plot2d` (`wxplot2d`).

Opción	*Val. por defecto*	*Descripción*
`logx`	no presenta	hace que el eje horizontal en los gráficos 2d se dibuje en la escala logarítmica (no necesita de parámetros adicionales)
`logy`	no presenta	hace que el eje vertical en los gráficos 2d se dibuje en la escala logarítmica (no necesita de parámetros adicionales)
`legend`	nombre de la función f, dada	etiquetas para las expresiones de los gráficos 2d. Si hay más expresiones que etiquetas, éstas se repetirán
`style`	`lines,1,1`	estilos a utilizar para las funciones o conjuntos de datos en gráficos 2d
`gnuplot_preamble`	“ ”	introduce instrucciones de gnuplot antes de que se haga el dibujo
`plot_format`	`gnuplot`	determina qué programa gráfico se va a utilizar
`gnuplot_term`	`default`	establece el terminal de salida para Gnuplot; algunos valores posibles son: *dumb*, *png*, *jpg*, *eps*, *pdf*, *gif*, *svg*, etc. (esta opción únicamente puede utilizarse con `plot2d`, y no con `wxplot2d`)
`plot_realpart`	`false`	si `plot_realpart` vale `true`, se representará la parte real de un valor complejo
`run_viewer`	`true`	controla si el visor apropiado para la salida gráfica debe ejecutarse o no

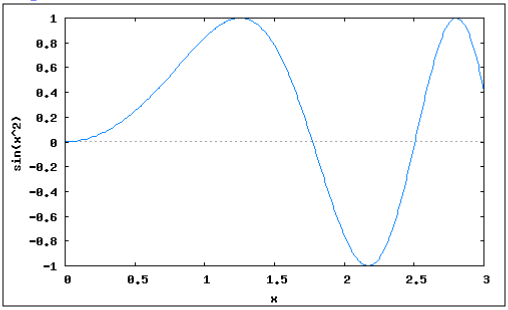
Algunas de las opciones de `plot2d` (`wxplot2d`).

Maxima

He aquí un gráfico para el cual todas las opciones tienen asignados sus valores por defecto.

```
(%i1) wxplot2d(sin(x^2),[x,0,3]);
```

(%t1)



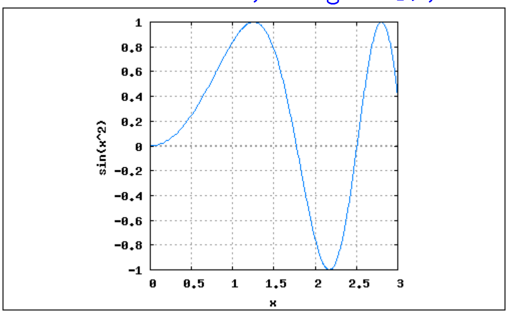
(%o1)

Maxima

La instrucción `set size ratio 1` iguala las escalas para los ejes x e y; y la instrucción `set grid` añade una rejilla al gráfico. Ambas instrucciones corresponden a la opción `gnuplot_preamble`.

```
(%i2) wxplot2d(sin(x^2),[x,0,3],[gnuplot_preamble,
        “set size ratio 1; set grid”]);
```

(%t2)

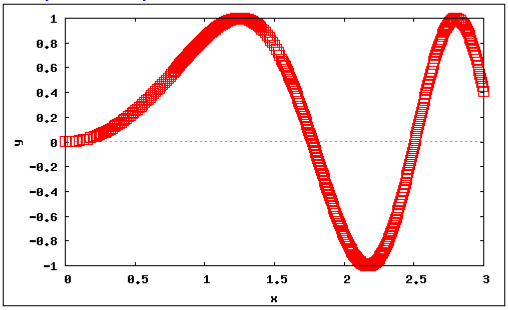


(%o2)

Maxima

Esto especifica el estilo del gráfico y la etiqueta para el eje y.

```
(%i3) wxplot2d(sin(x^2),[x,0,3],[style,[points,3,2,7]],
        [ylabel,y]);
```

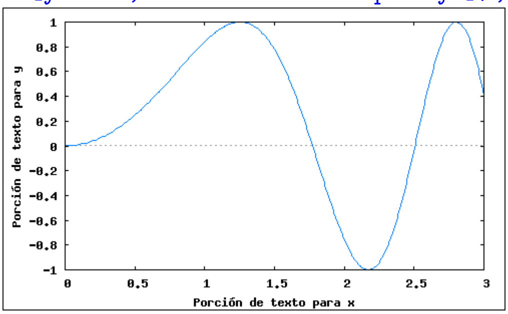
(%t3) 

(%o3)

Maxima

Las expresiones que se dan como etiquetas puede ser cualquier porción de texto, pero ésta debe ponerse entre comillas.

```
(%i4) wxplot2d(sin(x^2),[x,0,3],
        [xlabel,"Porción de texto para x"],
        [ylabel,"Porción de texto para y"]);
```

(%t4) 

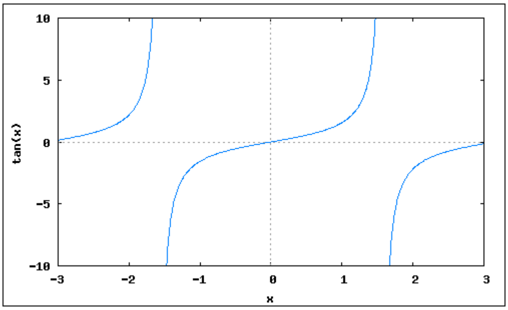
(%o4)

Maxima

Es posible trazar funciones que tienen singularidades. Para este efecto resulta bastante útil la opción y que permite indicar el rango.

```
(%i5) wxplot2d(tan(x),[x,-3,3],[y,-10,10]);
      plot2d: some values were clipped.
```

(%t5)



(%o5)

14.3 Gráficos de puntos y líneas

Con `plot2d` (`wxplot2d`) pueden graficarse puntos del plano (aislados) o poligonales que unen puntos del plano.

plot2d([discrete, $[[x_1, y_1], \ldots, [x_n, y_n]]$], [style, points])	devuelve el gráfico de los puntos aislados $(x_1, y_1), \ldots (x_n, y_n)$
plot2d([discrete, $[x_1, \ldots, x_n], [y_1, \ldots, y_n]$], [style, points])	devuelve el gráfico de los puntos aislados $(x_1, y_1), \ldots (x_n, y_n)$
plot2d([discrete, $[[x_1, y_1], \ldots, [x_n, y_n]]$], [style, lines])	devuelve el gráfico de una poligonal que une los puntos $(x_1, y_1), \ldots (x_n, y_n)$

Gráficos de puntos aislados y poligonales.

plot2d([discrete, $[x_1,\dots,x_n]$, $[y_1,\dots,y_n]$], [style, lines])	devuelve el gráfico de una poligonal que une los puntos $(x_1,y_1),\dots(x_n,y_n)$
plot2d([discrete, $[[x_1,y_1],\dots,[x_n,y_n]]$])	devuelve el gráfico de una poligonal que une los puntos $(x_1,y_1),\dots(x_n,y_n)$
plot2d([discrete, $[x_1,\dots,x_n]$, $[y_1,\dots,y_n]$])	devuelve el gráfico de una poligonal que une los puntos $(x_1,y_1),\dots(x_n,y_n)$

Gráficos de puntos aislados y poligonales.

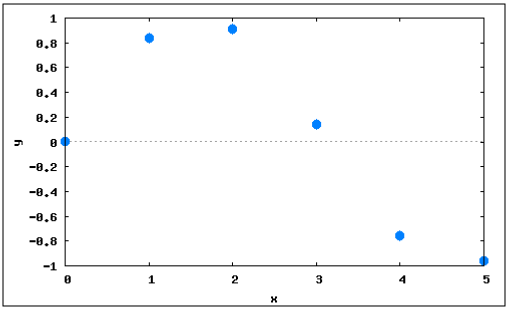
Maxima

De esta forma se grafica una lista de puntos aislados.

```
(%i1) pts:create_list([i,sin(i)],i,0,5);
```

(%o1) $[[0,0],[1,\sin(1)],[2,\sin(2)],[3,\sin(3)],[4,\sin(4)],[5,\sin(5)]]$

```
(%i2) wxplot2d([discrete,pts],[style,points]);
```

(%t2) 

(%o2)

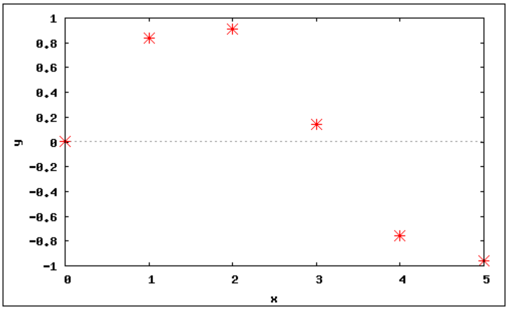
Cabe destacar que los estilos **points** y **lines** también aceptan opciones. Éstas deben ingresarse como una secuencia de tres números enteros positivos de la forma [points n_1, n_2, n_3] (en el caso de points), o dos números enteros positivos de la forma [lines n_1, n_2] (en el caso de lines). En ambos casos el primer valor, n_1, corresponde al tamaño de los puntos o grosor de la línea, según sea el caso; el segundo, n_2, al color ($1 \equiv$ *azul*, $2 \equiv$ *rojo*, $3 \equiv$ *verde*, $4 \equiv$ *violeta*, $5 \equiv$ *negro*, $6 \equiv$ *celeste*). Finalmente para points está el tercer valor, n_3, que corresponde a la forma en que se visualiza el punto ($1 \equiv$ *disco*, $2 \equiv$ *círculo*, $3 \equiv$ *cruz*, $4 \equiv$ *aspa*, $5 \equiv$ *asterisco*, $6 \equiv$ *cuadrado relleno*, 7 ≡ *cuadrado sin rellenar*, 8 ≡ *triángulo relleno*, 9 ≡ *triángulo sin rellenar*, 10 ≡ *triángulo relleno invertido*, 11 ≡ *triángulo sin rellenar invertido*, 12 ≡ *rombo relleno*, 13 ≡ *rombo sin rellenar*).

Maxima

Aquí se muestran los puntos de la salida `%o9` con opciones que cambian el tamaño, color y forma de los mismos.

```
(%i3) wxplot2d([discrete,pts],[style,[points,3,2,5]]);
```

(%t3)



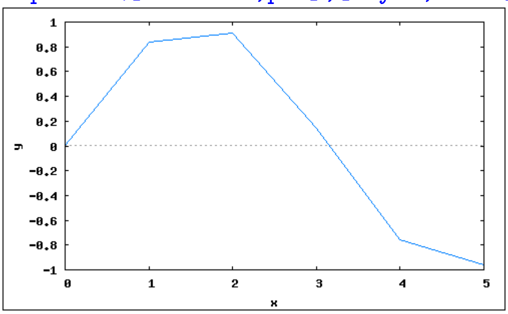
(%o3)

Maxima

Aquí se muestran los puntos de la salida `%o9` unidos por una línea quebrada.

```
(%i4) wxplot2d([discrete,pts],[style,lines]);
```

(%t4)



(%o4)

14.4 Gráficos paramétricos y polares

`plot2d([parametric,` $f_x, f_y, [t, t_{min}, t_{max}]$`])`	traza un gráfico paramétrico
`plot2d([` `[parametric,` $f_x, f_y,$ $[t, t_{min}, t_{max}]$`]`, `[parametric,` $g_x, g_y,$ $[s, s_{min}, s_{max}]$`]`, ...`])`	traza varias curvas paramétricas juntas

Gráficos de curvas definidas en forma paramétrica.

`plot2d(`$f(t)$, $[t, t_{min}, t_{max}]$, `[gnuplot_preamble, "set polar"])`	traza un gráfico polar
`plot2d([`$f(t), g(t), \ldots$`]` $[t, t_{min}, t_{max}]$, `[gnuplot_preamble, "set polar"])`	traza varias curvas polares juntas

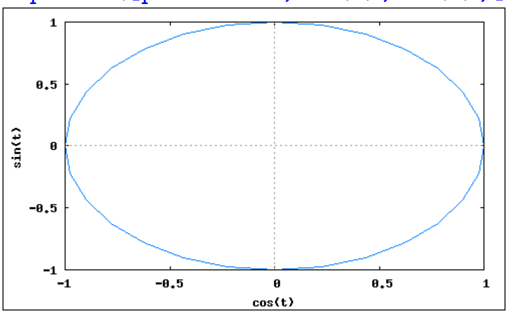
Gráficos de curvas definidas en forma polar.

Maxima

Esto devuelve la gráfica de la curva $t \to (\cos(t), \mathrm{sen}(t))$ para $t \in [0, 2\pi]$.

```
(%i1) wxplot2d([parametric,cos(t),sin(t),[t,0,2*%pi]]);
```

(%t1)



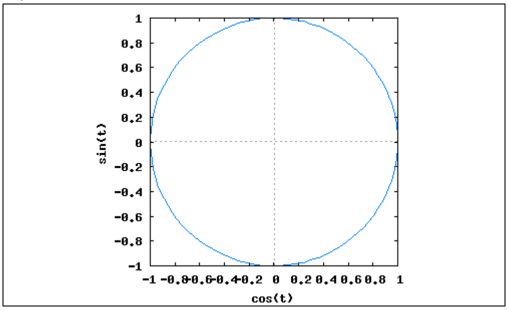
(%o1)

Maxima

Utilizando las opciones adecuadas es posible mejorar la presentación de la salida (%t1) .

```
(%i2) wxplot2d(
        [parametric,cos(t),sin(t),[t,0,2*% pi]],
        [gnuplot_preamble,“set size ratio 1”],
        [nticks,100]
      );
```

(%t2)



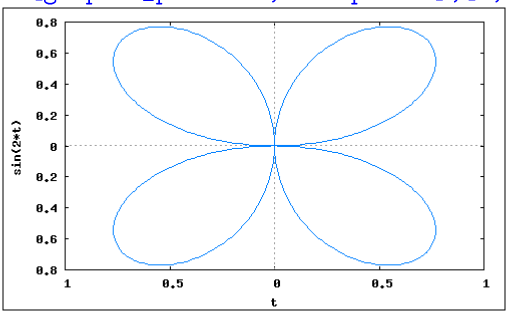
(%o2)

Maxima

Esto devuelve la gráfica de $\rho = \operatorname{sen}(2t)$.

```
(%i3) wxplot2d(sin(2*t),[t,0,2* %pi],
        [gnuplot_preamble,“set polar”],[x,-1,1]);
```

(%t3)



(%o3)

14.5 Combinación de gráficos

Adicionalmente, la función `plot2d` (`wxplot2d`) permite combinar varios tipos de gráficos (salvo los polares) y presentarlos en un mismo sistema coordenado.

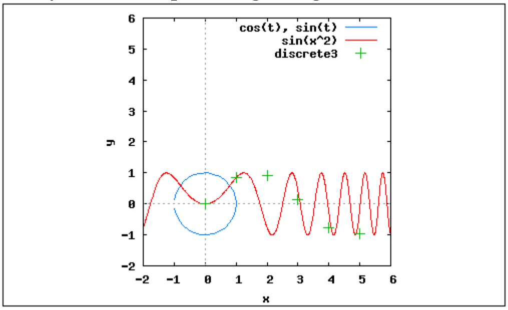
Maxima

Aquí se combinan varios tipos de gráficos.

```
(%i1) pts:create_list([i,sin(i)] , i, 0, 5)$
(%i2) wxplot2d([
        [parametric,cos(t),sin(t)],[t,0,2*%pi]
        sin(x^2),
        [discrete,pts]]
        [x,-2,6],[y,-2,6],
        [style,lines,lines,points]
        [gnuplot_preamble,“set size ratio 1”]);
```

plot2d: expression evaluates to non-numeric value everywhere in plotting range.

(%t2)



(%o2)

14.6 Gráficos de superficies tridimensionales

Para graficar superficies en $\mathbb{R}^3$ se utiliza la función `plot3d` (`wxplot3d`). Es preciso mencionar que, al utilizar la función `wxplot3d` el gráfico resultante será mostrado en el cuaderno de trabajo actual; no obstante, se pierde la interacción en tiempo real con el gráfico. Esto no sucede si se utiliza la función `plot3d`, ya que en este caso basta con hacer clic sobre la figura y, sin soltar el botón del mouse, arrastrarlo para que la superficie gire en tiempo real.

plot3d(f, $[x, x_{min}, x_{max}]$, $[y, y_{min}, y_{max}]$)	muestra un gráfico de $z = f(x, y)$, con $x_{min} \leq x \leq x_{max}$ y $y_{min} \leq y \leq y_{max}$, en una ventana independiente; en esta ventana es posible interactuar en tiempo real con dicho gráfico
wxplot3d(f, $[x, x_{min}, x_{max}]$, $[y, y_{min}, y_{max}]$)	muestra un gráfico de $z = f(x, y)$, con $x_{min} \leq x \leq x_{max}$ y $y_{min} \leq y \leq y_{max}$, en el cuaderno de trabajo actual y no hay interacción en tiempo real con el gráfico

Trazado básico de funciones 3D.

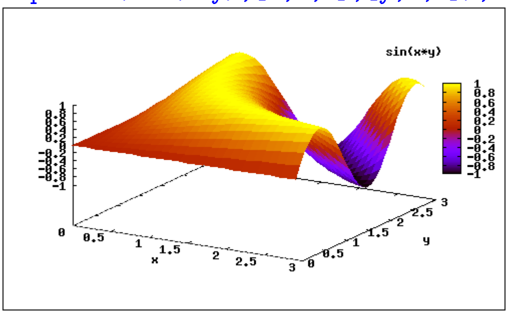
Al igual que plot2d (wxplot2d), la función plot3d (wxplot3d) también incluye una serie de opciones para obtener los mejores dibujos posibles.

Maxima

Esto traza un gráfico de la función $f(x, y) = \text{sen}(xy)$.

```
(%i1) wxplot3d(sin(x*y),[x,0,3],[y,0,3]);
```

(%t1)



(%o1)

Opción	*Val. por defecto*	*Descripción*
grid	30, 30	establece el número de puntos de la rejilla en las direcciones x e y
transform_xy	false	se usa para transformar las tres coordenadas
gnuplot_term	default	similar a plot2d (pág. 177)
gnuplot_preamble	" "	similar a plot2d (pág. 177)
plot_format	gnuplot	similar a plot2d (pág. 177)

Algunas de las opciones de `plot3d` (`wxplot3d`)

Maxima

Esto traza un gráfico de $f(x, y) = \text{sen}(xy)$ con el programa gráfico `openmath`.

```
(%i2) plot3d(sin(x*y),[x,0,3],[y,0,3],
        [plot_format,openmath]);
(%o2)
```

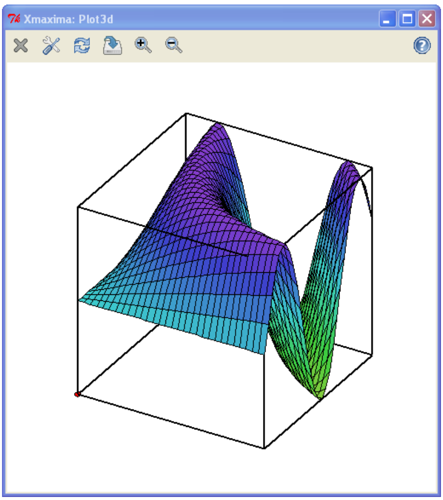


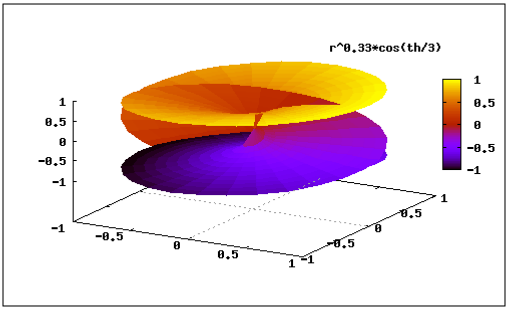
Figura 14.1: Gráfico obtenido a partir de (%i2) .

Maxima

Esto traza un gráfico tridimensional de la superficie definida por $r^{0.33}\cos(\frac{th}{3})$ en coordenadas cilíndricas. En este caso se ha realizado una transformación de coordenadas. El usuario puede definir sus propias transformaciones usando la función `make_transform`$(vars, f_x, f_y, f_z)$. Por ejemplo, aquí se ha usado la transformación `polar_to_xy : make_transform([r, th, z], r * cos(th), r * sin(th), z)`.

```
(%i3) wxplot3d(r^.33*cos(th/3),
        [r,0,1],[th,0,6*%pi],
        [grid,12,80],
        [transform_xy,polar_to_xy]);
```

(%t3)



(%o3)

`plot3d` (`wxplot3d`) también permite trazar la gráfica de una superficie definida en forma paramétrica; sin embargo, no permite graficar curvas en $\mathbb{R}^3$.

`plot3d`$([x(u,v), y(u,v), z(u,v)], [u, u_{min}, u_{max}], [v, v_{min}, v_{max}])$	traza el gráfico de una superficie paramétrica en una ventana independiente, siendo posible la interacción en tiempo real con dicho gráfico
`wxplot3d`$([x(u,v), y(u,v), z(u,v)], [u, u_{min}, u_{max}], [v, v_{min}, v_{max}])$	traza el gráfico de una superficie paramétrica en el cuaderno de trabajo actual, siendo imposible la interacción en tiempo real con dicho gráfico

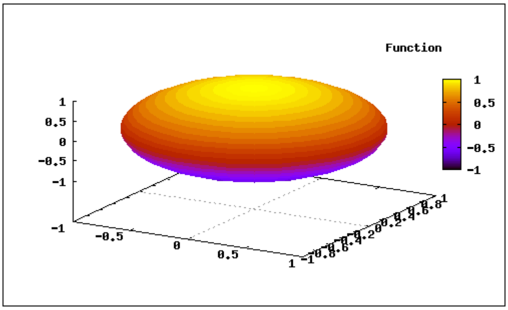
Trazado básico de superficies definidas en forma paramétrica.

Maxima

Esto devuelve la gráfica de la esfera unitaria definida paramétricamente mediante $(u, v) \to (\operatorname{sen} u \operatorname{sen} v, \cos u \operatorname{sen} v, \cos v)$, $0 \leq u \leq 2\pi$, $-\frac{\pi}{2} \leq v \leq \frac{\pi}{2}$.

```
(%i4) wxplot3d([cos(u)*cos(v),sin(u)*cos(v),sin(v)],
        [u,0,2*%pi],[v,-%pi/2,%pi/2]);
```

(%t4)



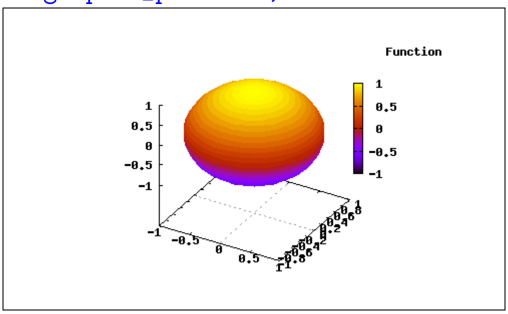
(%o4)

Maxima

Utilizando las opciones adecuadas es posible mejorar la presentación de la salida %o21.

```
(%i5) wxplot3d([cos(u)*cos(v),sin(u)*cos(v),sin(v)],
        [u,0,2*%pi],[v,-%pi/2,%pi/2],
        [gnuplot_preamble,"set size ratio 1"]);
```

(%t5)



(%o5)

14.7 Gráficos de densidad y contornos

Un gráfico de contorno le da esencialmente un "mapa topográfico" de una función. Los contornos unen puntos sobre la superficie que tienen la misma altura. El valor por defecto permite obtener contornos correspondientes a una secuencia de valores de z igualmente espaciados.

`plot3d`$(f(x,y)$, $[x, x_{min}, x_{max}]$, $[y, y_{min}, y_{max}]$, [`gnuplot_preamble`, "`set pm3d map`"])	traza un gráfico de densidad de $f(x,y)$ en el rectángulo $[x_{min}, x_{max}] \times [y_{min}, y_{max}]$
`contour_plot`$(f(x,y)$, $[x, x_{min}, x_{max}]$, $[y, y_{min}, y_{max}])$	dibuja las curvas de nivel de $f(x,y)$ en el rectángulo $[x_{min}, x_{max}] \times [y_{min}, y_{max}]$ (cualesquiera otros argumentos adicionales se tratan como en `plot3d`)

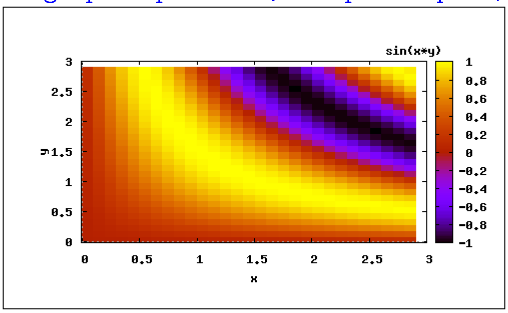
Gráficos de contornos.

Maxima

Esto traza un gráfico de densidad de la función $f(x,y) = \text{sen}(xy)$ en el rectángulo $[0,3] \times [0,3]$.

```
(%i1) wxplot3d(sin(x*y),[x,0,3],[y,0,3],
        [gnuplot_preamble,"set pm3d map"]);
```

(%t1)



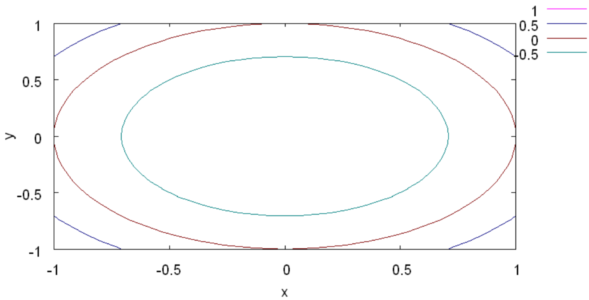
(%o1)

Maxima

He aquí un gráfico de contorno de $f(x,y) = x^2 + y^2 - 1$.

```
(%i2) contour_plot(x^2+y^2-1,[x,-1,1],[y,-1,1]);
```

(%t2)



(%o2)

14.8 Gráficos animados

También es posible crear animaciones, aunque únicamente en el entorno de *wxMaxima*. Para tal fin se utiliza la función `with_slider`[4] con la cual se genera un gráfico idéntico al que se genera con `wxplot2d`, sin embargo dicho gráfico puede ser animado selecionándolo y pulsando luego el botón Comenzar animación, del cuadro de controles, de la barra de herramientas.

Figura 14.2: Cuadro de controles, ubicado en la barra de herramientas, para la animación de gráficos.

`with_slider`	función para animar gráficos de `wxplot2d`

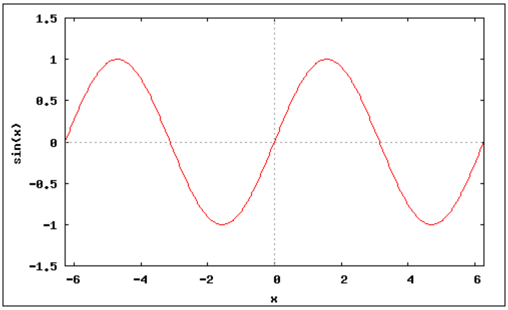
Obtención de gráficos animados.

[4]Es importante señalar que adicionalmente estan incluidas las funciones `with_slider_draw` y `with_slider_draw3d` relacionadas con las funciones `draw` y `draw3d` del paquete `draw`.

Maxima

Esto genera un gráfico listo para ser animado. Para conseguir la animación primero se selecciona el gráfico y luego se pulsa el botón ▷, del cuadro de controles, y automáticamente ésta es generada. Para detenerla basta pulsar el botón □, del mismo cuadro. También es posible navegar a través de cada cuadro de la animación con tan sólo arrastrar a voluntad el botón, después de haber seleccionado el gráfico.

```
(%i1) with_slider(a,[0,1,2,3,4,5],sin(x+a),
        [x,-2*%pi,2*%pi],[y,-1.5,1.5],[color,red]);
```

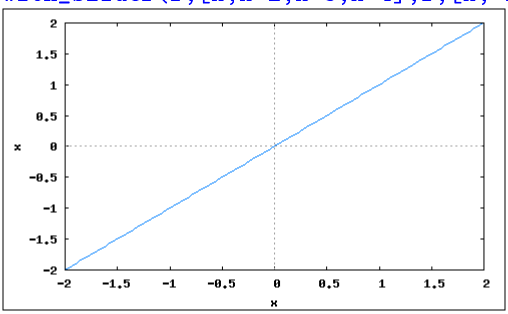
(%t1) 

(%o1)

Maxima

He aquí otro gráfico listo para animar.

```
(%i2) with_slider(f,[x,x^2,x^3,x^4],f,[x,-2,2]);
```

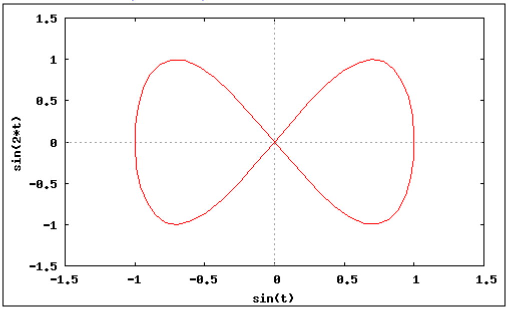
(%t2) 

(%o2)

Maxima

Este es un gráfico más listo para animar.

```
(%i3) with_slider(a,[0,1,2,3,4,5],
        [parametric,sin(t),sin(2*(t+a)),[t,0,2*%pi]],
        [x,-1.5,1.5],[y,-1.5,1.5],[color,red],
        [nticks,50]);
```

(%t3) 

(%o3)

CAPÍTULO 15

Utilidades de los menúes de *wxMaxima*

wxMaxima contiene menús y cuadros de diálogo para realizar las tareas más rutinarias, facilitando así al usuario novel el explorar las características de *Maxima*.

En este capítulo repasaremos las principales opciones de los menúes incorporados en la barra de menúes de *wxMaxima*.

15.1 El menú Archivo

El menú Archivo incluye las opciones (ver Fig. 15.1):

Abrir Permite abrir una sesión previamente guardada,

Abrir sesión reciente Permite abrir una sesión recientemente guardada,

Guardar Permite guardar la sesión actual,

Guardar como Permite guardar, con otro nombre, la sesión actual,

Cargar paquete Permite inicializar un paquete indicando su ubicación,

Archivo por lotes Permite inicializar y traducir un paquete, de extensión mac, indicando su ubicación,

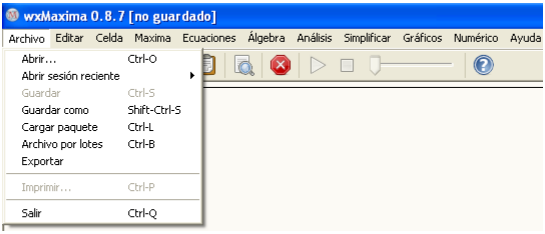


Figura 15.1: Despliegue de opciones del menú Archivo.

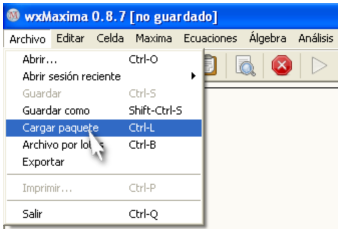


Figura 15.2: Elección de la opción **Cargar paquete**.

Exportar Permite exportar archivos al LaTeX y HTML (ver Cap. 18),

Imprimir Permite realizar la impresión del cuaderno actual,

Salir Finaliza la sesión actual.

Por ejemplo, la figura 15.2 muestra la elección de la opción **Cargar paquete** (también puede hacerse presionando en simultáneo Ctrl+L). Después de elegir tal opción aparece el cuadro de la figura 15.3 en el que se elegirá el paquete a cargar. Luego de indicar el tipo (que puede ser mac o lisp) y el nombre del paquete (teniendo en cuenta la ruta específica de donde esta guardado el mismo) se da clic en el botón $\boxed{\text{Abrir}}$. En nuestro caso elegiremos el paquete `table.lisp`.[1]

[1] El autor de éste paquete es Ziga Lenarcic y puede descargarse desde: `http://sourceforge.net/apps/phpbb/maxima/viewtopic.php?f=3&t=5&sid=716969b0736cc8416d959fdc5aded01e`

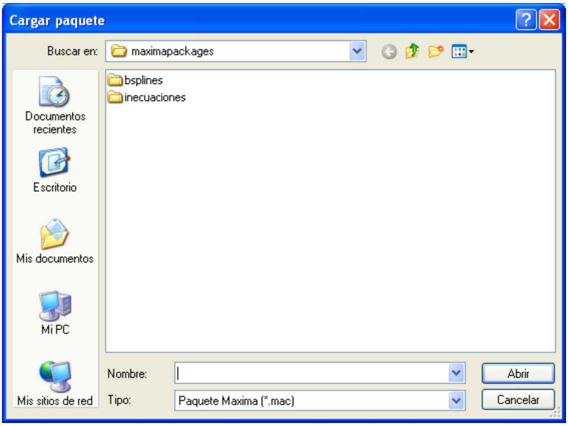


Figura 15.3: Cuadro para elegir el paquete a cargar.

Maxima

He aquí la sentencia que aparece después de haber elegido el paquete **`table.lisp`** y presionar Abrir.

(%i1) `load("D:/maximapackages/table.lisp")$`

Maxima

Ahora es posible usar la función **`table`** para generar una lista de los valores $\operatorname{sen}(t)$, con t variando de 0 a 2π y un incremento de $\frac{\pi}{4}$.

(%i2) `table(sin(t),[t,0,2*%pi,%pi/4]);`

(%o2) $[0, \frac{1}{\sqrt{2}}, 1, \frac{1}{\sqrt{2}}, 0, -\frac{1}{\sqrt{2}}, -1, -\frac{1}{\sqrt{2}}, 0]$

Maxima

En este caso se asume que el límite inferior es 1.

(%i3) `table(i^2,[i,5]);`

(%o3) $[1, 4, 9, 16, 25]$

15.2 El menú Editar

El menú Editar incluye las opciones (ver Fig. 15.4):

Deshacer Deshace cualquier entrada previa de una celda actual,

Cortar Corta cualquier entrada previa,

Copiar Copia cualquier entrada previa,

Copiar como texto Copia la selección de una salida (o un conjunto de celdas) como texto,

Copiar como LaTeX Copia la selección de una salida (o un conjunto de celdas) como código LaTeX,

Copiar como imagen Copia la selección de una salida (o un conjunto de celdas) como imagen,

Pegar Permite pegar el contenido del portapapeles en el cuaderno actual,

Buscar Permite buscar una expresión en el cuaderno actual,

Seleccionar todo Selecciona el contenido de todas las celdas del cuaderno actual actual,

Guardar selección en imagen Guarda la selección de una salida (o un conjunto de celdas) como imagen,

Ampliar Permite ampliar el tamaño de los caracteres del cuaderno actual,

Disminuir Permite disminuir el tamaño de los caracteres del cuaderno actual,

Establecer aumento Permite establecer el aumento del tamaño de los caracteres del cuaderno actual,

Pantalla completa Permite realizar una vista en pantalla completa del cuaderno actual,

Preferencias Permite editar la configuración de las opciones y estilos (ver Sec. 3.3).

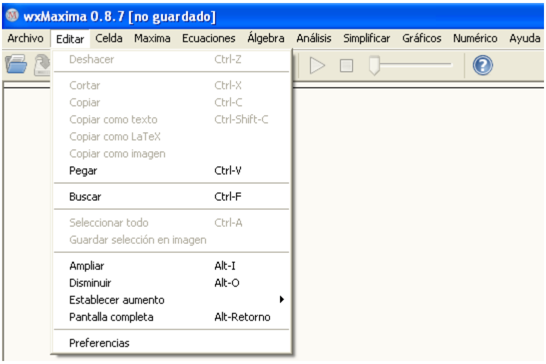


Figura 15.4: Despliegue de opciones del menú Editar.

A modo de ejemplo, seleccionaremos las celdas del cuaderno de la figura 15.5 luego elegimos la opción **Copiar como LaTeX** del menú Editar (ver Fig. 15.6). Al pegar el contenido del portapapeles en algún editor de LaTeX, y compilar, se obtiene la salida que se muestra a continuación:

Salida obtenida, con el código guardado, mediante un editor de LaTeX

(%i1) `load("D:/maximapackages/table.lisp")$`

(%i2) `table(sin(t),[t,0,2*%pi,%pi/4]);`

$$(\%o2)\quad [0,\frac{1}{\sqrt{2}},1,\frac{1}{\sqrt{2}},0,-\frac{1}{\sqrt{2}},-1,-\frac{1}{\sqrt{2}},0]$$

(%i3) `table(i^2,[i,5]);`

$$(\%o3)\quad [1,4,9,16,25]$$

En cambio, si se elige la opción **Guardar selección en imagen** del mismo menú, y se siguen las respectivas indicaciones, se obtiene un archivo png (ver Fig. 15.7), que puede insertarse luego en cualquier documento.

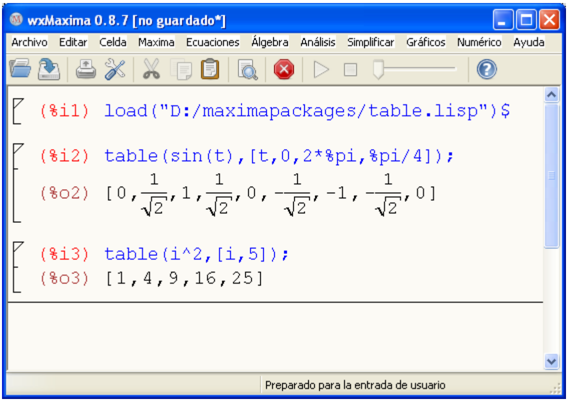


Figura 15.5: Cuaderno para el ejemplo de esta sección.

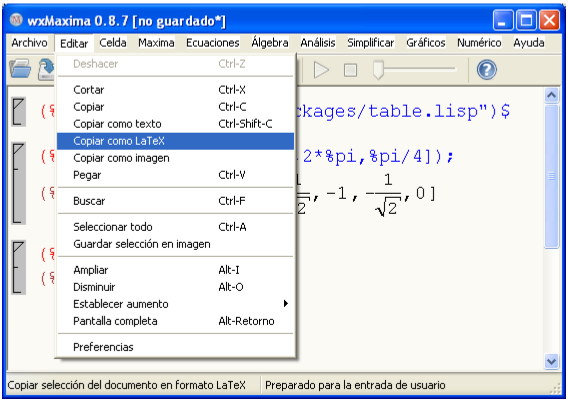


Figura 15.6: Eligiendo la opción opción **Copiar como LaTeX** del menú Editar.

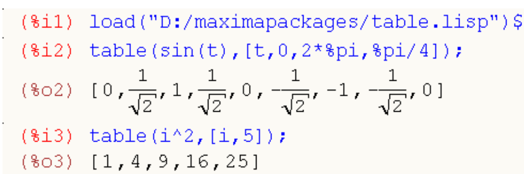


Figura 15.7: Imagen guardada mediante la opción **Guardar selección en imagen** del menú Editar.

15.3 El menú Celda

El menú Editar incluye las opciones (ver Fig. 15.8):

Evaluar celdas Evalúa las celdas seleccionadas del cuaderno actual,

Evaluar todas las celdas Evalúa todas las celdas del cuaderno actual,

Borrar todos los resultados Borrar todas las salidas del cuaderno actual,

Copiar entrada anterior Copia la entrada previa en el cuaderno actual,

Autocompletar Muestra un conjunto de posibles términos que completarían la entrada actual,

Mostrar plantilla Muestra un conjunto de posibles plantillas que completarían la entrada actual,

Nueva celda de entrada Inserta una celda de entrada en el cuaderno actual(ver Sec. 3.2),

Nueva celda de texto Inserta una celda de texto el cuaderno actual (ver Sec. 3.2),

Nueva celda de subsección Inserta una celda de subsección el cuaderno actual (ver Sec. 3.2),

Nueva celda de sección Inserta una celda de sección el cuaderno actual (ver Sec. 3.2),

Nueva celda de título Inserta una celda de título el cuaderno actual (ver Sec. 3.2),

Insertar salto de página Inserta un salto de página en el cuaderno actual,

Insertar imagen Permite insertar una imagen en el cuaderno actual (cabe destacar que tal imagen no prevalece al guardar el cuaderno),

Instrucción anterior Permite navegar hacia atrás entre las instrucciones dadas en el cuaderno actual (al estilo MatLab),

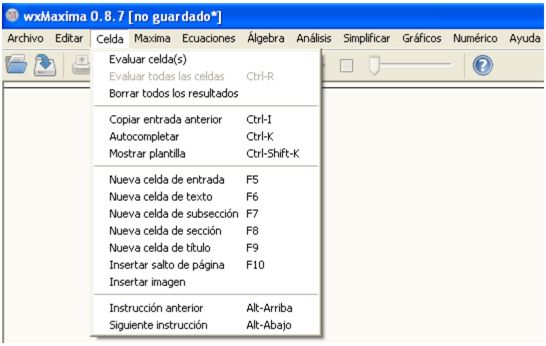


Figura 15.8: Despliegue de opciones del menú Celda.

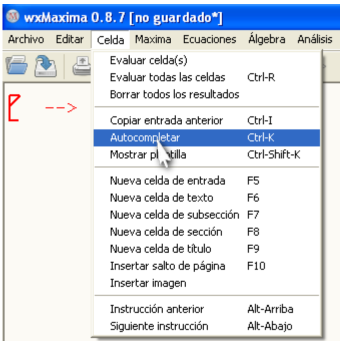


Figura 15.9: Activando la opción **Autocompletar**.

Siguiente instrucción Permite navegar hacia adelante entre las instrucciones dadas en el cuaderno actual (al estilo MatLab).

Por ejemplo, si se desea calcular $\int \cos(x)\, dx$ y no se recuerda el comando específico, pero si se recuerda que empieza con `int`, entonces puede usarse la opción **Autocompletar** para obtener una lista de funciones entre las que figura: `integrate` (ver Figs. 15.9 y 15.10). En cambio, si se elige la opción **Mostrar plantilla**, aparecerá una lista de plantillas (ver).

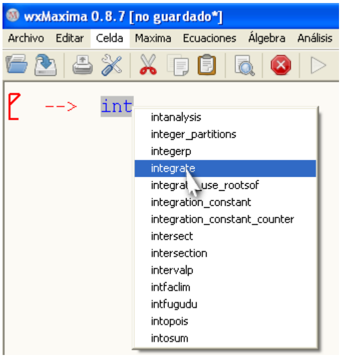


Figura 15.10: Lista de funciones que empiezan con `int`.

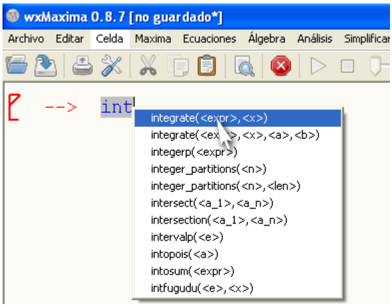


Figura 15.11: Lista de plantillas que empiezan con `int`.

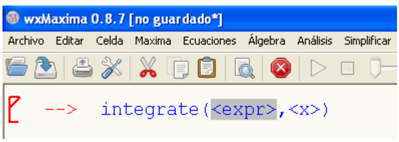


Figura 15.12: Plantilla para calcular una integral indefinida.

15.4 El menú Maxima

El menú Maxima incluye las opciones (ver Fig. 15.13):

Paneles Permite mostrar u ocultar paneles para *Matemáticas generales*, *Estadística*, *Historial*, *Insertar celda* y *Barra de Herramientas*,

Interrumpir Interrumpe el cálculo actual,

Reiniciar Maxima Reinicia *Maxima* sin salir de *wxMaxima* [2],

Limpiar memoria Elimina todas las asignaciones de todas las variables,

Añadir ruta Especifica listas de directorios en los que deben buscar la funciones `load`, `demo` y algunas otras,

Mostrar funciones Genera una lista que contiene los nombres de las funciones definidas por el usuario,

Mostrar definición Permite obtener la definición de una función específica,

Mostrar variables Genera una lista de todas las variables que el usuario ha creado,

Borrar función Permite borrar todas o algunas de las funciones definidas por el usuario,

Borrar variable Permite borrar todas o algunas de las variables creadas por el usuario,

Conmutar pantalla de tiempo Permite que el tiempo de cálculo y el tiempo de retardo se impriman junto con la salida de cada expresión,

Cambiar pantalla 2D Permite seleccionar el algoritmo de salida matemática (ver Sec. 4.4),

Mostrar formato TeX Devuelve la expresión actual en un formato apropiado para para ser incorporado a un documento basado en TeX (ver Sec. 18.1).

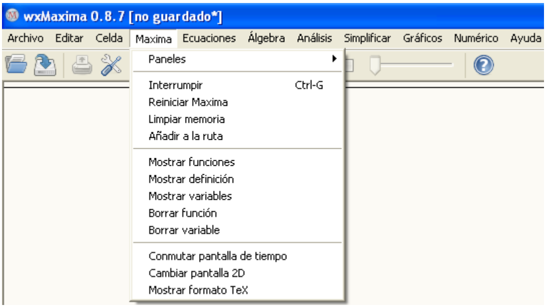


Figura 15.13: Despliegue de opciones del menú Maxima.

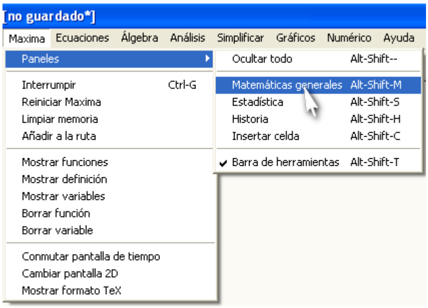


Figura 15.14: Eligiendo el panel *Matemáticas Generales* de la opción **Paneles** del menú Maxima.

Por ejemplo para desplegar el panel *Matemáticas generales* elegimos dicho panel de la opción **Paneles** del menú Maxima (ver Figs. 15.14 y 15.15)

15.5 El menú Ecuaciones

El menú Ecuaciones incluye las opciones (ver Fig. 15.16):

Resolver Permite resolver una ecuación algebraica,

Resolver (to_poly) Permite resolver una ecuación,

[2]Eso es útil cuando, por ejemplo, generamos sin querer un bucle en la ejecución de *Maxima*.

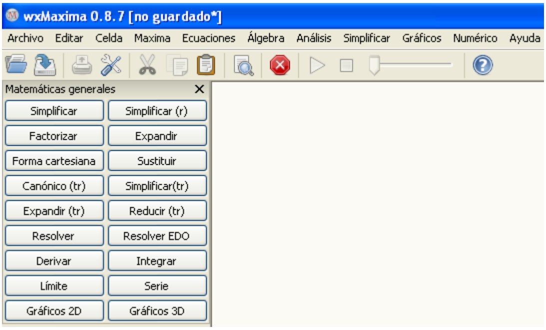


Figura 15.15: Panel *Matemáticas Generales* desplegado.

Calcular raíz Permite aproximar una raíz en un intervalo dado,

Raíces de un polinomio Aproxima las raíces reales y complejas de un polinomio ingresado en la celda de entrada actual,

Raíces reales grandes de un polinomio Aproxima las raíces reales y complejas, en formato bigfloat, de un polinomio ingresado en la celda de entrada actual,

Resolver sistema lineal Permite resolver un sistema de ecuaciones lineales,

Resolver sistema algebraico Permite resolver un sistema de ecuaciones algebraicas,

Eliminar variable Elimina variables de ecuaciones (o de expresiones que se supone valen cero) tomando resultantes sucesivas,

Resolver EDO Permite resolver ecuaciones diferenciales ordinarias de primer y segundo orden,

Problema de valor inicial (1) Permite añadir condiciones de valor inicial a la solución de una ecuación diferencial ordinaria de primer orden,

Problema de valor inicial (2) Permite añadir condiciones de valor inicial a la solución de una ecuación diferencial ordinaria de segundo orden,

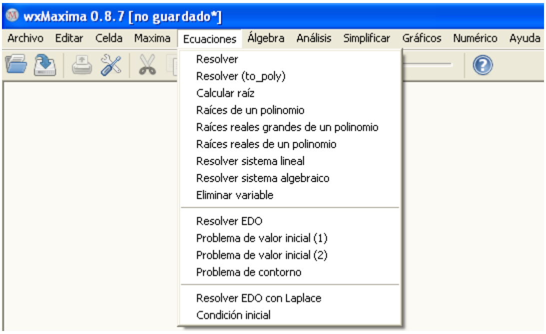


Figura 15.16: Despliegue de opciones del menú Ecuaciones.

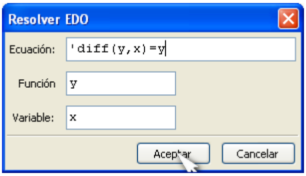


Figura 15.17: Uso de la opción **Resolver EDO** para dar solución a la ecuación diferencial $y' = y$.

Problema de contorno Permite añadir condiciones de frontera a la solución de una ecuación diferencial ordinaria de segundo orden,

Resolver EDO con Laplace Permite resolver sistemas de ecuaciones diferenciales ordinarias lineales utilizando la transformada de Laplace,

Condición inicial Permite asignar un determinado valor a cierta expresión dada en un punto específico.

Por ejemplo, para resolver el problema de valor inicial $y' = y$, $y(0) = 1$; podemos usar las opciones **Resolver EDO** y **Problema de valor inicial (1)** tal como se muestra en las figuras .

Maxima

Resultados obtenidos al seguir los procesos indicados en las figuras 15.17 y 15.18.

```
(%i1) ode2('diff(y,x)=y,y,x);
```

(%o1) $y = \%c * \%e^{x}$

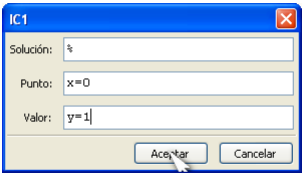


Figura 15.18: Uso de la opción **Problema de valor inicial (1)** para añadir la condición inicial $y(0) = 1$ a la ecuación diferencial de la figura 15.17 (aquí se asume que tal ecuación figura en la entrada actual).

```
(%i2) ic1(%,x=0,y=1);
```

(%o2) $y = \%e^x$

15.6 El menú Álgebra

El menú Álgebra incluye las opciones (ver Fig. 15.19):

Generar matriz Permite generar una matriz,

Generar matriz a partir de expresión Permite generar una matriz a partir de una expresión en términos de i, j,

Introducir matriz Permite generar una matriz introduciendo sus términos en tiempo real,

Invertir matriz Devuelve la matriz inversa de una matriz actualmente definida,

Polinomio característico Permite generar el polinomio característico de cierta matriz especificando la variable,

Determinante Devuelve el determinante de la matriz actual,

Valores propios Devuelve los valores propios de la matriz actual,

Vectores propios Devuelve los vectores propios de la matriz actual,

Matriz adjunta Devuelve la matriz adjunta de la matriz actual,

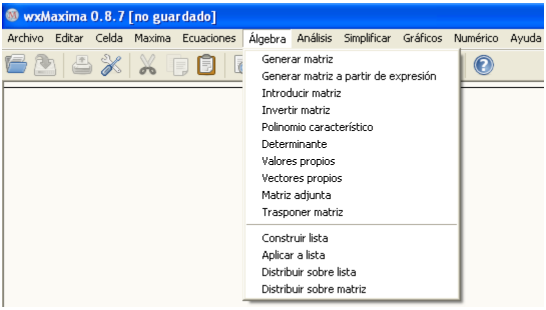


Figura 15.19: Despliegue de opciones del menú Álgebra.

Trasponer matriz Devuelve la traspuesta de la matriz actual,

Construir lista Permite construir una lista en términos de cierta expresión que depende de una variable,

Aplicar a lista Permite aplicar un operador a una lista,

Distribuir sobre lista Permite aplicar una función a cada uno de los elementos de una lista,

Distribuir sobre matriz Permite aplicar una función a cada uno de los elementos de una matriz.

Maxima

Aquí se define una función anónima.

```
(%i1) lambda([i,j],1/(i+j-1))$
```

Maxima

A continuación, usando la opción **Generar matriz** y llenando los respectivos datos (ver Fig. 15.20) se obtiene:

```
(%i2) A: genmatrix(%,3,3);
```

(%o2) $$\begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{3} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \end{pmatrix}$$

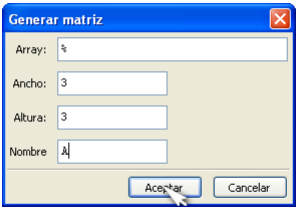


Figura 15.20: Llenando los datos del cuadro asociado a la opción **Genarar matriz**.

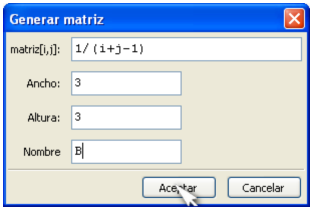


Figura 15.21: Llenando los datos del cuadro asociado a la opción **Generar matriz a partir de expresión**.

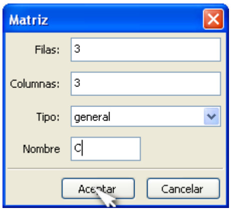


Figura 15.22: Llenando los datos del cuadro asociado a la opción **Introducir matriz**.

Maxima

No obstante, usando la opción **Generar matriz a partir de expresión** y llenando los respectivos datos (ver Fig. 15.21) se obtiene:

```
(%i3) B:genmatrix(lambda([i,j],1/(i+j-1)),3,3);
```

(%o3) $$\begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{3} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \end{pmatrix}$$

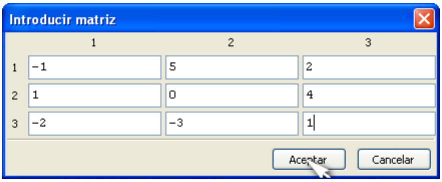


Figura 15.23: Introduciendo los elementos de la matriz, previamente creada, en tiempo real.

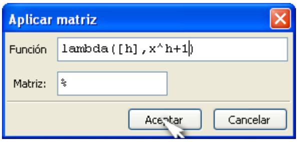


Figura 15.24: Ingresando la función anónima que se desea aplicar a cada uno de los elementos de la matriz actual.

Maxima

Eligiendo la opción **Introducir matriz** y llenando los datos respectivos para definir una matriz (15.22) es posible introducir, en tiempo real, cada uno de los elementos de dicha matriz (15.23).

```
(%i4) C: matrix(
        [-1,5,2],
        [1,0,4],
        [-2,-3,1]
      );
```

$$(\%o4) \begin{pmatrix} -1 & 5 & 2 \\ 1 & 0 & 4 \\ -2 & -3 & 1 \end{pmatrix}$$

Maxima

Si se quiere aplicar una función a cada uno de los elementos de la matriz actual, por ejemplo la matriz C, puede usarse la opción **Distribuir sobre matriz** (15.24).

```
(%i5) matrixmap(lambda([h],x^h+1),%);
```

(%o5) $$\begin{pmatrix} \frac{1}{x}+1 & x^5+1 & x^2+1 \\ x+1 & 2 & x^4+1 \\ \frac{1}{x^2}+1 & \frac{1}{x^3}+1 & x+1 \end{pmatrix}$$

15.7 El menú Análisis

El menú Análisis incluye las opciones (ver Fig. 15.25):

Integrar Permite calcular la integral indefinida o definida de una expresión,

Integración Risch Permite calcular la integral indefinida de una expresión utilizando el caso trascendental del algoritmo de Risch,

Cambiar variable Permite efectuar un cambio de variable en una integral indefinida o definida de un expresión,

Derivar Permite calcular la derivada n-ésima de una expresión,

Calcular límite Permite calcular el límite de una expresión,

Calcular mínimo Permite encontrar una solución aproximada para el problema de minimización sin restricciones de una función objetivo dada,

Calcular serie Permite expandir una expresión en un desarrollo de Taylor,

Aproximación de Padé Permite obtener la lista de todas las funciones racionales que tienen el desarrollo de Taylor dado, en las que la suma de los grados del numerador y denominador es menor o igual que el nivel de truncamiento de la serie de potencias,

Calcular suma Permite calcular la suma de los valores de una expresión según la variación del respectivo índice,

Calcular producto Permite calcular el producto de los valores de una expresión según la variación del respectivo índice,

Transformada de Laplace Permite calcular la transformada de Laplace de una expresión dada,

Transformada inversa de Laplace Permite calcular la transformada inversa de Laplace de una expresión dada,

Máximo común divisor Permite obtener el MCD de dos polinomios dados,

Mínimo común múltiplo Permite obtener el mcm de dos polinomios dados,

Dividir polinomios Permite calcular el cociente y el resto de la división de dos polinomios dados,

Fracciones simples Expande una expresión en fracciones parciales,

Fracción continua Permite obtener una fracción continua a partir de la representación en formato lista de la misma.

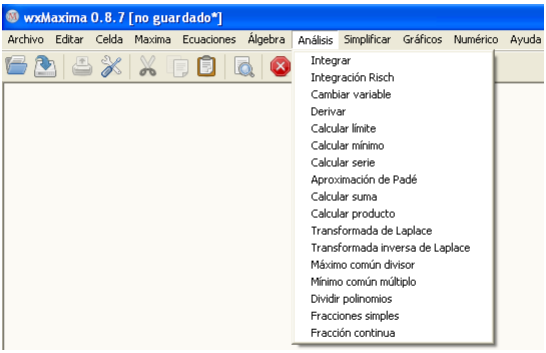


Figura 15.25: Despliegue de opciones del menú Análisis.

Maxima

Aquí se introduce una integral sin evaluar.

```
(%i1) 'integrate(sin(x)^2*cos(x),x);
```

(%o1) $\int \cos(x)\, \sin(x)^2 dx$

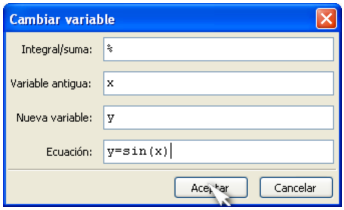


Figura 15.26: Llenando los datos del cuadro asociado a la opción **Cambiar variable**.

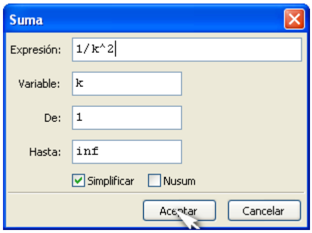


Figura 15.27: Llenando los datos del cuadro asociado a la opción **Calcular suma**.

Maxima

Después de usar la opción **Cambiar variable** (15.26) se obtiene el siguiente resultado:

```
(%i2) changevar(%,y=sin(x),y,x);
```

(%o2) $\int y^2 dy$

Maxima

Después de usar la opción **Calcular suma** (15.27) se obtiene el siguiente resultado:

```
(%i3) sum(1/k^2,k,1,inf),simpsum;
```

(%o3) $\frac{\pi^2}{6}$

15.8 El menú Simplificar

El menú Simplificar incluye las opciones (ver Fig. 15.28):

Simplificar expresión Simplifica una expresión actual y todas sus subexpresiones, incluyendo los argumentos de funciones no racionales,

Simplificar radicales Simplifica una expresión actual, que puede contener logaritmos, exponenciales y radicales, convirtiéndola a una forma canónica,

Factorizar expresión Factoriza una expresión actual, que puede contener cualquier número de variables o funciones, en factores irreducibles respecto de los enteros,

Factorizar complejo Factoriza un polinomio actual sobre los enteros gaussianos (un entero gaussiano es de la forma $a + ib$ donde a y b son números enteros),

Expandir expresión Expande la expresión actual,

Expandir logaritmos Activa el valor *super* de la variable opcional `logexpand`,

Contraer logaritmos Realizar contracciones en una expresión actual logarítmica,

Factoriales y gamma Permite mostrar u ocultar paneles para *Convertir a factoriales*, *Convertir a gamma*, *Simplificar factoriales* y *Combinar factoriales*,

Simplificación trigonométrica Permite mostrar u ocultar paneles para *Simplificar trigonometría*, *Reducir trigonometría*, *Expandir trigonometría* y *Forma canónica*,

Simplificación compleja Permite mostrar u ocultar paneles para *Convertir a forma cartesiana*, *Convertir a forma polar*, *Calcular parte real*, *Calcular parte imaginaria*, *Demoivre* y *Exponencializar*,

Sustituir Permite realizar sustituciones en expresiones,

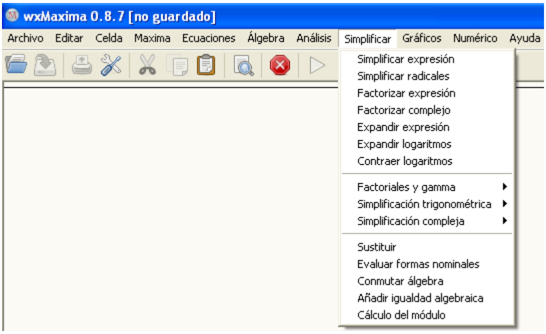


Figura 15.28: Despliegue de opciones del menú Simplificar.

Evaluar formas nominales Permite evaluar formas cuya evaluación a sido evitada por el operador comilla simple ’,

Conmutar álgebra Permite conmutar el valor actual de la variable opcional `algebraic` para que se pueda hacer o no la simplificación de enteros algebraicos,

Añadir igualdad algebraica Permite añadir al anillo de enteros algebraicos conocidos por *Maxima* los elementos que son soluciones de los polinomios ingresados,

Cálculo del módulo Permite indicar el módulo mediante el cual se realizan las operaciones con números racionales.

Maxima

Usando la opción **Añadir igualdad algebraica** e ingresando la respectiva igualdad en el cuadro asociado a dicha opción (15.28) obtenemos:

```
(%i1) tellrat(a+1/a=sqrt(3));
```

(%o1) $[a^2 - \sqrt{3}\,a + 1]$

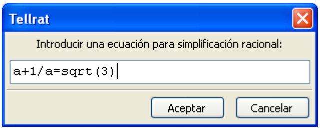


Figura 15.29: Llenando los datos del cuadro asociado a la opción **Añadir igualdad algebraica**.

Maxima

Ahora, es preciso incluir la siguiente sentencia para poder arribar al resultado que se desea mostrar.

```
(%i2) algebraic:true;
(%o2) true
```

Maxima

Finalmente, averiguamos el valor para $a^4+\frac{1}{a^4}$. Téngase presente que $a+\frac{1}{a}=\sqrt{3}$ implica : $a^4+\frac{1}{a^4}=\left(\left(a+\frac{1}{a}\right)^2-2\right)^2-2=(\sqrt{3}^{\,2}-2)^2-2=-1$.

```
(%i3) ratsimp(a^4+1/a^4);
(%o3) -1
```

15.9 El menú Gráficos

El menú Gráficos incluye las opciones (ver Fig. 15.30):

Gráficos 2D Permite realizar gráficos bidimensionales,

Gráficos 3D Permite realizar gráficos tridimensionales,

Formato de gráficos Permite establecer el formato de los gráficos.

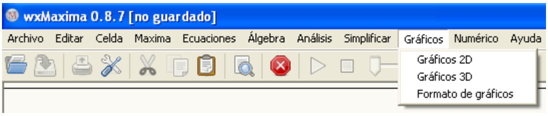


Figura 15.30: Despliegue de opciones del menú Gráficos.

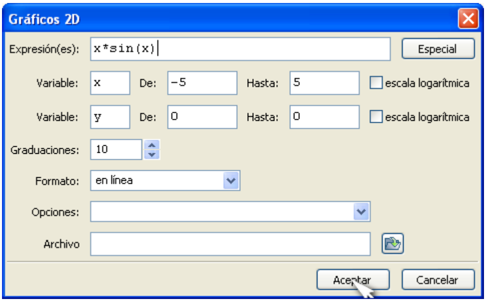


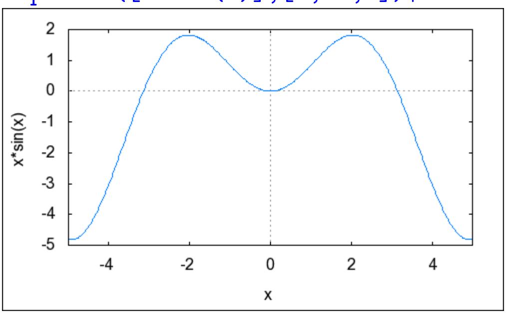
Figura 15.31: Llenando los datos del cuadro asociado a la opción Gráficos.

Maxima

Llenando los datos del cuadro asociado a la opción **Gráficos 2D** (ver Fig. 15.31) se genera la sentencia adecuada para obtener la gráfica de $x \to x * \operatorname{sen}(x)$.

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

(%t1)



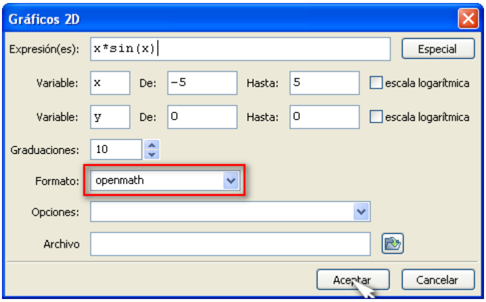


Figura 15.32: Eligiendo, adicionalmente, el formato **`openmath`**.

Maxima

Si se elije el formato **`openmath`** (ver Fig. 15.32) se genera la sentencia adecuada para obtener la gráfica de $x \to x * \mathrm{sen}(x)$ con el programa gráfico **`openmath`** (ver Fig. 15.33).

```
(%i2) wxplot2d([x*sin(x)],[x,-5,5])$
```

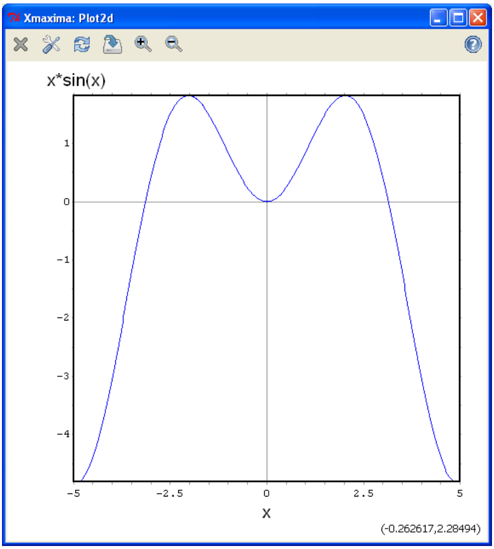


Figura 15.33: Gráfico obtenido a partir de `(%i2)` .

15.10 El menú **Numérico**

El menú Numérico incluye las opciones (ver Fig. 15.34):

Conmutar salida numérica Activa la variable `float` para obtener resultado numéricos,

A real Devuelve el valor numérico de la expresión actual,

A real grande (bigfloat) Convierte todos los números y funciones numéricas a números decimales de punto flotante grandes,

Establecer precisión Permite establecer el número de dígitos significativos en la aritmética con números decimales de punto flotante grandes .

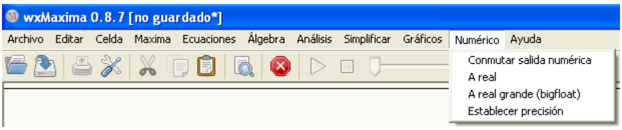


Figura 15.34: Despliegue de opciones del menú Numérico.

Maxima

Por defecto los resultados son simbólicos.

```
(%i1) sqrt(2)+sqrt(3);
```

(%o1) $\sqrt{3}+\sqrt{2}$

Maxima

Después de activar la opción **Conmutar salida numérica** se obtiene el código adecuado para obtener resultados en forma numérica.

```
(%i2) if numer#false then numer:false else
      numer:true;
```

(%o2) *true*

Maxima

Ahora se obtienen resultados numéricos.

```
(%i3) sqrt(2)+sqrt(3);
(%o3) 3.146264369941973
```

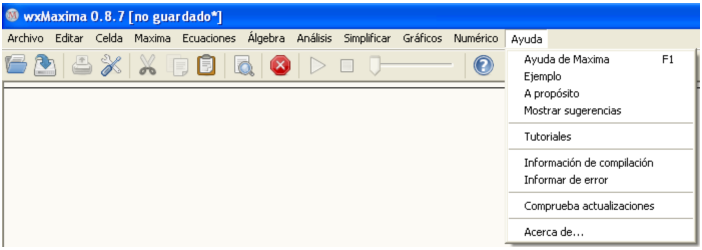


Figura 15.35: Despliegue de opciones del menú Ayuda.

15.11 El menú Ayuda

El menú Ayuda incluye las opciones (ver Fig. 15.35):

Ayuda de Maxima Muestra los libros de ayuda de *Maxima* que estén disponibles en el sistema. Estos libros se pueden navegar de forma interactiva y en cada uno de ellos se pueden realizar búsquedas de palabras clave,

Ejemplo Muestra un ejemplo de uso de cualquier función de *Maxima*,

A propósito Muestra funciones de *Maxima* similares a una palabra,

Mostrar sugerencias Muestra una idea relacionada con el uso de *wxMaxima*,

Tutoriales Carga la página oficial de *wxMaxima*,

Información de compilación Imprime un resumen de los parámetros que se usaron para construir la versión de *Maxima* que se está usando,

Informar de error Imprime las versiones de *Maxima* y de *Lisp* y proporciona un enlace a la página web sobre informe de fallos del proyecto *Maxima*,

Comprueba actualizaciones Permite comprobar actualizaciones de *wxMaxima*,

Acerca de... Muestra información técnica relacionada con el funcionamiento del programa.

CAPÍTULO 16

Gráficos con draw

El paquete `draw` se distribuye conjuntamente con *Maxima* y constituye una interfaz que comunica de manera muy eficiente *Maxima* con *Gnuplot*. Este paquete incorpora una considerable variedad de funciones y opciones que permiten obtener la representación de un amplio número de objetos gráficos bidimensionales y tridimensionales.

Para poder utilizar el paquete `draw` es preciso cargarlo en la memoria, y para ello se utiliza la función `load` (ver sección 3.7).

`load(draw) $`	"carga" (inicializa) el paquete `draw`
`draw`(*gr2d*, ..., *gr3d*, ..., *opciones*)	representa gráficamente una serie de escenas; sus argumentos son objetos *gr2d* y/o *gr3d*, junto con algunas opciones. Por defecto, las escenas se representan en una columna
`draw2d`(*opciones*, *objeto_gráfico*, ...)	esta función es un atajo para `draw`(*gr2d*(*opciones*, *objeto_gráfico*, ...))
`draw3d`(*opciones*, *objeto_gráfico*, ...)	esta función es un atajo para `draw`(*gr3d*(*opciones*, *objeto_gráfico*, ...))

Inicialización del paquete `draw` y descripción de sus tres funciones principales.

16.1 Objetos gráficos bidimensionales

`points`$([[x_1, y_1], [x_2, y_2], \ldots])$	puntos bidimensionales
`points`$([x_1, x_2, \ldots], [y_1, y_2, \ldots])$	puntos bidimensionales
`points`$([y_1, y_2, \ldots])$	equivale a `points`$([[1, y_1], [2, y_2], \ldots])$
`points(matrix`$([x_1, y_1], [x_2, y_2], \ldots)$	puntos bidimensionales
`bars`$([x_1, h_1, w_1], \ldots)$	dibuja barras verticales centradas en x_i, de alturas h_i y anchos w_i
`polygon`$([[x_1, y_1], [x_2, y_2], \ldots])$	polígono
`polygon`$([x_1, x_2, \ldots], [y_1, y_2, \ldots])$	polígono
`rectangle`$([x_1, y_1], [x_2, y_2])$	rectángulo de vértices opuestos (x_1, y_1) y (x_2, y_2)
`quadrilateral`$([x_1, y_1], [x_2, y_2], [x_3, y_3], [x_4, y_4])$	cuadrilátero
`triangle`$([x_1, y_1], [x_2, y_2], [x_3, y_3])$	triángulo
`image`$(im, x_0, y_0, width, height)$	imagen im en la región rectangular desde el vértice (x_0, y_0) hasta $(x_0 + width, y_0 + height)$
`vector`$([p_1, p_2], [v_1, v_2])$	vector (v_1, v_2) con punto de aplicación (p_1, p_2)
`label`$([cadena, x, y], \ldots)$	etiquetas para gráficos bidimensionales
`ellipse`$(x_c, y_c, a, b, ang_1, ang_2)$	sector elíptico de centro (x_c, y_c) con semiejes horizontal y vertical a y b, respectivamente, comenzando en el ángulo ang_1 y trazando un arco de amplitud igual al ángulo ang_2

Principales objetos gráficos bidimensionales incorporados en `draw`.

`explicit`$(f(x), x, x_{min}, x_{max})$	función explícita f cuya variable x toma valores desde x_{min} hasta x_{max}
`implicit`$(E(x, y), x, x_{min}, x_{max}, y, y_{min}, y_{max})$	expresión implícita E a ser representada en el rectángulo $[x_{min}, x_{max}] \times [y_{min}, y_{max}]$
`parametric`$(f_x, f_y, t, t_{min}, t_{max})$	curva paramétrica bidimensional, cuyo parámetro t toma valores desde t_{min} hasta t_{max}
`polar`$(r(\theta), \theta, \theta_{min}, \theta_{max})$	función polar r cuya variable θ toma valores desde θ_{min} hasta θ_{max}
`region`$(expr, x, x_{min}, x_{max}, y, y_{min}, y_{max})$	región del plano definida por desigualdades a ser representada en el rectángulo $[x_{min}, x_{max}] \times [y_{min}, y_{max}]$

Principales objetos gráficos bidimensionales incorporados en `draw`.

Las funciones `draw`, `draw2d` y `draw3d` devuelven las salidas gráficas en una ventana de *Gnuplot*, aparte de la ventana de trabajo actual. No obstante el entorno gráfico *wxMaxima* permite utilizar las funciones `wxdraw`, `wxdraw2d` y `wxdraw3d` que sí devuelven las salidas en el cuaderno de trabajo actual. Debe tenerse presente que al utilizar la función `wxdraw3d`, el punto de vista de la gráfica obtenida no puede ser cambiado en tiempo real.

En este capítulo se van a mostrar los resultados mediante las funciones `wxdraw`, `wxdraw2d` y `wxdraw3d`, pero todos los ejemplos mostrados pueden ser ejecutados, sin ningún problema, con las funciones `draw`, `draw2d` y `draw3d`, según sea el caso.

Maxima

Aquí se genera una lista de listas. Los elementos de la misma, generados aleatoriamente, representan las coordenadas de puntos del plano.

```
(%i1) p:create_list([random(20),random(50)],k,1,10);
```

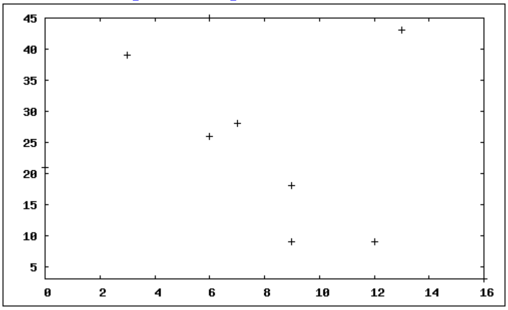
(%o1) $[[9,9],[3,39],[6,26],[16,3],[0,21],[12,9],[9,18],[6,45],[13,43],[7,28]]$

Maxima

Después de aplicar la función **points** a la variable p, en la que se han almacenado los puntos, se obtiene un objeto gráfico el cual puede graficarse con **draw2d**.

```
(%i2) wxdraw2d( points(p) );
```

(%t2)



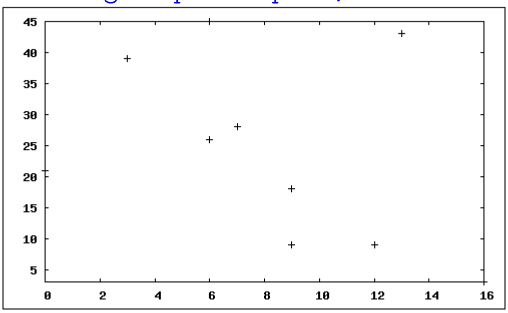
(%o2)

Maxima

Con **gr2d** y **draw** siempre se obtendrá el mismo resultado que con **draw2d**.

```
(%i3) wxdraw( gr2d(points(p)) );
```

(%t3)



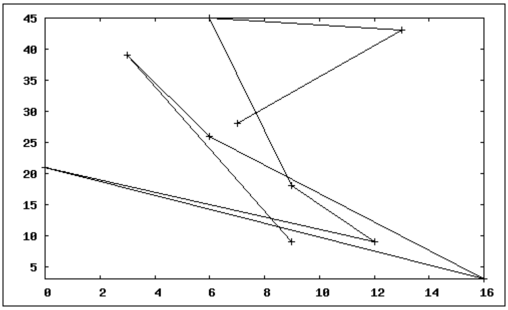
(%o3)

`draw` no cuenta con una función específica, como *line*, para definir el objeto gráfico línea; simplemente se asigna el valor `true` a la opción `points_joined` para unir los puntos dados mediante segmentos.

Maxima

Gráfica de los puntos p unidos mediante segmentos.

```
(%i4) wxdraw2d(
        points_joined=true,
        points( p ) );
```

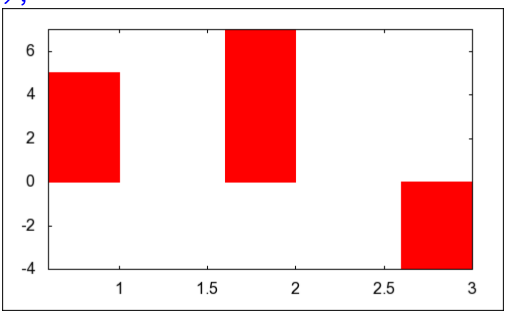
(%t4) 

(%o4)

Maxima

De esta forma se visualiza la gráfica de barras verticales.

```
(%i5) wxdraw2d(
        bars([0.8,5,0.4],[1.8,7,0.4],[2.8,-4,0.4])
      );
```

(%t5) 

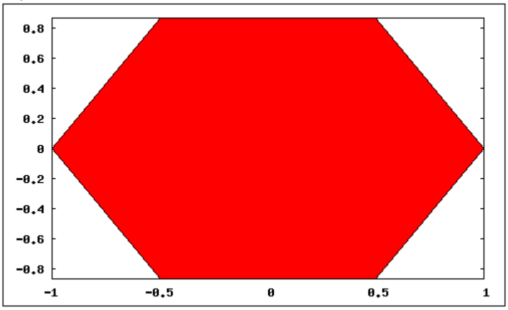
(%o5)

Maxima

Así se visualiza la gráfica de un hexágono.

```
(%i6) wxdraw2d(
        polygon([
        [1/2,sqrt(3)/2],[-1/2,sqrt(3)/2],[-1,0],
        [-1/2,-sqrt(3)/2],[1/2,-sqrt(3)/2],[1,0]
        ])
      );
```

(%t6)



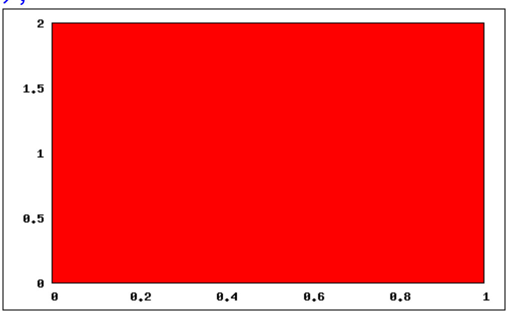
(%o6)

Maxima

De esta forma se visualiza la gráfica de un rectángulo.

```
(%i7) wxdraw2d(
        rectangle([0,0],[1,2])
      );
```

(%t7)



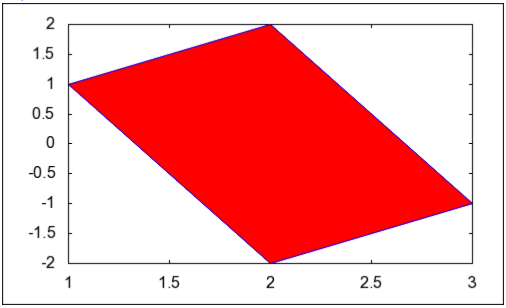
(%o7)

Maxima

Así se visualiza la gráfica de un cuadrilátero.

```
(%i8) wxdraw2d(
        quadrilateral([1,1],[2,2],[3,-1],[2,-2])
      );
```

(%t8)



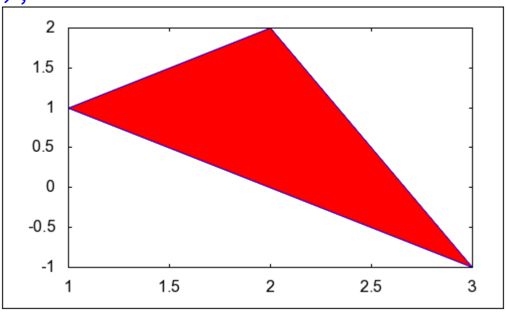
(%o8)

Maxima

Y así, la gráfica de un triángulo.

```
(%i9) wxdraw2d(
        triangle([1,1],[2,2],[3,-1])
      );
```

(%t9)



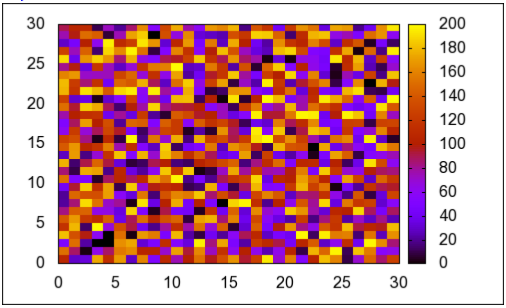
(%o9)

Maxima

Aquí se obtiene la gráfica de una imagen.

```
(%i10) im:apply(matrix,
         makelist(makelist(random(200),i,1,30),i,1,30))$
(%i11) wxdraw2d(
         image(im,0,0,30,30)
       );
```

(%t11)



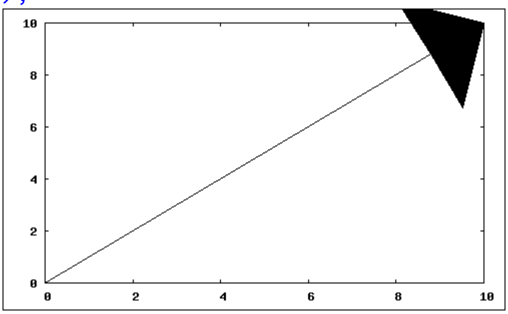
(%o11)

Maxima

De esta manera se obtiene la gráfica del vector (10, 10), cuyo punto de aplicación es el origen.

```
(%i12) wxdraw2d(
         vector([0,0],[10,10])
       );
```

(%t12)



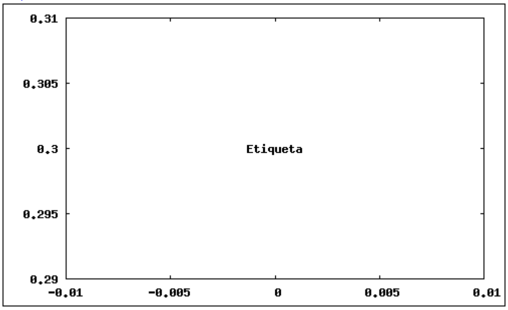
(%o12)

Maxima

Para insertar texto en un punto cualquiera de un gráfico se utiliza la función `label`. En este caso el punto de inserción es $(0, 0 \cdot 3)$.

```
(%i13) wxdraw2d(
         label([“Etiqueta”,0,0.3])
       );
```

(%t13)



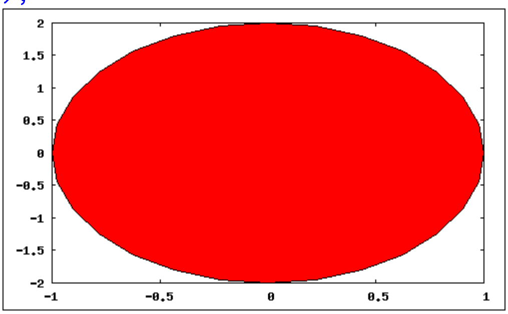
(%o13)

Maxima

Esto devuelve la gráfica de un sector elíptico, de 0° a 360°, cuyo centro es el origen, su semieje horizontal es 1, su semieje vertical es 2.

```
(%i14) wxdraw2d(
         ellipse(0,0,1,2,0,360)
       );
```

(%t14)



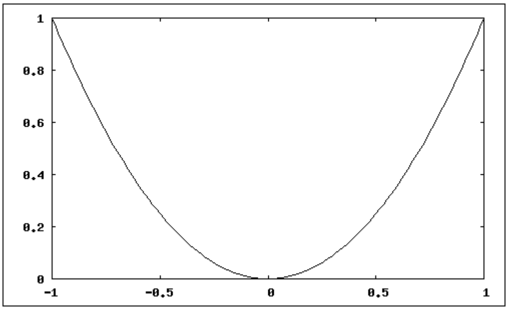
(%o14)

Maxima

De esta manera se obtiene la gráfica de la función $x \to x^2$, con $-1 \le x \le 1$.

```
(%i15) wxdraw2d(
         explicit(x^2,x,-1,1)
       );
```

(%t15)



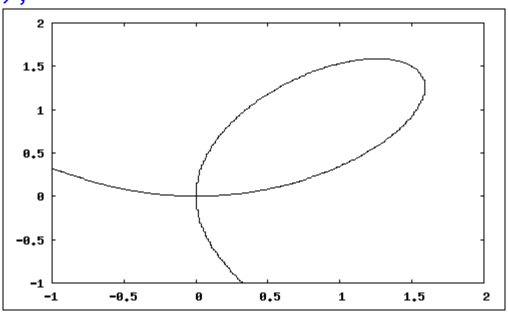
(%o15)

Maxima

Esto muestra la gráfica de la ecuación $x^3 + y^3 - 3xy = 0$ en el rectángulo $[-1, 2] \times [-1, 2]$.

```
(%i16) wxdraw2d(
         implicit(x^3+y^3-3*x*y=0,x,-1,2,y,-1,2)
       );
```

(%t16)



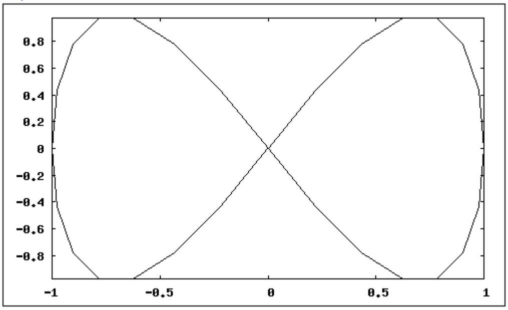
(%o16)

Maxima

Aquí se muestra la gráfica de la curva definida en forma paramétrica mediante $t \to (sin(t), sin(2t))$, con $0 \leq t \leq 2\pi$.

```
(%i17) wxdraw2d(
         parametric(sin(t),sin(2*t),t,0,2*%pi)
       );
```

(%t17)



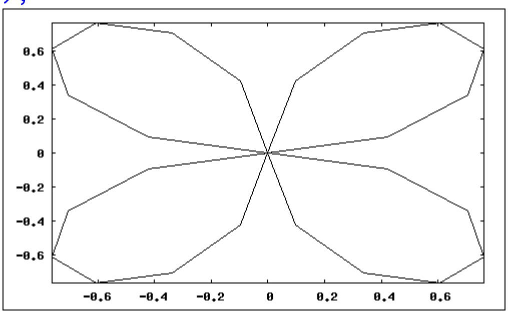
(%o17)

Maxima

Aquí se muestra la gráfica de la función definida en coordenadas polares mediante $t \to \text{sen}(t)$, tal que $0 \leq t \leq 2\pi$.

```
(%i18) wxdraw2d(
         polar(sin(2*t),t,0,2*%pi)
       );
```

(%t18)



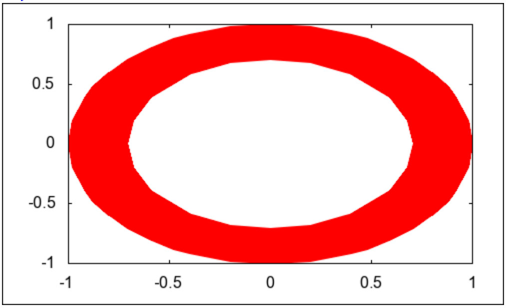
(%o18)

Maxima

Esto muestra la gráfica de los puntos de la región $-1 \leq x \leq 1$, $-1 \leq y \leq 1$ que satisfacen $\frac{1}{2} < x^2 + y^2 < 1$.

```
(%i19) wxdraw2d(
         region(1/2<x^2+y^2 and x^2+y^2<1,
         x,-1,1,y,-1,1)
       );
```

(%t19)



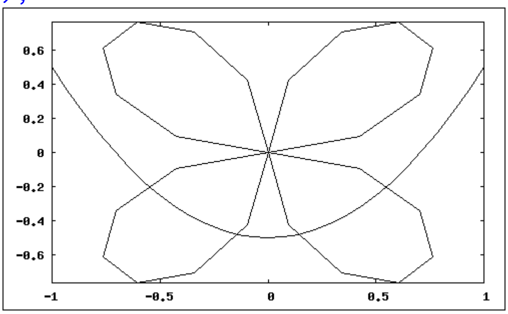
(%o19)

Maxima

Combinar objetos gráficos bidimensionales es sencillo con `draw2d`.

```
(%i20) wxdraw2d(
         parametric(t,t^2-1/2,t,-1,1),
         polar(sin(2*t),t,0,2*%pi)
       );
```

(%t20)



(%o20)

16.2 Opciones para los objetos gráficos bidimensionales

16.2.1 Opciones locales

Las opciones locales sólo son relevantes para objetos gráficos específicos y para tal efecto deben digitarse antes de dicho objeto. Si una opción gráfica es digitada antes de un objeto gráfico al cual no corresponde, no se produce ningún mensaje de error, simplemente la gráfica se muestra sin presentar alteración alguna.

Opción	*Val. por defecto*	*Descripción*
`point_size`	1	establece el tamaño de los puntos dibujados (debe ser un número no negativo)
`point_type`	1	indica la forma que tendrán los puntos aislados
`points_joined`	`false`	indica si los puntos aislados se unen mediante segmentos o no

Opciones de `draw2d` para el objeto gráfico `points`.

Opción	*Val. por defecto*	*Descripción*
`fill_color`	`red`	especifica el color del relleno del polígono
`fill_density`	0	especifica la transparencia del color del relleno del polígono (asume valores entre 0 y 1, incluidos)

Opciones de `draw2d` para el objeto gráfico `bars`.

Opción	*Val. por defecto*	*Descripción*
`nticks`	29	indica el número de puntos a utilizar para generar las gráficas
`border`	`true`	especifica si debe dibujarse el borde o no
`transparent`	`false`	establece si el polígono debe rellenarse o no
`fill_color`	`red`	especifica el color del relleno del polígono

Opciones de `draw2d` para los objetos gráficos bidimensionales `polygon`, `rectangle`, `quadrilateral` y `triangle`.

Opción	*Val. por defecto*	*Descripción*
`palette`	$[7, 5, 15]$	vector con sus componentes tomando valores enteros en el rango desde -36 a $+36$

Opciones de `draw2d` para el objeto gráfico bidimensional `image`.

Opción	*Val. por defecto*	*Descripción*
`head_both`	`false`	indica si el vector será bidireccional o no
`head_length`	2	indica, en las unidades del eje x, la longitud de las flechas de los vectores
`head_angle`	45	indica el ángulo, en grados, entre la flecha y el segmento del vector
`head_type`	`filled`	especifica cómo se habrán de dibujar las flechas de los vectores
`unit_vectors`	`false`	especifica si los vectores se dibujan con módulo unidad

Opciones de `draw2d` para el objeto gráfico `vector`.

Opción	*Val. por defecto*	*Descripción*
`label_alignment`	`center`	especifica el color que tendrá el vector
`label_orientation`	`horizontal`	especifica el color que tendrá el vector

Opciones de `draw2d` para el objeto gráfico `label`.

Opción	*Val. por defecto*	*Descripción*
`nticks`	29	indica el número de puntos a utilizar para generar las gráficas
`adapt_depth`	10	indica el número máximo de particiones utilizadas por la rutina gráfica adaptativa

Opciones de `draw2d` para el objeto gráfico `explicit`.

Opción	*Val. por defecto*	*Descripción*
`filled_func`	`false`	establece cómo se van a rellenar las regiones limitadas por las gráficas
`fill_color`	`red`	especifica el color para rellenar las regiones limitadas por las gráficas

Opciones de `draw2d` para el objeto gráfico `explicit`.

Opción	*Val. por defecto*	*Descripción*
`ip_grid`	$[50, 50]$	establece la rejilla del primer muestreo para la gráfica
`ip_grid_in`	$[5, 5]$	establece la rejilla del segundo muestreo para la gráfica

Opciones de `draw2d` para el objeto gráfico `implicit`.

Opción	*Val. por defecto*	*Descripción*
nticks	29	indica el número de puntos a utilizar para generar las gráficas

Opciones de draw2d para los objetos gráficos bidimensionales parametric y polar.

Opción	*Val. por defecto*	*Descripción*
fill_color	red	especifica el color del relleno del polígono
x_voxel	10	es el número de particiones en la dirección x a utilizar por el algoritmo utilizado para graficar regiones definidas por desigualdades y operadores booleanos
y_voxel	10	es el número de particiones en la dirección y a utilizar por el algoritmo utilizado para graficar regiones definidas por desigualdades y operadores booleanos

Opciones de draw2d para el objeto gráfico bidimensional region.

16.2.2 Opciones locales genéricas

Las opciones locales genéricas son opciones comunes a todos los objetos gráficos bidimensionales. Aunque hay excepciones que cabe destacar, por ejemplo las opciones line_width y line_type que, por razones obvias, únicamente no son relevantes con los objetos gráficos point y label.

Opción	*Val. por defecto*	*Descripción*
`color`	`black`	especifica el color para dibujar líneas, puntos y bordes de polígonos
`line_width`	1	indica el ancho de las líneas a dibujar
`line_type`	`solid`	indica cómo se van a dibujar las líneas (valores posibles son `solid` y `dots`)
`key`	“”	indica la clave del gráfico en la leyenda

Opciones locales genéricas de `draw2d`.

16.2.3 Opciones globales

Las opciones globales se caracterizan porque afectan a toda la escena y su posición dentro de la descripción de ésta no reviste importancia.

Opción	*Val. por defecto*	*Descripción*
`proportional_axes`	`none`	indica si una escena bidimensional se dibujará con los ejes proporcionales a sus longitudes relativas (valores posibles: `none` y `xy`)
`xrange`	`auto`	permite especificar un intervalo para x (valores posibles: `auto` y $[x_{min}, x_{max}]$)
`yrange`	`auto`	permite especificar un intervalo para y (valores posibles: `auto` y $[y_{min}, y_{max}]$)
`logx`	`false`	permite especificar si el eje x se dibujará en la escala logarítmica

Algunas opciones globales de `draw2d`.

Opción	*Val. por defecto*	*Descripción*
`logy`	`false`	permite especificar si el eje y se dibujará en la escala logarítmica
`terminal`	`screen`	selecciona el terminal a utilizar por *Gnuplot* (valores posibles son: `screen`, `png`, `jpg`, `eps`, `eps_color`, `pdf`, `pdfcairo`, `gif`, `animated_gif`, `wxt` y `aquaterm`)
`file_name`	`maxima_out`	indica el nombre del fichero en el que los terminales `png`, `jpg`, `eps`, `eps_color`, `pdf`, `pdfcairo`, etc. guardarán el gráfico
`font`	“”	permite seleccionar el tipo de fuente a utilizar por el terminal
`font_size`	12	permite seleccionar el tamaño de la fuente a utilizar por el terminal
`grid`	`false`	permite especificar si se dibujará una rejilla sobre xy
`title`	“”	almacena una cadena con el título de la escena
`xlabel`	“”	almacena una cadena con la etiqueta del eje x
`ylabel`	“”	almacena una cadena con la etiqueta del eje y
`xtics`	`auto`	controla la forma en la que se dibujarán las marcas del eje x (valores posibles: `auto`, `none`, $[inicio, inc, fin]$, $\{n_1, n_2, \ldots\}$ y también $\{[$“`label1`”$, n_1]$, $[$“`label1`”$, n_1], \ldots\})$

Algunas opciones globales de `draw2d`.

Opción	*Val. por defecto*	*Descripción*
`xtics_axis`	`false`	indica si las marcas y sus etiquetas se dibujan sobre el propio eje x, o se colocan a lo largo del borde del gráfico
`ytics`	`auto`	similar que `xtics` pero con el eje y
`ytics_axis`	`false`	similar que `xtics_axis` pero con el eje y
`xaxis`	`false`	especifica si se dibujará el eje x
`xaxis_width`	1	indica el ancho del eje x
`xaxis_type`	`dots`	indica cómo se debe dibujar el eje x (valores admisibles: `solid` y `dots`)
`xaxis_color`	`black`	indica el color para el eje x
`yaxis`	`false`	especifica si se dibujará el eje y
`yaxis_width`	1	indica el ancho del eje y
`yaxis_type`	`dots`	similar que `xaxis_type` pero con el eje y
`yaxis_color`	`black`	indica el color para el eje y
`file_bgcolor`	“`xffffff`”	establece el color de fondo (en código hexadecimal *rgb*) para los terminales `png`, `jpg` y `gif`
`delay`	5	establece el retraso en centésimas de segundo entre imágenes en los ficheros `gif` animados
`eps_width`	12	indica el ancho (en *cm*) del archivo Postscript generado por los terminales `eps` y `eps_color`
`eps_height`	12	indica el largo (en *cm*) del archivo Postscript

Algunas opciones globales de `draw2d`.

Opción	*Val. por defecto*	*Descripción*
`pdf_width`	$21 \cdot 0$	especifica el ancho (en *cm*) del documento PDF generado por los terminales `pdf` y `pdfcairo`
`pdf_height`	$29 \cdot 7$	especifica el largo (en *cm*) del documento PDF
`pic_width`	640	especifica la anchura del fichero de imagen de bits generado por los terminales `png` y `jpg`
`pic_height`	640	especifica el largo del fichero de imagen de bits
`user_preamble`	“”	inserta código *Gnuplot*

Algunas opciones globales de `draw2d`.

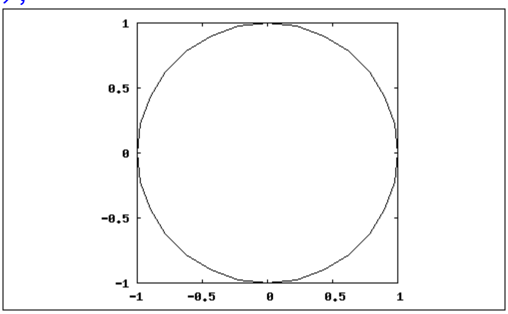
16.2.4 Ejemplos ilustrativos

Maxima

Aquí se usa `user_preamble` para insertar código *Gnuplot*. El resultado, en este caso, equivale a asignar el valor `xy` a la opción `proportional_axes`.

```
(%i1) wxdraw2d(
        parametric(cos(t),sin(t),t,0,2*%pi),
        user_preamble="set size ratio 1"
      );
```

(%t1)



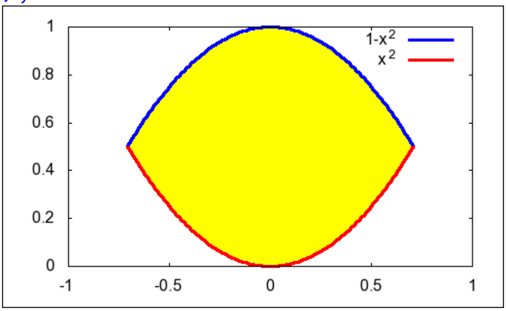
(%o1)

Maxima

He aquí la región encerrada por las parábolas $y = x^2$ y $y = 1 - x^2$.

(%i2)
```
wxdraw2d(
    fill_color=yellow,
    region(x^2<y and y<1-x^2,x,-1,1,y,0,1),
    line_width=3,
    key="1-x^2",
    explicit(1-x^2,x,-sqrt(2)/2,sqrt(2)/2),
    key="x^2",
    color=red,
    explicit(x^2,x,-sqrt(2)/2,sqrt(2)/2)
);
```

(%t2)



(%o2)

Maxima

Aquí se resuelve un sistema de ecuaciones para encontrar los puntos de intersección de las curvas definidas por $x^2 + y^2 = 1$ y $y - 2x^2 + \frac{3}{2} = 0$.

(%i3)
```
sol:solve([x^2+y^2=1,y-2*x^2+3/2=0],[x,y]);
```

(%o3)
$$\left[\left[x = -\frac{\sqrt{\sqrt{5}+5}}{2^{\frac{3}{2}}}, y = \frac{\sqrt{5}-1}{4}\right], \left[x = \frac{\sqrt{\sqrt{5}+5}}{2^{\frac{3}{2}}}, y = \frac{\sqrt{5}-1}{4}\right], \left[x = -\frac{\sqrt{5-\sqrt{5}}}{2^{\frac{3}{2}}}, y = -\frac{\sqrt{5}+1}{4}\right], \left[x = \frac{\sqrt{5-\sqrt{5}}}{2^{\frac{3}{2}}}, y = -\frac{\sqrt{5}+1}{4}\right]\right]$$

Maxima

Esto almacena las coordenadas de los puntos de intersección, previamente calculados, en la variable **pts**.

```
(%i4) pts:map(lambda([h],subst(h,[x,y])),sol);
```

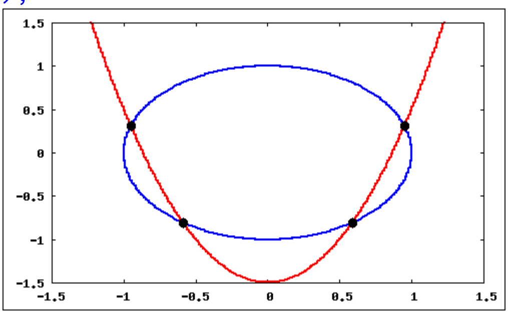
(%o4) $$\left[\left[-\frac{\sqrt{\sqrt{5}+5}}{2^{\frac{3}{2}}},\frac{\sqrt{5}-1}{4}\right],\left[\frac{\sqrt{\sqrt{5}+5}}{2^{\frac{3}{2}}},\frac{\sqrt{5}-1}{4}\right],\left[-\frac{\sqrt{5-\sqrt{5}}}{2^{\frac{3}{2}}},-\frac{\sqrt{5}+1}{4}\right],\left[\frac{\sqrt{5-\sqrt{5}}}{2^{\frac{3}{2}}},-\frac{\sqrt{5}+1}{4}\right]\right]$$

Maxima

He aquí un gráfico de las curvas definidas por $x^2+y^2=1$ y $y-2x^2+\frac{3}{2}=0$ y los respectivos puntos de intersección.

```
(%i5) wxdraw2d(
        line_width=2,
        color=blue,
        implicit(x^2+y^2=1,x,-1.5,1.5,y,-1.5,1.5),
        color=red,
        implicit(y-2*x^2+3/2=0,x,-1.5,1.5,
        y,-1.5,1.5)
        color=black
        point_size=1.5,
        point_type=7,
        points(pts),
      );
```

(%t5)



(%o5)

Maxima

Aquí se define la curva paramétrica `a`.

```
(%i6) a(t):=[t,sin(t)] $
```

Maxima

Esto define el campo vectorial tangente asociado a la curva `a`.

```
(%i7) define(“a'”(t),diff(a(t),t) ) $
```

Maxima

Aquí se construyen la lista `T` de vectores tangentes a la curva `a` y la lista `P` de puntos de aplicación de éstos. Los valores t_0, para tal construcción, son tomados de la lista `t0`.

```
 (%i8) t0:create_list(i*%pi/4,i,0,8) $
 (%i9) T:create_list( vector(a(i),“a'”(i)),i,t0) $
(%i10) P:map(a,t0) $
```

Maxima

Aquí se definen los objetos gráficos curva, a partir de `a`, un conjunto de vectores unitarios tangentes a ésta y los puntos de aplicación de los mismos.

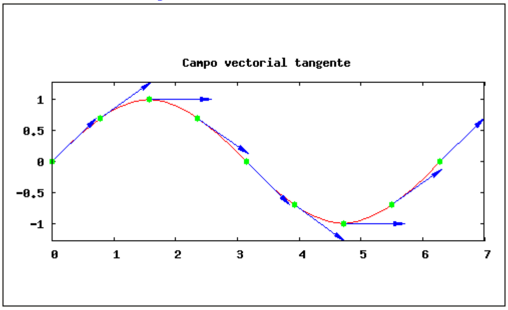
```
(%i11) objetos: [
         color=red,
         apply( parametric,append(a(t),[t,0,2*%pi]) ),
         color=blue,head_length=0.2,
         head_angle=10,
         unit_vectors=true,T,
         color=green,
         point_type=7,points(P),
         proportional_axes=xy,
         title=“Campo vectorial tangente”
       ] $
```

Maxima

De esta manera se grafican los objetos gráficos previamente definidos.

```
(%i12) wxdraw2d(objetos);
```

(%t12)



(%o12)

Maxima

A continuación se construye la lista `N` de vectores normales a la curva **a**.

```
(%i13) J(v):=[-last(v),first(v)] $
(%i14) N:create_list(vector(a(i),J(“a′”(i))),i,t0) $
```

Maxima

Aquí se definen los objetos gráficos curva, a partir de **a**, un conjunto de vectores unitarios tangentes a ésta, un conjunto de vectores unitarios normales a la misma y los puntos de aplicación de los vectores.

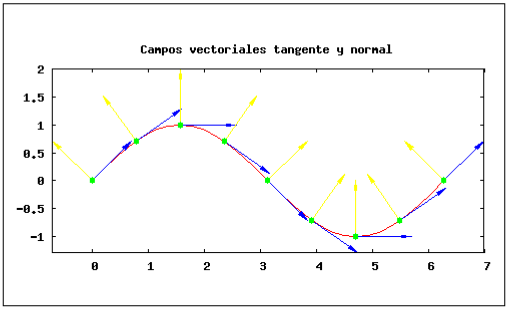
```
(%i15) objetos: [
          color=red,
          apply(parametric,append(a(t),[t,0,2*%pi])),
          head_length=0.2,head_angle=10,
          unit_vectors=true,
          color=blue,T,  color=yellow,N,
          color=green,point_type=7,points(P),
          proportional_axes=xy,
          title=“Campos vectoriales tangente y normal”
       ] $
```

Maxima

De esta manera se grafican los objetos gráficos previamente definidos.

```
(%i16) wxdraw2d(objetos);
```

(%t16)



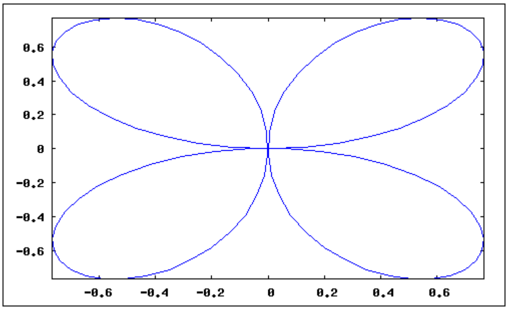
(%o16)

Maxima

He aquí un gráfico polar mejorado.

```
(%i17) wxdraw2d(color=blue,nticks=100,
        polar(sin(2*t),t,0,2*%pi))$
```

(%t17)



16.3 Objetos gráficos tridimensionales

`points`$([[x_1, y_1, z_1], \ldots])$	puntos tridimensionales

Principales objetos gráficos tridimensionales incorporados en `draw`.

`points`$([x_1, x_2, \ldots],$ $[y_1, y_2, \ldots], [z_1, z_2, \ldots])$	puntos tridimensionales
`points(matrix(` $[x_1, y_1, z_1], [x_2, y_2, z_2],$ $\ldots)$	puntos tridimensionales
`vector`$([p_1, p_2, p_3],$ $[v_1, v_2, v_3])$	vector (v_1, v_2, v_3) con punto de aplicación (p_1, p_2, p_3)
`label`$([cadena, x, y, z],$ $\ldots)$	etiquetas para gráficos tridimensionales
`explicit`$(f(x, y),$ $x, x_{min}, x_{max},$ $y, y_{min}, y_{max})$	función explícita f cuyas variables satisfacen $x_{min} \leq x \leq x_{max}$ y $y_{min} \leq y \leq y_{max}$
`implicit`$(E(x, y, z),$ $x, x_{min}, x_{max},$ $y, y_{min}, y_{max})$ $z, z_{min}, z_{max})$	expresión implícita E a ser representada en el paralelepípedo $[x_{min}, x_{max}] \times [y_{min}, y_{max}] \times [z_{min}, z_{max}]$
`parametric`$(f_x, f_y, f_z$ $t, t_{min}, t_{max})$	curva paramétrica tridimensional, cuyo parámetro t satisface $t_{min} \leq t \leq t_{max}$
`parametric_surface(` $f_x, f_y, f_z, \quad u, u_{min}, u_{max},$ $v, v_{min}, v_{max})$	superficie definida paramétricamente, cuyos parámetros satisfacen $u_{min} \leq u \leq u_{max}$ y $v_{min} \leq v \leq v_{max}$
`cylindrical`$(r(z, \theta),$ $z, z_{min}, z_{max},$ $\theta, \theta_{min}, \theta_{max})$	función cilíndrica r cuyas variables satisfacen $z_{min} \leq z \leq z_{max}$ y $\theta_{min} \leq \theta \leq \theta_{max}$
`spherical`$(r(\phi, \theta),$ $\phi, \phi_{min}, \phi_{max},$ $\theta, \theta_{min}, \theta_{max})$	función esférica r cuyas variables satisfacen $\phi_{min} \leq \phi \leq \phi_{max}$ y $\theta_{min} \leq \theta \leq \theta_{max}$
`mesh`$(m, x_0, y_0, ancho,$ $largo)$	define un gráfico tridimensional de la matriz m (los valores z se toman de m, las abscisas van desde x_0 hasta $x_0 + ancho$ y las ordenadas desde y_0 hasta $y_0 + largo$)
`tube`$(f_x, f_y, f_z,$ $f_r, t, t_{min}, t_{max})$	superficie tubular de radio f_r

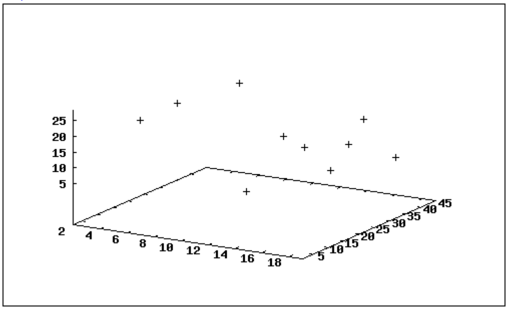
Principales objetos gráficos tridimensionales incorporados en `draw`.

Maxima

Aquí se genera, en forma aleatoria, un conjunto de puntos tridmensionales. Luego se muestra la gráfica de los dichos puntos.

```
(%i1) p:create_list(map(random,[20,50,30]),k,1,10)$
(%i2) wxdraw3d(
        points(p)
      );
```

(%t2)



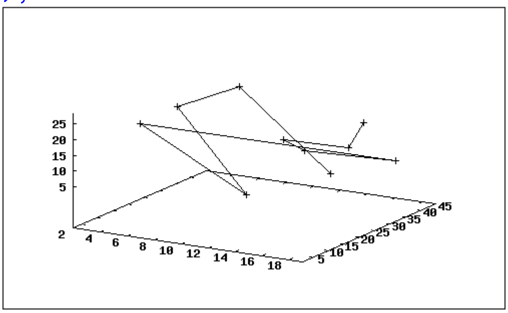
(%o2)

Maxima

Asignando el valor `true` a la opción `plot_joined` se obtiene la poligonal que une los puntos almacenados en `p`.

```
(%i3) wxdraw3d(
        points_joined=true,
        points(p)
      );
```

(%t3)



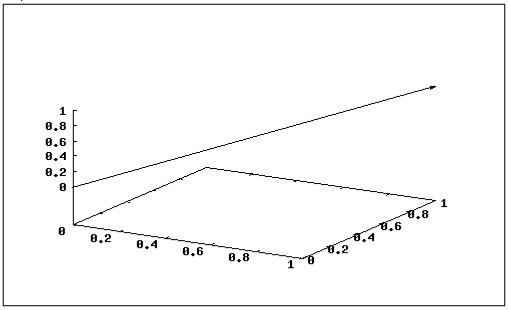
(%o3)

Maxima

Aquí se muestra la gráfica del vector cuyo punto de aplicación es el origen y cuya parte vectorial es $(1, 1, 1)$.

```
(%i4) wxdraw3d(
        vector([0,0,0],[1,1,1])
      );
```

(%t4)



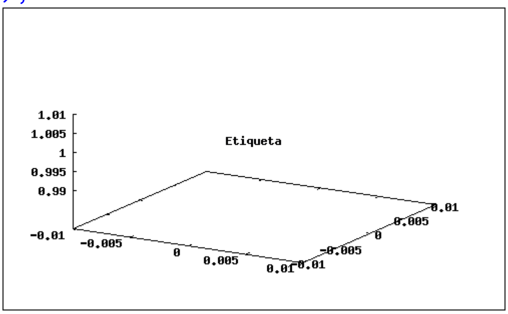
(%o4)

Maxima

He aquí un texto insertado en el punto $(0, 0, 1)$.

```
(%i5) wxdraw3d(
        label([“Etiqueta”,0,0,1])
      );
```

(%t5)



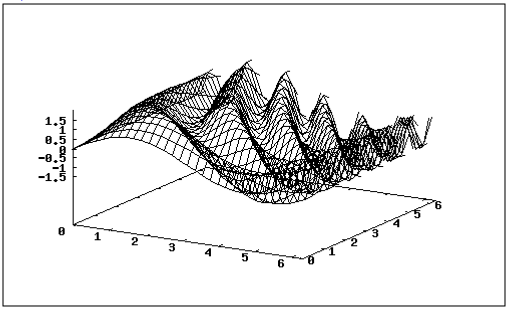
(%o5)

Maxima

Esto muestra la gráfica de la función $(x, y) \to \operatorname{sen} x + \operatorname{sen}(xy)$, con $0 \leq x \leq 2\pi$ y $0 \leq y \leq 2\pi$.

```
(%i6) wxdraw3d(
        explicit(sin(x)+sin(x*y),x,0,2*%pi,y,0,2*%pi)
      );
```

(%t6)



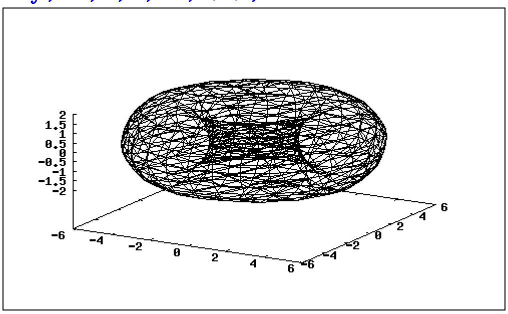
(%o6)

Maxima

Aquí se muestra la gráfica de la superficie generada a partir de la ecuación $(\sqrt{x^2+y^2}-4)^2+z^2=4$, con $-6 \leq x \leq 6$, $-6 \leq y \leq 6$ y $-2 \leq z \leq 2$.

```
(%i7) wxdraw3d(
        implicit((sqrt(x^2+y^2)-4)^2+z^2=4,x,-6,6,
        y,-6,6,z,-2,2) );
```

(%t7)



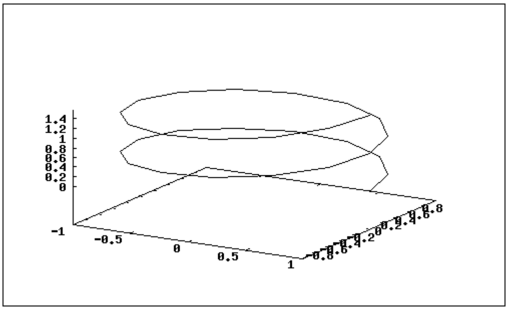
(%o7)

Maxima

Esta es la gráfica de la curva $t \to (\cos t, \operatorname{sen} t, \frac{t}{8})$, con $0 \leq t \leq 4\pi$.

```
(%i8) wxdraw3d(
        parametric(cos(t),sin(t),t/8,t,0,4*%pi)
      );
```

(%t8)



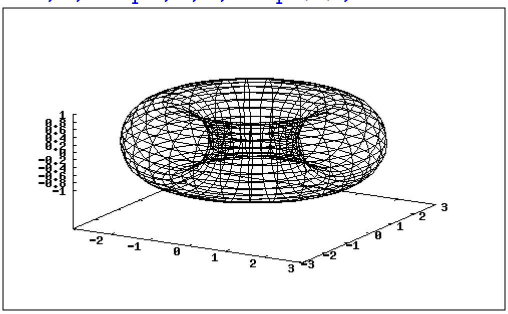
(%o8)

Maxima

Aquí se muestra la gráfica de la superficie definida por $(u, v) \to ((2 + \cos v)\cos u, (2 + \cos v)\sin u, \sin v)$, con $0 \leq u \leq 2\pi$ y $0 \leq v \leq 2\pi$.

```
(%i9) wxdraw3d(
        parametric_surface((2+cos(v))*cos(u),
        (2+cos(v))*sin(u),sin(v),
        u,0,2*%pi,v,0,2*%pi) );
```

(%t9)



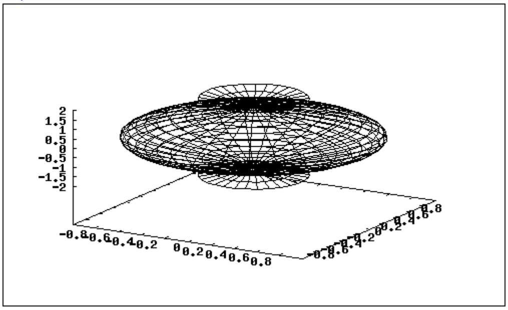
(%o9)

Maxima

Esta es la gráfica de la superficie definida en coordenadas cilíndricas mediante $(z,t) \to \cos z$, con $-2 \le z \le 2$ y $0 \le t \le 2\pi$.

```
(%i10) wxdraw3d(
          cylindrical(cos(z),z,-2,2,t,0,2*%pi)
       );
```

(%t10)



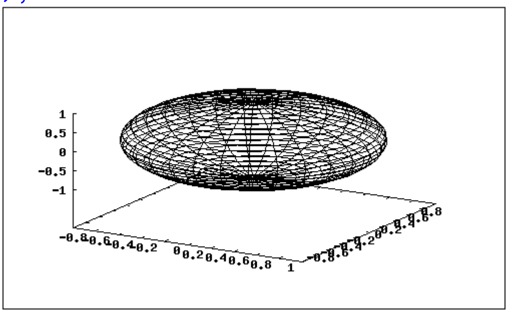
(%o10)

Maxima

He aquí la gráfica de la superficie definida en coordenadas esféricas mediante $(a,t) \to 1$, con $0 \le a \le 2\pi$ y $0 \le t \le \pi$.

```
(%i11) wxdraw3d(
          spherical(1,a,0,2*%pi,t,0,%pi)
       );
```

(%t11)



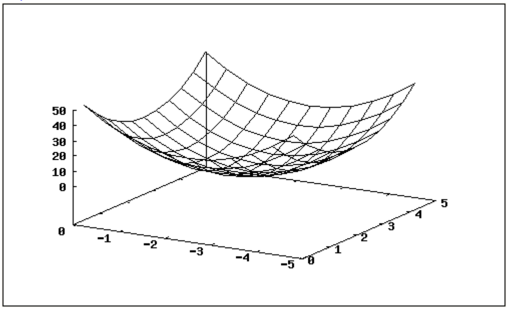
(%o11)

Maxima

Esta es la gráfica de un objeto geométrico tridimensional de tipo **mesh**.

```
(%i12) m:apply(matrix,create_list(create_list(k^2+i^2,
         k,-5,5),i,-5,5))$
(%i13) wxdraw3d(
         mesh(m,0,0,-5,5)
       );
```

(%t13)



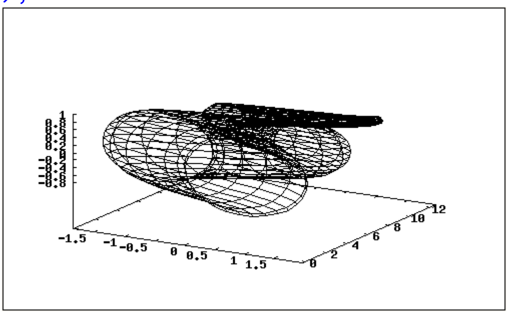
(%o13)

Maxima

He aquí la gráfica de la superficie tubular, de radio $\cos\left(\frac{t}{10}\right)^2$, generada por la curva $t \to (\cos t, t, 0)$, con $0 \leq t \leq 4\pi$.

```
(%i14) wxdraw3d(
         tube(cos(t),t,0,cos(t/10)^2,t,0,4*%pi)
       );
```

(%t14)



(%o14)

16.4 Opciones para objetos gráficos tridimensionales

16.4.1 Opciones locales

Para los objetos gráficos `points` y `label` las opciones locales son las mismas en `draw2d` y `draw3d`. En cambio para el objeto gráfico `vector` las opciones `head_length` y `head_angle` únicamente son relevantes en `draw2d` (vea la subsección 16.2.1). Por este motivo, en esta sección, no se presentan tablas de opciones para los tres objetos gráficos mencionados.

Opción	*Val. por defecto*	*Descripción*
`xu_grid`	30	indica el número de coordenadas de x para formar la rejilla de puntos muestrales
`yv_grid`	30	indica el número de coordenadas de y para formar la rejilla de puntos muestrales

Opciones de `draw3d` para `explicit`, `parametric_surface`, `cylindrical`, `spherical` y `tube`.

Opción	*Val. por defecto*	*Descripción*
`x_voxel`	10	indica el número de voxels en la dirección x a utilizar por el algoritmo implementado
`y_voxel`	10	indica el número de voxels en la dirección y a utilizar por el algoritmo implementado
`z_voxel`	10	indica el número de voxels en la dirección z a utilizar por el algoritmo implementado

Opciones de `draw3d` para `implicit`.

Opción	*Val. por defecto*	*Descripción*
`nticks`	29	indica el número de puntos a utilizar para generar las gráficas

Opciones de `draw3d` para `parametric`.

16.4.2 Opciones locales genéricas

Vea la subsección 16.2.2.

16.4.3 Opciones globales

A excepción de la opción `proportional_axes` todas las opciones globales de `draw2d` son las mismas de `draw3d` (vea la subsección 16.2.3). Además `draw3d` incorpora opciones que complentan, de forma natural, las incorporadas en `draw2d`. Por ejemplo, estan las opciones `zrange`, `logz`, `zlabel`, `ztics`, `zaxis`, `zaxis_type` y `zaxis_color`. Otras opciones se describen a continuación.

Opción	*Val. por defecto*	*Descripción*
`xyplane`	`false`	coloca el plano xy en escenas tridimensionales (para el valor `false`, el plano xy se coloca automáticamente; en cambio, si toma un valor real, éste intersectará con el eje z a ese nivel)
`rot_vertical`	60	indica el ángulo (en grados) de la rotación vertical (alrededor del eje x) para situar el punto del observador en las escenas tridimensionales (el ángulo debe pertenecer al intervalo $[0, 180]$)

Algunas opciones globales de `draw3d`.

Opción	*Val. por defecto*	*Descripción*
`rot_horizontal`	30	indica el ángulo (en grados) de la rotación horizontal (alrededor del eje z) para situar el punto del observador en las escenas tridimensionales (el ángulo debe pertenecer al intervalo $[0, 360]$)
`axis_3d`	`true`	indica si los ejes x, y y z, tradicionales, permanecerán visibles
`palette`	`color`	es un vector de longitud tres con sus componentes tomando valores enteros en el rango desde -36 a $+36$; cada valor es un índice para seleccionar una fórmula que transforma los niveles numéricos en las componentes cromáticas rojo, verde y azul (`palette = gray` y `palette = color` son atajos para `palette = [3,3,3]` y `palette = [7,5,15]`, respectivamente)
`enhanced3d`	`false`	si `enhanced3d` vale `true`, los objetos gráficos se colorearán activando el modo *pm3d* de *Gnuplot*. Si se da una expresión a `enhanced3d` (excepto en `implicit`), ésta se utilizará para asignar colores de acuerdo con el valor de `palette`; las variables de esta expresión deben ser las mismas que luego se utilicen para la descripción de la superficie

Algunas opciones globales de `draw3d`.

Opción	*Val. por defecto*	*Descripción*
`surface_hide`	`false`	establece si las partes ocultas se mostrarán o no en las superficies
`contour`	`none`	permite decidir dónde colocar las líneas de nivel (valores posibles: `none`, `base`, `surface`, `both` y `map`)
`contour_levels`	5	controla cómo se dibujarán las líneas de nivel (valores posibles: n, $[inicio, inc, fin]$ y $\{n_1, n_2, \ldots\}$

Algunas opciones globales de `draw3d`.

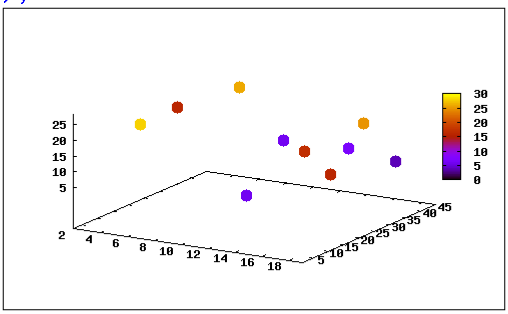
16.4.4 Ejemplos ilustrativos

Maxima

Aquí se han cambiado los valores por defecto de dos opciones de **`draw3d`**. El resultado se aprecia al mostrar los puntos generados en (%i30) .

```
(%i1) wxdraw2d(
        enhanced3d=true,
        point_type=7,point_size=2,
        points(p)
      );
```

(%t1)



(%o1)

Maxima

Esto define una curva, algunos puntos sobre ésta y algunos vectores tangentes a la misma.

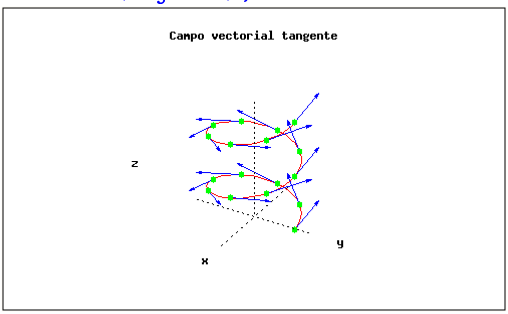
```
(%i2) a(t):=[cos(t),sin(t),t/8] $
(%i3) define(“a′”(t),diff(a(t),t) ) $
(%i4) t0:create_list(i*%pi/4,i,0,16) $
(%i5) T:create_list( vector(a(i),“a′”(i)),i,t0 ) $
(%i6) P:map(a,t0) $
```

Maxima

He aquí la gráfica de todos los objetos gráficos tridimensionales previamente definidos.

```
(%i7) objetos: [
        nticks=100,color=red,
        apply(parametric, append(a(t),[t,0,4*%pi]) ),
        color=blue,unit_vectors=true,T,
        color=green,point_type=7,points(P),
        user_preamble=“set size ratio 1”,
        title=“Campo vectorial tangente”
        xyplane=0,axis_3d=false,
        xtics=false,ytics=false,ztics=false,
        xaxis=true,yaxis=true,zaxis=true,
        xlabel=“x”,ylabel=“y”,zlabel=“z”
      ]) $
(%i8) wxdraw3d(objetos);
```

(%t8)



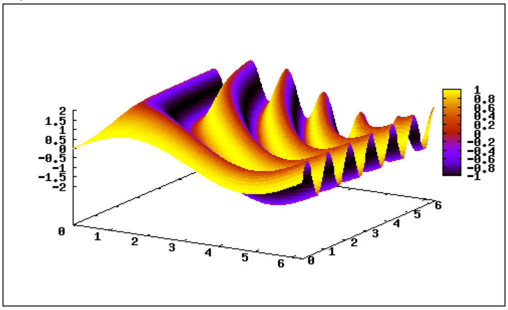
(%o8)

Maxima

Esta es la gráfica de (%t35) después de cambiar algunas opciones.

```
(%i9) wxdraw3d(
        xu_grid=150, yv_grid=150,
        enhanced3d=cos(x*y),
        explicit(sin(x)+sin(x*y),x,0,2*%pi,y,0,2*%pi)
      );
```

(%t9)



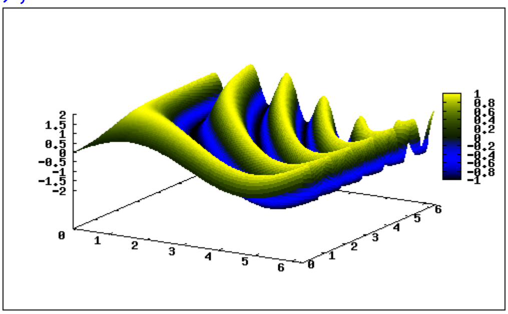
(%o9)

Maxima

Esta es la gráfica de (%t35) después de más cambios de opciones.

```
(%i10) wxdraw3d(
        xu_grid=150,yv_grid=150,
        palette=[6,5,15],enhanced3d=sin(x*y),
        explicit(sin(x)+sin(x*y),x,0,2*%pi,y,0,2*%pi)
      );
```

(%t10)



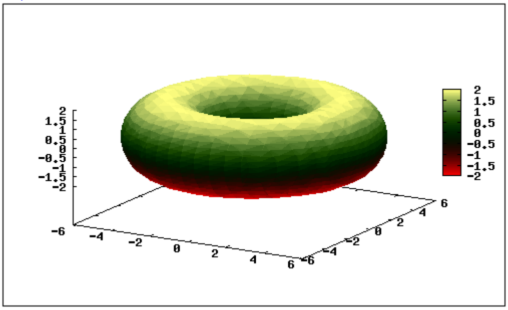
(%o10)

Maxima

He aquí la superficie obtenida en (%t36) .

```
(%i11) wxdraw3d(
         x_voxel=17,y_voxel=17,
         z_voxel=15,
         palette=[12,5,27],
         enhanced3d=true,
         implicit((sqrt(x^2+y^2)-4)^2+z^2=4,x,-6,6,
         y,-6,6,z,-2,2)
       );
```

(%t11)



(%o11)

16.5 Fijación de valores para opciones

En algunas ocasiones se requiere dibujar varios gráficos con las mismas opciones y para evitar escribirlas en cada escena puede optarse por fijar, inicialmente, los valores deseados para dichas opciones. Para hacer factible esto `draw` cuenta con una función específica.

`set_draw_defaults(` *opción gráfica,...* , *opción gráfica,...*`)`	fija los valores para las opciones gráficas del usuario (llamando a la función sin argumentos se borran las opciones fijadas por el usuario)

Fijando valores para opciones.

Maxima

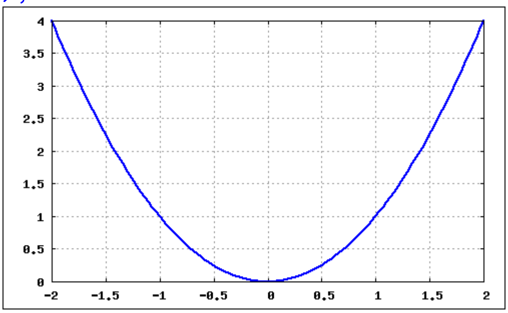
Aquí se fijan los valores de algunas opciones.

```
(%i1) set_draw_defaults(
        surface_hide=true,
        color=blue,
        grid=true,
        line_width=2
      );
```

Maxima

A continuación se grafica una parábola sin especificaciones para ninguna opción.

```
(%i2) wxdraw2d(
        explicit(x^2,x,-2,2)
      );
```

(%t2) 

(%o2)

16.6 Gráficos múltiples

Opción	*Val. por defecto*	*Descripción*
`columns`	1	indica el número de columnas a considerar cuando se realizan gráficos múltiples

Opción para gráficos múltiples.

Maxima

Esto define dos escenas una bidimensional y otra tridmensional.

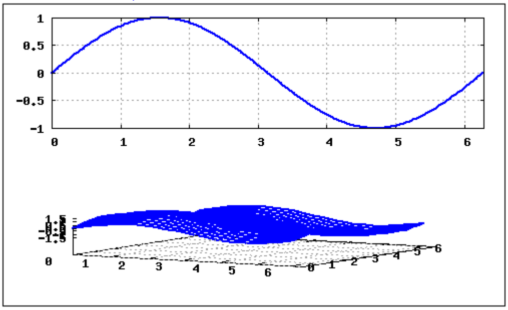
```
(%i1) f1:gr2d(explicit(sin(x),x,0,2*%pi))$
(%i2) f2:gr3d(explicit(sin(x)+sin(y),x,0,2*%pi,
        y,0,2*%pi))$
```

Maxima

Con **draw** la dos escenas previas se presentan por defecto en una sola columna.

```
(%i3) wxdraw(f1,f2)$
```

(%t3)



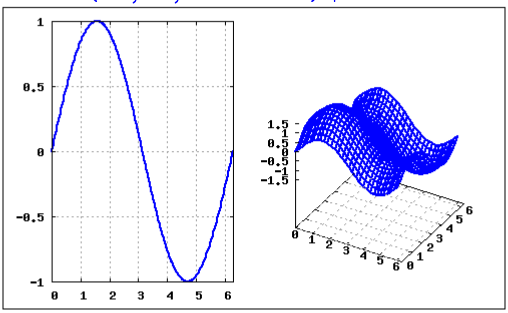
(%o3)

Maxima

Con la opción **columns** es posble cambiar el número de columnas.

```
(%i4) wxdraw(f1,f2,columns=2)$
```

(%t4)



(%o4)

Maxima

De esta manera se borran las opciones que fueron fijadas en (%i51) .

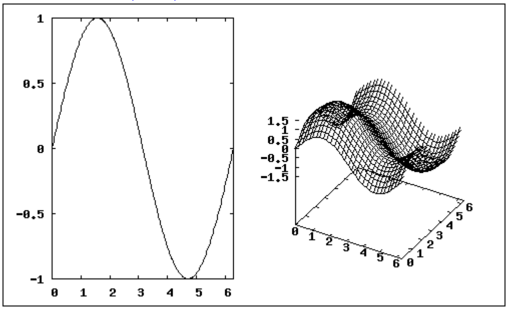
```
(%i5) set_draw_defaults() $
```

Maxima

El aspecto de las gráficas luce ahora diferente, pues se han restituido los valores por defecto de las opciones.

```
(%i6) wxdraw(f1,f2,columns=2) $
```

(%t6)



(%o6)

16.7 Gráficos animados

Del mismo modo que en la sección 14.8 se pueden crear animaciones únicamente en el entorno de *wxMaxima*. En este caso se utilizan las funciones `with_slider_draw` y `with_slider_draw3d` con las cuales se generan gráficos idénticos a los que se generan con `wxplot2d` y `wxplot3d`, respectivamente, sin embargo tales gráficos pueden ser animados después de seleccionarlos y pulsar, luego, el botón Comenzar animación, del cuadro de controles, de la barra de herramientas.

Figura 16.1: Cuadro de controles, ubicado en la barra de herramientas, para la animación de gráficos.

with_slider_draw	función para animar gráficos de wxdraw2d
with_slider_draw3d	función para animar gráficos de wxdraw3d

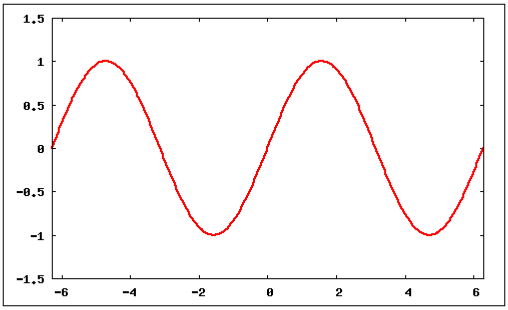
Obtención de gráficos animados con draw.

Maxima

Esto genera un gráfico listo para ser animado. Para conseguir la animación primero se selecciona el gráfico y luego se pulsa el botón, del cuadro de controles, y automáticamente ésta es generada. Para detenerla basta pulsar el botón, del mismo cuadro. También es posible navegar a través de cada cuadro de la animación con tan sólo arrastrar a voluntad el botón, después de haber seleccionado el gráfico.

```
(%i1) with_slider_draw(
        a,[0,1,2,3,4,5],
        color=red,line_width=2,yrange=[-1.5,1.5],
        explicit(sin(x+a),x,-2*%pi,2*%pi)
      );
```

(%t1)



(%o1)

Maxima

A continuación se define la función infija "/*" para calcular el producto vectorial.

```
(%i2) infix("/*")$
(%i3) "/*"(a,b):=[a[2]*b[3]-b[2]*a[3],
        b[1]*a[3]-a[1]*b[3],a[1]*b[2]-b[1]*a[2]]$
```

Maxima

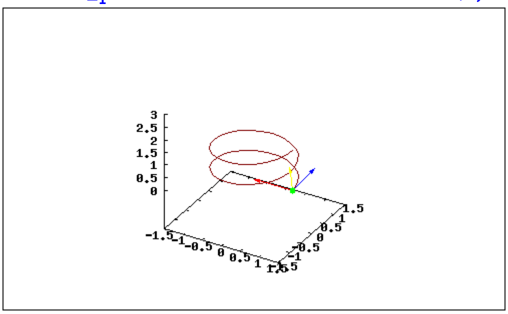
Esto define la curva `a`, su primera derivada y su segunda derivada.

```
(%i4) a(t):=[cos(t),sin(t),t/8] $
(%i5) define(“a′”(t),diff(a(t),t)) $
(%i6) define(“a′′”(t),diff(a(t),t,2)) $
```

Maxima

Al animar la siguiente secuencia de cuadros se visualiza el tiedro de Frenet recorriendo la curva `a`.

```
(%i7) with_slider_draw3d(
        t,create_list(i*%pi/4,i,0,16),
        color=dark-red,nticks=50,
        parametric(cos(u),sin(u),u/8,u,0,4*%pi),
        unit_vectors=true,
        color=blue,
        vector(a(t),“a′”(t)),
        color=yellow,
        vector(a(t),“a′”(t)/*“a′′”(t)),
        color=red,
        vector(a(t),(“a′”(t)/*“a′′”(t))/*“a′”(t)),
        color=green,point_type=7,
        points([a(t)]),
        xrange=[-1.5,1.5],yrange=[-1.5,1.5],
        zrange=[0,3],
        user_preamble=“set size ratio 1”);
```

(%t7) 

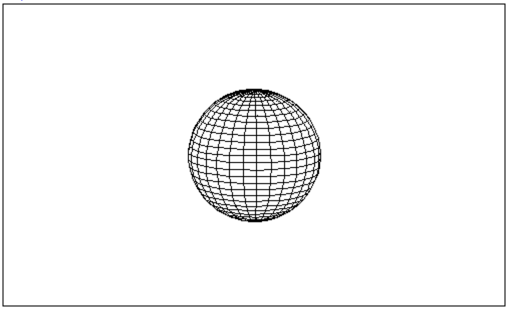
(%o7)

Maxima

Esta es la animación de un morfismo entre la esfera y el toro.

```
(%i8) with_slider_draw3d(
        a,create_list(i/8,i,0,8),
        surface_hide=true,
        user_preamble="set size ratio 1",
        xtics=false,ytics=false,ztics=false,
        axis_3d=false,
        rot_vertical=80,rot_horizontal=100,
        parametric_surface(
        (1-a)*cos(t)*sin(u/2)-(a*sin(u))/3,
        (a*sin(t)*(2-cos(u)))/3+
        (1-a)*sin(t)*sin(u/2),
        (a*cos(t)*(2-cos(u)))/3+(1-a)*cos(u/2),
        t,0,2*%pi,u,0,2*%pi)
      );
```

(%t8)



(%o8)

CAPÍTULO 17

Campos de direcciones con `plotdf`

El paquete `plotdf` permite crear gráficos de campos de direcciones para Ecuaciones Diferenciales Ordinarias (EDO) de primer orden, o para un sistema de dos EDO's autónomas, de primer orden.

Como se trata de un paquete adicional, para poder usarlo debe cargarlo primero con el comando `load("plotdf")`. También es necesario que *Xmaxima* esté instalado, a pesar de que ejecute *Maxima* desde otra interface diferente (esto quiere decir que con *wxMaxima* no puede usarse una función como `wxplotdf`).

Maxima

Inicialización del paquete `plotdf`.

```
(%i9) load(plotdf)$
```

Maxima

Campo de direcciones de la ecuación diferencial $y' = e^{-x} + y$ y la solución que pasa por $(2, -0.1)$.

```
(%i10) plotdf(exp(-x)+y,[trajectory_at,2,-0.1]);
(%o10) 0
```

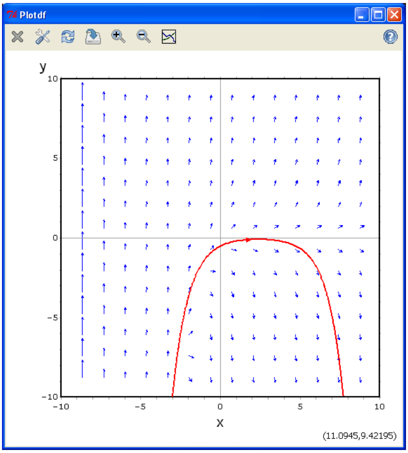


Figura 17.1: Gráfico obtenido con (%i10) .

Maxima

Campo de direcciones de la ecuación diferencial $y' = x - y^2$ y la solución de condición inicial $y(-1) = 3$.

```
(%i11) plotdf(x-y^2,[xfun,“sqrt(x);-sqrt(x)”],
        [trajectory_at,-1,3], [direction,forward],
        [yradius,5],[xcenter,6]);
(%o11) 0
```

Maxima

Campo de direcciones de un oscilador armónico, definido por las ecuaciones $\frac{dx}{dt} = y$ y $\frac{dy}{dt} = -\frac{kx}{m}$, y la curva integral que pasa por $(x, y) = (6, 0)$, con una barra de deslizamiento que permitirá cambiar el valor de m interactivamente (k permanece fijo a 2.)

```
(%i12) plotdf([y,-k*x/m],[parameters,“m=2,k=2”],
        [sliders,“m=1:5”], [trajectory_at,6,0]);
(%o12) 0
```

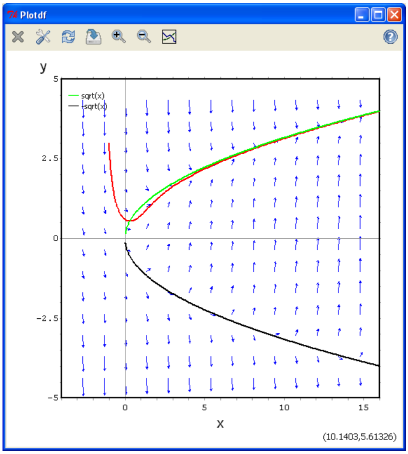


Figura 17.2: Gráfico obtenido con `(%i11)`.

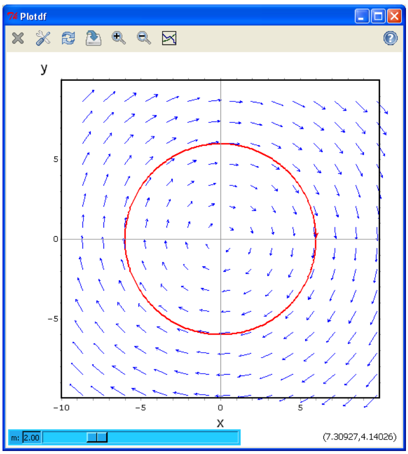


Figura 17.3: Gráfico obtenido con `(%i12)`.

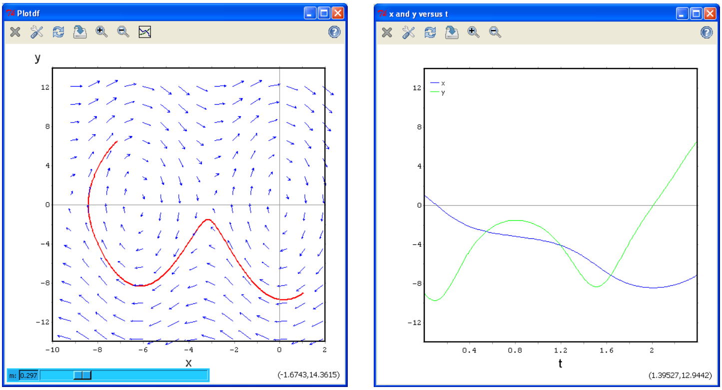


Figura 17.4: Gráficos obtenido con `(%i13)`.

Maxima

Campo de direcciones de un péndulo amortiguado, incluyendo la solución para condiciones iniciales dadas, con una barra de deslizamiento que se puede utilizar para cambiar el valor de la masa, m, y con el gráfico de las dos variables de estado como funciones del tiempo

```
(%i13) plotdf([y,-g*sin(x)/l - b*y/m/l],
         [parameters,“g=9.8,l=0.5,m=0.3,b=0.05”],
         [trajectory_at,1.05,-9],[tstep,0.01],
         [xradius,6],[yradius,14],[xcenter,-4],
         [direction,forward],[nsteps,300],
         [sliders,“m=0.1:1”], [versus_t,1]);
(%o13) 0
```

CAPÍTULO 18

Archivos y operaciones externas

18.1 Generación de expresiones y archivos TEX

Si el usuario quiere combinar su trabajo con material existente en TEX, puede encontrar conveniente usar la función `tex` para convertir expresiones de *Maxima* en forma conveniente de entrada para TEX. El resultado que así se obtiene es un fragmento de código que puede incluirse en un documento mayor, pero que no puede ser procesado aisladamente.

`tex(`*expr*`)`	imprime en la consola la representación en TEX de *expr*
`tex(`*expr*`,false)`	devuelven el código TEX en formato de cadena
`tex(`*expr*`,`*destino*`)`	añade la salida al archivo *destino*

Salidas de *Maxima* para TEX

Maxima

He aquí una expresión, impresa en forma estándar de *Maxima*.

```
(%i1) (x+y)^2/sqrt(x*y);
```

(%o1) $\frac{(y+x)^2}{\sqrt{x\,y}}$

Maxima

He aquí la expresión previa en forma de entrada para TEX.

```
(%i2) tex(%);
      $${{\left(y+x\right)^2}\over{\sqrt{x\,y}}}$$
(%o2) false
```

Maxima

Esto añade la expresión $\frac{(y+x)^2}{\sqrt{x\,y}}$, traducida a TEX, al archivo **ejemplo.tex** ubicado en **d:/maximatex/**.

```
(%i3) tex((x+y)^2/sqrt(x*y),
        "d:/maximatex/ejemplo.tex");
(%o3) false
```

`texput`(a,f)	establece el formato, en TEX, del átomo a, el cual puede ser un símbolo o el nombre de un operador
`get_tex_environment`(op)	devuelve el entorno TEX que se aplica al operador op. Si no se ha asignado ningún entorno, devolverá el que tenga por defecto
`set_tex_environment`(op, $antes$, $después$)	asigna el entorno TEX al operador op
`get_tex_environment_default`()	devuelve el entorno TEXque se aplica a expresiones para las cuales el operador de mayor rango no tiene entorno TEXasignado
`set_tex_environment_default`($antes$, $después$)	asigna el entorno TEXpor defecto

Salidas de *Maxima* para TEX

Maxima

De esta forma se asigna código TEXpara una variable.

```
(%i4) texput(s,“\\sqrt{2}”);
```

(%o4) $\backslash sqrt\{2\}$

Maxima

Ahora puede usarse la asignación anterior para generar más código TEX.

```
(%i5) tex(s+1/2);
      $$\sqrt{2}+{{1}\over{2}}$$
```

(%o5) $false$

Maxima

El entorno TEX aplicado, por defecto, a expresiones, en *Maxima*, es $$ $$.

```
(%i6) tex(3/4);
      $${{3}\over{4}}$$
```

(%o6) $false$

Maxima

Con la función set_tex_environment_default es posible cambiar el entorno TEX. En este caso se ha anulado todo entorno.

```
(%i7) set_tex_environment_default(“ “, ” ”);
```

(%o7) $[\ ,\ ]$

Maxima

He aquí el resultado del nuevo código TEX.

```
(%i8) tex(3/4);
      {{3}\over{4}}
```

(%o8) $false$

Además de traducir expresiones individuales a TEX, *Maxima* también traduce cuadernos completos a documentos pdf LATEX. Para ello el usuario debe digitar el nombre que asignará al archivo resultante, así como la respectiva extensión, `tex`, en la casilla Nombre de la ventana Exportar que aparece después de elegir la opción Exportar del menú Archivo.

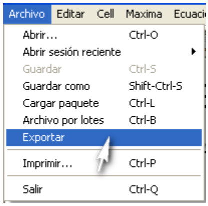


Figura 18.1: Primer paso para exportar un cuaderno.

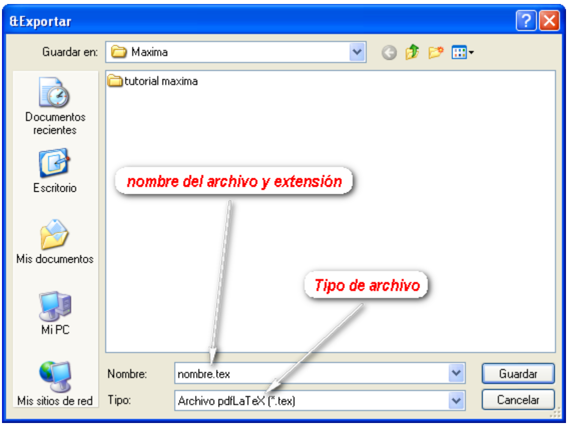


Figura 18.2: Exportando un cuaderno como documento pdf LaTeX

18.2 Generación de archivos HTML

Adicionalmente, *Maxima* brinda capacidades para convertir cuadernos completos a páginas web. El proceso que debe seguirse para la traducción es el mismo que se siguió en la página 273, pero en este caso la extensión será `html`.

18.3 Generación de expresiones Lisp y Fortran

Si el usuario tiene programas escritos en Lisp o Fortran, puede ser que quiera tomar fórmulas que ha generado en *Maxima* e insertarlas en el código original de sus programas. *Maxima* permite convertir expresiones matemáticas en expresiones Lisp y Fortran.

`:lisp $%i`*n*	escribe para Lisp la expresión a la que hace referencia la etiqueta `%i`*n*
`fortran(`*expr*`)`	escribe *expr* para Fortran

Salidas de *Maxima* para Lisp y Fortran.

Maxima

Aquí se obtienen expresiones para Lisp y Fortran.

(%i1) `x^2+2*x+1;`

(%o1) $x^2+2\,x+1$

(%i2) `:lisp $%i9;`

```
((MPLUS) ((MEXPT) $X 2) ((MTIMES) 2 $X) 1)
```

(%i3) `fortran(x^2+2*x+1);`

$x**2+2*x+1$

(%o3) *done*

CAPÍTULO 19

Programación con *Maxima*

La elaboración de programas con el lenguaje de programación de *Maxima* permite al usuario definir sus propias funciones. De esta manera se hace posible la automatización de secuencias de operaciones que son útiles para abordar la solución de un determinado tipo de problema.

Además, es posible implementar varias funciones, relacionadas con cierto tema, y guardarlas en un solo archivo que luego se pueda ejecutar sin necesidad de visualizar todo el código elaborado. A tal archivo se le conoce como *paquete*[1] de funciones.

19.1 Operadores relacionales y lógicos

De manera similar que cualquier lenguaje de programación *Maxima* incluye operadores relacionales y lógicos, así como estructuras de control (que se utilizan para controlar el flujo del programa en una rutina).

En *Maxima*, los operadores lógicos pueden ser *infijos* o *prefijos*. Un operador recibe el nombre de infijo cuando éste debe escribirse entre los operandos, por ejemplo el operador `and` cuya sintaxis es p `and` q, para ciertos operandos p y q. Por otra parte, un operador prefijo es

[1] Del inglés *package*. Un paquete es almacenado en forma automática por *Maxima* como archivo de extensión `lisp` y el código es convertido internamente al lenguaje de programación `Lisp`.

aquel que debe escribirse antes del operando, por ejemplo el operador `not` cuya sintaxis es `not` p, para cierto operando p.

Cabe destacar que, en *Maxima*, casi todos los operadores lógicos son infijos y únicamente hay un operador lógico prefijo.

Operador	*Símbolo*	*Tipo*
menor que	$<$	operador relacional infijo
menor o igual que	$<=$	operador relacional infijo
igualdad (sintáctica)	$=$	operador relacional infijo
negación de $=$	$\#$	operador relacional infijo
igualdad (por valor)	`equal`	operador relacional infijo
negación de `equal`	`notequal`	operador relacional infijo
mayor o igual que	$>=$	operador relacional infijo
mayor o igual que	$>=$	operador relacional infijo
y	`and`	operador lógico infijo
o	`or`	operador lógico infijo
no	`not`	operador lógico prefijo

Operadores relacionales y lógicos.

Controlador	*Descripción*
`if`	permite, mediante una condición, que se ejecute o no se ejecute determinada tarea o línea de código
`for`	es utilizado para generar una repetición de instrucciones entre un número inicial y un número final que deben ser indicados o, también, entre un conjunto de elementos de una lista
`while`	repetirá sin detenerse un determinado código mientras se cumpla una condición
`unless`	repetirá sin detenerse un determinado código hasta que se cumpla una condición
`do`	se utiliza para realizar iteraciones con `for`, `while` y `unless`

Principales estructuras de control.

Maxima

He aquí un ejemplo sencillo en el que se muestra la sintaxis de `if`.

```
(%i1) if 2=3 then 1;
(%o1) false
(%i2) if 3=3 then 1;
(%o2) 1
```

Maxima

En este ejemplo se añade el resultado ha obtener en caso de que la condición sea falsa.

```
(%i3) if 2=3 then 1 else 3;
(%o3) 3
```

Téngase en cuenta que las expresiones consideradas en los ejemplos de las entradas (%i1) y (%i2) equivalen a las expresiones:
`if 2 = 3 then 1 else false` y `if 3 = 3 then 1 else false`, respectivamente.

Maxima

Aquí se obtiene el valor ligado a la expresión verdadera más próxima (en este caso `3 = 3`). Si todas las expresiones fuesen falsas, entonces se obtendría el último valor (en este caso `4`).

```
(%i4) if 2=3 then 1 elseif 3=3 then 5 else 4;
(%o4) 5
```

Maxima

Mediante este otro ejemplo se muestra la sintaxis de `for`.

```
(%i5) for i:1 thru 5 step 2 do print(i);
        1
        3
        5
(%o5) done
```

Maxima

Si no se indica el paso (incremento) se asume uno, por defecto.

```
(%i6) for i:1 thru 5 do print(i);
        1
        2
        3
(%o6) done
```

Maxima

Para inicializar el contador en `while` se utiliza `for`.

```
(%i7) for i:1 while i<=3 do print(i);
        1
        2
        3
(%o7) done
```

Maxima

Para inicializar el contador en `unless` también se utiliza `for`.

```
(%i8) for i:1 unless i>=4 do print(i);
        1
        2
        3
(%o8) done
```

19.2 Operadores y argumentos

Casi todo en *Maxima* es un objeto de la forma

$$\mathbf{fun}(a_1, \ldots, a_n),$$

es decir, una expresión que comprende un operador como **fun** y los argumentos de éste, $a_1, \ldots, a_n$. Las funciones `op` y `args` permiten averiguar la estructura de las expresiones.

`op`(*expr*)	permite obtener el operador de la expresión *expr*
`args`(*expr*)	permite obtener una lista cuyos elementos son los argumentos de la expresión *expr*

Obtención del operador y de los argumentos de una expresión.

Maxima

Aquí se define la expresión $a+b$, la cual es almacenada en la variable **`expr`**.

```
(%i1) expr:a+b;
```
(%o1) $b+a$

Maxima

A continuación, con la primera operación se obtiene el operador de la expresión previamente definida; y en la segunda, se obtiene una lista con los argumentos de dicha expresión.

```
(%i2) op(expr);
```
(%o2) $+$

```
(%i3) args(expr);
```
(%o3) $[b,a]$

Maxima

Es posible "reconstruir" la expresión usando la función **`apply`**.

```
(%i4) apply(op(expr),args(expr));
```
(%o4) $b+a$

Maxima

Esto define otra expresión.

```
(%i5) expr:ejemplo(x,y,z);
```
(%o5) $ejemplo(x,y,z)$

Maxima

Aquí, nuevamente, se obtienen el operador y los argumentos de la expresión.

```
(%i6) op(expr);
```
(%o6) $ejemplo$

```
(%i7) args(expr);
```
(%o7) $[x, y, z]$

Maxima

También es posible "reconstruir" la última expresión usando la función `apply`.

```
(%i8) apply(op(expr),args(expr));
```
(%o8) $ejemplo(x, y, z)$

Maxima

Algunas expresiones (`plot2d`, `integrate`, etc.) requieren de una comilla simple.

```
(%i9) expr:'plot2d(x^2,[x,-1,1]);
```
(%o9) $plot2d(x^2, [x, -1, 1])$

Maxima

Ahora si es posible obtener el operador y los argumentos de la expresión.

```
(%i10) op(expr);
```
(%o10) $plot2d$

```
(%i11) args(expr);
```
(%o11) $\left[x^2, [x, -1, 1]\right]$

Maxima

También las listas y los conjuntos son objetos de la forma $\mathbf{fun}(a_1, \ldots, a_n)$. Aquí se almacena una lista cualquiera en la variable `expr`.

```
(%i12) expr:[a,b,c,d];
```
(%o12) $[a, b, c, d]$

Maxima

Aquí también se obtienen el operador y una lista con los argumentos.

```
(%i13) op(expr);
(%o13) [
(%i14) args(expr);
(%o14) [a, b, c, d]
```

19.3 Programación funcional

La *programación funcional* es la programación que pone énfasis en el uso de funciones. *Maxima* incluye las funciones predefinidas `apply`, `map`, `lambda` que permiten apreciar la potencia de la programación funcional.

`apply`$(f, [expr_1, \ldots, expr_n])$	construye y evalúa la expresión $f(arg_1, \ldots, arg_n)$
`map`$(f, expr_1, \ldots, expr_n)$	devuelve una expresión cuyo operador principal es el mismo que aparece en las expresiones $expr_1, \ldots, expr_n$ pero cuyas subpartes son los resultados de aplicar f a cada una de las subpartes de las expresiones
`lambda`$([x_1, \ldots, x_m], expr_1, \ldots, expr_n)$	define y devuelve una expresión lambda (es decir, una función anónima) con argumentos $x_1, \ldots, x_m$; la cual devuelve el valor $expr_n$
`lambda`$([\,[L]\,], expr_1, \ldots, expr_n)$	define y devuelve una expresión lambda con argumento opcional L; la cual devuelve el valor $expr_n$
`lambda`$([\,x_1, \ldots, x_m, [L]\,], expr_1, \ldots, expr_n)$	define y devuelve una expresión lambda con argumentos $x_1, \ldots, x_m$, argumento opcional L; la cual devuelve el valor $expr_n$

Funciones predefinidas para programación funcional.

Maxima

Aquí se define la función G, la cual es aplicada luego a una lista cuyos elementos pasan a ser los argumentos de G.

```
(%i1) G(x,y,z):=x^2+y^2+z^2 $
(%i2) apply(G,[x-y,a+b,u]);
```

(%o2) $(x-y)^2+u^2+(b+a)^2$

Maxima

En este caso se aplica la función predefinida `min`.

```
(%i3) apply(min,[7,9,3,4]);
```

(%o3) 3

Maxima

Esto define la función F y luego la mapea en una lista.

```
(%i4) F(x):=x^3-1 $
(%i5) map(F,[2,3,5,a]);
```

(%o5) $\left[7, 26, 124, a^3-1\right]$

Maxima

Aquí se muestra la definición de una función lambda f, la misma que posee dos argumentos.

```
(%i6) f:lambda([x,y],x+y) $
(%i7) f(a,b);
```

(%o7) b+a

Maxima

Ahora se define una función lambda con argumento opcional.

```
(%i8) f:lambda([ [x] ],x^2) $
(%i9) f(p);
```

(%o9) $\left[p^2\right]$

```
(%i10) f(p,q,r,s,t);
```

```
(%o10) [p^2, q^2, r^2, s^2, t^2]
```

Maxima

En este ejemplo se define una función lambda con dos argumentos y un argumento opcional. Luego esta función es evaluada en tres argumentos y se obtiene una lista con el resultado esperado, no obstante al evaluar la función en más argumentos se obtiene una lista con tantos elementos como argumentos adicionales hay.

```
(%i11) f:lambda([ x,y,[z] ],x*z+y) $
(%i12) f(p,q,r);
(%o12) [p r + q]
(%i13) f(p,q,r,s,t,u,v,w);
(%o13) [p r + q, p s + q, p t + q, p u + q, p v + q, p w + q]
```

19.4 Implementación del paquete: `ejemplo`

En esta sección se implementará el paquete `ejemplo`, que incorporará las funciones `triangulo` y `circunferencia`.

Maxima

Aquí se define la función `triangulo`. Esta función permite calcular el área de un triángulo y el baricentro de un conjunto de puntos de $\mathbb{R}^2$, dados. Además, si los puntos son colineales, devuelve el mensaje: *Los puntos son colineales.*

```
(%i1) triangulo(pts):=
        block([f:lambda([h],endcons(1,h)),pts1,M,d,a],
        pts1:map(f,pts),
        M:apply(matrix,pts1),
        d:determinant(M),
        a:abs(d/2),
        if d=0 then string("Los puntos son colineales")
        else
        (b:apply("+",pts)/3,
        [sconcat(Area,": ",a," ",u^2),
        sconcat(Baricentro,": ",b)])
        ) $
```

Maxima

Dado el conjunto de puntos del plano $\{(1,2),(3,-1),(2,3)\}$, no colineales, la función **triangulo** devuelve el área y las coordenadas del baricentro del triángulo definido por los puntos dados.

```
(%i2) p:[ [1,2],[3,-1],[2,3] ] $
(%i3) triangulo(p);
```

(%o3) $[\ Area:\ 5/2\ u\hat{}2,\ Baricentro:\ [2,4/3]\ ]$

Maxima

Para visualizar los gráficos se utilizará el paquete **draw**.

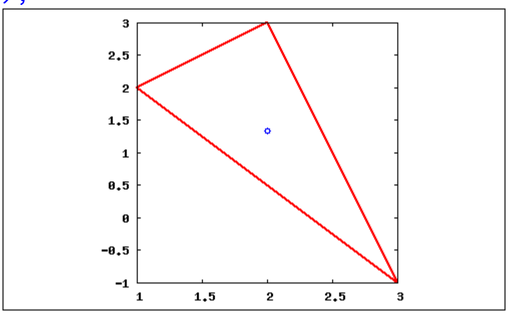
```
(%i4) load(draw) $
```

Maxima

Esto muestra la gráfica del triángulo previamente analizado, conjuntamente con el punto que corresponde al baricentro del mismo.

```
(%i5) wxdraw2d(
         color=red,fill_color=white,line_width=2,
         polygon(p),
         point_type=6,color=blue,
         points([ [2,4/3] ]),
         user_preamble="set size ratio 1"
      );
```

(%t5)



(%o5)

Maxima

Puesto que, en este caso, los puntos del conjunto $\{(1,1),(2,2),(3,3)\}$ son colineales, la función `triangulo` devuelve un mensaje indicando éste hecho.

```
(%i6) p:[ [1,1],[2,2],[3,3] ] $
(%i7) triangulo(p);
```

(%o7) "*Los puntos son colineales*"

Maxima

Aquí se define la función `circunferencia` que permite obtener la ecuación de la circunferencia definida por tres puntos dados.

```
(%i8) circunferencia(pts):=
        block([f:lambda([h],endcons(1,h)),
        g:lambda([h],cons(h[1]^2+h[2]^2,h)),
        pts1,M,d,eq],
        pts1:map(f,pts),
        M:apply(matrix,pts1),
        d:determinant(M),
        if d=0 then string("Los puntos son colineales"),
        else
        (aux:map(g,cons([x,y,1],pts1)),
        M:apply(matrix,aux),
        d:determinant(M),
        eq:expand(d),
        expand(eq/coeff(eq,x^2))=0)
      ) $
```

Maxima

Dado el conjunto de puntos del plano $\{(1,2),(3,-1),(2,3)\}$, no colineales, la función `circunferencia` devuelve la ecuación de la circunferencia definida por estos puntos.

```
(%i9) p:[ [1,2],[3,-1],[2,3] ] $
(%i10) circunferencia(p);
```

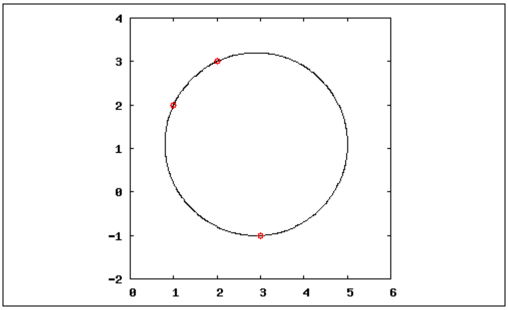
(%o10) $y^2 - \frac{11\,y}{5} + x^2 - \frac{29\,x}{5} + \frac{26}{5} = 0$

Maxima

Esto muestra la gráfica de la circunferencia previamente analizada, conjunatmente con los puntos que la definen.

```
(%i11) wxdraw2d(
         implicit(circunferencia(p),x,0,6,y,-2,4),
         point_type=6,color=red,point_size=1,
         points(p),
         user_preamble="set size ratio 1"
       );
```

(%t11)



(%o11)

Maxima

Igual que con la función **`triangulo`**, para el conjunto de puntos del plano $\{(1,1),(2,2),(3,3)\}$, la función **`circunferencia`** devuelve un mensaje indicando que éstos son colineales.

```
(%i12) p:[ [1,1],[2,2],[3,3] ] $
(%i13) circunferencia(p);
```

(%o13) "*Los puntos son colineales*"

Hasta aquí se han definido, y se ha verificado el correcto funcionamiento de, las funciones `triangulo` y `circunferencia`. Seguidamente se guardará la definición de estas funciones en un fichero de nombre *ejemplo* y de extensión *lisp*, el cual constituirá un ejemplo de un paquete de funciones definido por el usuario.

Naturalmente, si el usuario desea puede copiar solamente la definición de las citadas funciones y luego guardarlas sin necesidad de copiar y ejecutar los ejemplos.

Maxima

Esto guarda todas las funciones definidas por el usuario en el archivo `ejemplo.lisp`. El directorio donde se guardará el fichero es indicado por el usuario, en este caso éste es: `d:/maximapackages`.

```
(%i14) save("d:/maximapackages/ejemplo.lisp",
       functions) $
```

Una vez que se ha guardado la definición de las funciones en el mencionado fichero es posible inicializarlo como si se tratara de cualquier paquete incorporado en *Maxima*.

En este caso se asume que el usuario ha cerrado el cuaderno de trabajo actual de *Maxima* y luego ha abierto un nuevo cuaderno desde el cual inicializará el paquete `ejemplo.lisp`.

Maxima

Esto inicializa el paquete `ejemplo.lisp` desde el directorio en el que se guardó (vea 15.1).

```
(%i15) load("d:/maximapackages/ejemplo.lisp") $
```

Maxima

Ahora es posible ejecutar cualquiera de las funciones incorporadas en el paquete `ejemplo.lisp` sin necesidad de exponer el código del mismo.

```
(%i16) p:[ [0,0],[2,0],[1,sqrt(3)] ] $
(%i17) triangulo(p);
```

(%o17) $[\,Area:\ sqrt(3)\ u\hat{}2,\ Baricentro:\ [1,1/sqrt(3)]\,]$

Bibliografía

[1] Fokker, J. PROGRAMACIÓN FUNCIONAL. http://people.cs.uu.nl/jeroen/courses/fpsp.pdf (1996).

[2] Rodríguez, J. R. MAXIMA CON WXMAXIMA: SOFTWARE LIBRE EN EL AULA DE MATEMÁTICAS. http://knuth.uca.es/repos/maxima (2007).

[3] Rodríguez, M. y Villate, J. MANUAL DE MAXIMA ver. 5.18. http://maxima.sourceforge.net/es/documentation.html (2009).

[4] Rodríguez, M. PRIMEROS PASOS EN MAXIMA. www.telefonica.net/web2/biomates/maxima/max.pdf (2008).

[5] Rodríguez, M. SOFTWARE MATEMÁTICO BÁSICO: MAXIMA. www.telefonica.net/web2/biomates/maxima/i-math.pdf (2008).

[6] Rodríguez, M. MAXIMA: UNA HERRAMIENTA DE CÁLCULO. http://softwarelibre.uca.es/cursos/maxima/cadiz.pdf (2006).