UNIVERSIDAD NACIONAL AUTÓNOMA
DE MÉXICO

FACULTAD DE CIENCIAS

DESCRIPCIÓN DEL SISTEMA DE ÁLGEBRA COMPUTACIONAL MAPLE, CON ÉNFASIS EN ÁLGEBRA LINEAL

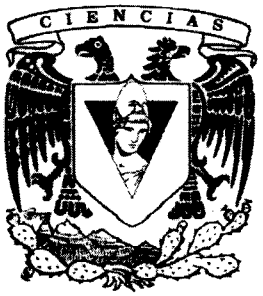
T E S I S

QUE PARA OBTENER EL TÍTULO DE

M A T E M Á T I C O

P R E S E N T A :

JOSÉ LUIS TORRES RODRÍGUEZ



FACULTAD DE CIENCIAS
UNAM

DIRECTOR DE TESIS:

DR. ROBERTO ALICIAN SUSSMAN LIVOVSKY

MÉXICO, D. F. 2004

Autorizo a la Dirección [illegible] de la
UNAM a difundir en [illegible] impreso y
en [illegible] electrónico el
contenido de mi trabajo recepcional.
NOMBRE: José Luis Torres
Rodríguez
FECHA: febrero 10, 2004
FIRMA:

A la memoria de Aryan, por todo lo que ha significado para mi.

A Mitzha, mi *"chaparrita"* preciosa, por ser uno de mis mayores tesoros, por ser mi inspiración, mi aliento, mi alegría y mi mejor amiguita. Gracias por existir.

A Obed, mi *"chaparrito"* lindo, por haber traido alegría y unión a nuestra familia, por ser el complemento esencial en nuestras vidas. Gracias por haber llegado a nosotros.

A mis hijos, por darme el privilegio y el honor de ser padre; por todas sus enseñanzas, por las alegrías y los hermosos momentos que hemos vivido, por todas las esperanzas y realidades que me he forjado inspirado en ellos, y por ayudarme a comprender el verdadero significado de la vida.

A Yolanda, por ser la madre de mis hijos y mi compañera en la vida, por su paciencia y apoyo incondicional, por todas las alegrías, los buenos recuerdos y por haber estado a mi lado en los momentos difíciles.

A mis padres y hermanos, por su apoyo, por ser mis guías, compañeros y amigos en todo momento.

A mis sobrinos: Miriam, Juan, Omar, Luis y Ana Karen, por ser mis amiguitos y la inspiración de mi familia.

A Roberto Sussman, por todo el impulso que dio a este trabajo; por la paciencia, el apoyo y la amistad brindados.

A Guillermo Gómez, por recordarme que debía finalizar este trabajo, por ser en muchas ocasiones maestro, guía y sobre todo amigo.

A todos mis sinodales, por el apoyo y los consejos esenciales para el desarrollo de esta tesis.

A mis amigos, en especial a Héctor, Luis Enrique y Luis Alberto, por todo el apoyo incondicional, por los momentos compartidos, por la comprensión invaluable que siempre han demostrado, por recordarme que la amistad es uno de los mayores tesoros en la vida.

A mis maestros, compañeros y a todos los que contribuyeron de alguna forma, para que pudiera alcanzar esta meta.

Por último, pero no por ello menos importante, a Pablo Rosell, por compartir sus vastos conocimientos de LaTeXpara la creación de este trabajo.

DRA. MARÍA DE LOURDES ESTEVA PERALTA
Jefa de la División de Estudios Profesionales de la
Facultad de Ciencias
Presente

Comunicamos a Usted que hemos revisado el trabajo escrito:

"Descripción del Sistema de Álgebra Computacional Maple, con énfasis en Álgebra Lineal"

realizado por José Luis Torres Rodríguez con número de cuenta 8619688-0

quién cubrió los créditos de la carrera de Matemáticas

Dicho trabajo cuenta con nuestro voto aprobatorio.

Atentamente

Director de Tesis
Propietario Dr. Roberto Allan Sussman Livovsky

Propietario M. en C. Guillermo Gómez Alcaraz

Propietario Dr. Pedro Eduardo Miramontes Vidal

Suplente M. en C. José Luis Gutiérrez Sánchez

Suplente Mat. Luis Manuel Hernández Gallardo Luis Manuel Hernández Gallardo

Consejo Departamental de Matemáticas

M. en C. José Antonio Gómez Ortega

FACULTAD DE CIENCIAS
CONSEJO DEPARTAMENTAL
DE
MATEMATICAS

Descripción del sistema de álgebra computacional "Maple", con énfasis en Álgebra Lineal

José Luis Torres Rodríguez

enero de 2004

Contenido

Introducción

Los primeros sistemas de cómputo existentes presentaban multiples obstáculos a quienes hacían uso de ellos para resolver algún problema. Se trataba de equipos con requerimientos muy particulares, tales como sistemas especiales de ventilación y de instalaciones eléctricas (debido al tipo de componentes en base a los cuales estaban construidos). Además, recuérdese que estos primeros equipos fueron creados a partir de proyectos de investigación, por lo cual se trataba de ingenios únicos en el mundo, lo cual implicaba una buena cantidad de problemas de compatibilidad. Otro de los inconvenientes eran las capacidades tan limitadas de almacenamiento de información que ofrecían. Por otro lado, las personas que estaban capacitadas para operar este tipo de equipos generalmente eran los mismos expertos que habian participado en el diseño y construcción de los mismos, debido a que necesitaban conocer perfectamente su funcionamiento, por lo cual existian pocas personas capaces de usarlos adecuadamente. Aunado a esto, para poder crear un programa se debía recurrir a lenguajes de programación poco amigables, como el ensamblador o el Fortran; con todas las deficiencias y dificultades de uso que representan, como herramientas de uso general para desarrollo de aplicaciones.

Otro obstáculo a salvar eran las grandes cantidades de dinero que debían invertirse para poder contar con una computadora, de hecho solamente empresas grandes, universidades y centros de investigación con presupuestos suficientes podían tener acceso a una de ellas. Esto solo por mencionar algunas de las desventajas de los primeros equipos de cómputo.

En cuanto a las aplicaciones de uso científico la historia es similar. Los primeros trabajos en el área de diseño de algoritmos y sistemas para realizar cálculos matemáticos simbólicos (actualmente conocidos como "Sistemas de Álgebra Computacional"), datan de principios de los 60's y fueron desarrollados en instituciones como el M.I.T. Las aplicaciones de este tipo desarrolladas durante la decada de los 70's (por ejemplo Macsyma y Reduce), eran enormes programas basados en Lisp que requerían grandes cantidades de memoria y de tiempo de procesador para llevar a cabo cálculos matemáticos rutinarios. En consecuencia, solo un pequeño grupo de personas (generalmente investigadores), con acceso a grandes equipos de cómputo, podían explotar este tipo de tecnología; todo ello sin tomar en cuenta que generalmente el tiempo de uso de la computadora era por demás restringido, ya que se debía compartir con varios equipos de trabajo.

Estos problemas (entre muchos otros), dificultaban el que una computadora pudiera ser usada en las actividades diarias en un ámbito académico; sobre todo, en estas condiciones era muy difícil que cualquier alumno de una universidad pudiera hacer uso de esta herramienta como apoyo.

Afortunadamente el desarrollo de esta tecnología continuó y en la década de los 80's, principalmente con el advenimiento de las PC's y las Macintosh, comienza a ser cada vez más común e indispensable la presencia de una computadora en la vida diaria. Cada día es más fácil que una persona pueda tener acceso a un equipo con la capacidad suficiente para ejecutar programas que le faciliten o le auxilien en el desarrollo de sus labores cotidianas; por lo cual cada vez existen menos razones para no hacer uso de este tipo de equipos como apoyo en estas labores, ya no se diga en un ámbito académico.

Paralelamente a la evolución de las computadoras se han venido desarrollando aplicaciones cada vez más completas en las áreas más diversas. Actualmente existe mayor disponibilidad de programas para satisfacer a una cada vez más demandante y creciente comunidad de usuarios, abarcando practicamente todas las actividades humanas; desde software de entretenimiento, aplicaciones de oficina, programas para diseño y

manipulación de imágenes, video y sonido, ambientes para desarrollo de aplicaciones, manejadores de bases de datos, generadores de código, multitud de lenguajes de programación para diferentes paradigmas, hasta software educativo y aplicaciones de uso científico, entre muchos otros. En esta última área podemos encontrar programas tales como Maple y Mathematica, los cuales nos proporcionan gran cantidad de funciones y otras facilidades para la solución de problemas de matemáticas y otras áreas afines.

El objetivo de este trabajo es precisamente mostrar las capacidades de una de estas aplicaciones de uso científico: Maple; de tal forma que pueda ser de utilidad como herramienta de apoyo en un ambito académico, auxiliando y facilitando las actividades y tareas necesarias para el estudio de temas propios de carreras científicas.

Características de Maple

Maple, al igual que otros programas tales como Mathematica, Maxima y Fermat (entre muchos otros), es conocido como una aplicación de "Álgebra Computacional". Esta clasificación agrupa aquellos sistemas creados en base a algoritmos para realizar cálculos matemáticos simbólicos; aunque estrictamente hablando, Maple puede ser mejor definido como un Sistema de Álgebra Computacional con capacidades numéricas, simbólicas y gráficas.

Básicamente, "cálculo simbólico" (una de las capacidades fundamentales de Maple) implica cálculos matemáticos sobre números, símbolos, expresiones y fórmulas, de una manera exacta; lo opuesto al cálculo numérico, el cual trata el manejo de números de punto flotante y por lo tanto de aproximaciones. Entre las operaciones típicas del cálculo simbólico se pueden incluir: diferenciación, integración, solución de ecuaciones y sistemas de ecuaciones (de manera simbólica), manipulación de polinómios, entre muchas otras. En esta área, Maple proporciona una gran cantidad de funciones para llevar a cabo este tipo de operaciones y muchas más, en las cuales se pueden incluir expresiones simbólicas.

Además de cálculos simbólicos, Maple tiene la capacidad de llevar a cabo cálculos numéricos tan complejos como sea necesario (por ejemplo, tiene la capacidad para manejar números de precisión infinita). Al respecto, este sistema proporciona también una gran cantidad de funciones que nos permiten obtener aproximaciones a partir de expresiones numéricas de todo tipo.

Otra de las características importantes de Maple es su capacidad para generar gráficas, en dos y tres dimensiones, de todo tipo de funciones. Entre otras se pueden mencionar: funciones explícitas, implícitas, paramétricas, de contornos, de densidades y de vectores; es capaz de desplegar gráficas a partir de listas arbitrarias de puntos, curvas de nivel, de soluciones de ecuaciones diferenciales, de campos vectoriales y gráficas de funciones en los complejos entre muchas otras. Además, puede desplegar estas gráficas en diferentes sistemas de coordenadas (cilíndricas, esféricas, parabólicas, cónicas, etc). También permite crear animaciones de todo tipo de funciones, tanto en dos como en tres dimensiones, proporcionando al usuario control total sobre el despliegue de éstas.

Otra característica notable es el soporte de un lenguaje de programación, a partir del cual están creadas la mayor parte de las funciones proporcionadas, y a partir del cual el usuario puede definir las suyas propias, incrementando de manera personalizada las capacidades del sistema.

Un punto más en el que es importante abundar es en la estructura de Maple, la cual está construida de manera modular. La parte central de este sistema es un pequeño módulo escrito en C, conocido como Kernel, que contiene un conjunto de funciones básicas, soporte para las reglas sintácticas, las estructuras de datos manejadas y para el lenguaje de programación mencionado, entre otras cosas. El resto de las funciones están escritas en este lenguaje y son agrupadas por su área de aplicación en archivos independientes conocidos como "paquetes". Cada vez que se inicia la ejecución de Maple lo único que se carga en la memoria de la computadora es el kernel, el cual proporciona los elementos básicos; posteriormente el usuario puede agregar funciones de los diferentes paquetes a la memoria para su uso, o bien crear sus propias rutinas a partir del lenguaje de programación de la aplicación. Este tipo de diseño permite optimizar el espacio de memoria utilizado, ya que no es necesario cargar todos los elementos del sistema para poder hacer uso de él;

inicialmente solo se tiene acceso a una pequeña parte (el núcleo), pero el resto puede accederse gradualmente conforme el usuario lo requiera.

Aunado a esto, este sistema tiene la capacidad de interactuar directamente con aplicaciones como Matlab, Excel y Scientific Workplace. También puede convertir archivos de Maple a los formatos HTML y Latex. Otra característica importante es que puede generar código en C, Fortran y Java, a partir de procedimientos propios; más aun, puede ser invocado desde programas escritos en estos lenguajes o bien ejecutar este tipo de programas para interactuar con instrucciones propias de Maple.

En Maple 8 también se ha incluido, como parte de la aplicación, soporte para creación y manejo de *maplets*; los cuales son similares a los applets usados en páginas Web. Usando el paquete Maplets, proporcionado por esta versión, es posible crear ventanas, cuadros de diálogo y otros elementos para generar una interfaz interactiva para el usuario. Estos maplets pueden ser ejecutados directamente en la hoja de trabajo donde fueron creados; o bien, pueden ser guardados como archivos independientes y visualizados como una aplicación fuera de Maple, por medio del programa MapletViewer8, capaz de interpretar y desplegar el maplet.

También, en esta versión de Maple, se ha incluido soporte para manejo de documentos de XML, así como un conjunto de rutinas para acceso a los directorios del sistema, directamente desde el código de esta aplicación.

Origenes de Maple

El proyecto a partir del cual se generó esta aplicación fue concebido en noviembre de 1980 en la Universidad de Waterloo. El objetivo principal era diseñar un sistema de álgebra computacional que pudiera ser puesto masivamente a disposición de investigadores de las áreas de matemáticas, ingeniería y ciencias en general; además de un gran número de estudiantes, con propósitos educacionales.

Otra de las principales ideas durante el diseño era lograr la mayor eficiencia en espacio y tiempo de ejecución; para ello se decidió hacer la implementación del sistema en algún lenguaje de la familia BCPL en lugar de Lisp, como era la tendencia para varios de los sistemas creados con anterioridad. Inicialmente esta implementación se llevó a cabo en lenguaje B en un equipo Honeywell, aunque posteriormente se reconoció que la mejor opción para ello era C, dadas sus características; por lo cual el núcleo del sistema fue migrado a este lenguaje. Esta migración también se debió a otro de los objetivos principales del diseño; hacer el sistema disponible a una gran cantidad de usuarios requería que fuera portable, de tal manera que la implementación pudiera ser llevada a una amplia variedad de computadoras, conforme estas aparecieran en el mercado. Otro punto importante necesario para cumplir con el aspecto de la eficiencia fue llevado a cabo a través de la investigación en el diseño de algoritmos para operaciones matemáticas, con el objetivo de maximizar el rendimiento de éstos en cuanto a tiempo de procesador.

Desde sus inicios, el proyecto de Maple tomó el enfoque de implementar de manera inmediata los conceptos generados en el diseño, lo cual permitió que en diciembre de 1980 apareciera una primera versión, aunque con capacidades limitadas. Por otro lado, también desde el inicio las diferentes ideas surgidas en el diseño han sido aceptadas o rechazadas rápidamente, de tal forma que se puede concebir a Maple como un sistema en constante evolución. Este proceso de evolución continúa hasta la fecha a un paso significativo; y dado que el propósito principal es la "mecanización de las matemáticas", tomando en cuenta que el campo que abarca esta disciplina es muy grande, no puede señalarse un punto de finalización de este proyecto.

Las primeras presentaciones de Maple al público se hicieron en 1982, en diferentes eventos; sin embargo la más importante fue la conferencia:

> *"The design of Maple: A compact, portable, and powerful computer algebra system"*
>
> B.W. Char, K.O. Geddes, W.M. Gentleman, and G.H. Gonnet

la cual se llevó a cabo en 1983; en ésta fueron presentados los criterios principales del diseño. Dicha conferencia apareció publicada posteriormente en el documento:

Computer Algebra (Proceedings of EUROCAL '83)
J. A. van Hulzen (ed.), Lecture Notes in Computer Science, No. 162
Springer-Verlag, Berlin, 1983, pp. 101-115.

Como se mencionó antes, la primera implementación de este sistema fue creada pocas semanas después del inicio del proyecto; unos meses más tarde comenzo a ser usado como soporte para el curso de algoritmos algebraicos: "Introduction to Symbolic Computation", dirigido a estudiantes de la Universidad de Waterloo y varios matemáticos de esta misma universidad comenzaron a incluirlo en sus trabajos de investigación a partir de 1982.

Posteriormente, en 1983, investigadores de diferentes instituciones comenzaron a usar este software en diversas disciplinas, incluyendo matemáticas, ciencias de la computación, ingeniería, física y economía; de tal manera que a finales de este año ya existian alrededor de 50 instalaciones de Maple fuera de Waterloo. A partir de entonces comenzó a darse de manera significativa una demanda de versiones para diferentes sistemas operativos, así como de soporte para su instalación y uso.

En 1984, el grupo de investigadores que iniciaron el desarrollo creó un convenio con la empresa WATCOM Products Inc. para que ésta se hiciera cargo de la licencia y distribución de Maple. Se determinó que las capacidades matemáticas del sistema debían ser mejoradas en ciertas áreas, que éste debía ser migrado a los sistemas VM/CMS de IBM y VMS de DEC; además se sugirieron cambios considerables en el Manual de Referencia, de tal forma que pudiera servir al usuario como una verdadera guía. Estas modificaciones fueron llevadas a cabo y un año después apareció una nueva versión lista para su distribución, la cual inició a mediados de 1985; a partir de entonces la demanda de este sistema ha venido creciendo gradualmente. En 1987 existían alrededor de 300 instalaciones a nivel mundial en múltiples sistemas operativos, posteriormente entre 1988 y 1990 se alcanzó la cifra de aproximadamente 2000 instalaciones a nivel mundial. Para tener una idea del crecimiento que ha tenido este software hasta la actualidad, cabe mencionar que actualmente solo en la UNAM existen varios cientos de instalaciones disponibles a los miembros de esta Universidad, mientras que a nivel mundial la cifra asciende a varios cientos de miles.

Aplicaciones de Maple

A lo largo de los años Maple ha probado ser de gran utilidad en diversas áreas. Para tener una idea, a continuación mancionaremos algunos ejemplos de las muchas aplicaciones en las que ha estado involucrado recientemente:

- Robótica. La contribución de Canada a la Estación Espacial Internacional es el "Mobile Servicing System", un complejo mecanismo que incluye "brazos" robotizados capaces de manipular objetos de cualquier dimensión, desde muy pequeños hasta muy grandes. Debido a que fue diseñado para su uso en el espacio, no es posible probar de manera efectiva este manipulador en la tierra; por lo tanto, la Agencia Espacial Canadiense ha optado por usar modelos matemáticos y simuladores desarrollados en Maple, para llevar a cabo tales pruebas y de esta forma garantizar el correcto funcionamiento de este aparato en condiciones de ingravidez.

- Medicina. Por ejemplo, investigadores del "University of Connecticut Health Center (UCHC)" están explorando aplicaciones de Maple en la interpretación de imágenes obtenidas por resonancia magnética, para la detección oportuna y caracterización de tumores; esto les ha permitido - entre otras cosas -, reducir considerablemente el tiempo de análisis de tales imágenes de alrededor de 200 horas a unos cuantos segundos (en los mejores casos).

- Telecomunicaciones. La empresa Nortel Networks ha hecho uso de Maple para generar un modelo diseñado para estudiar el efecto de la introducción de sus servicios de banda ancha.

- Cine. La empresa Pacific Data Images, de Palo Alto California, creó varias de las escenas que aparecen en la película Antz (Hormiguitas), con ayuda de la versión 6 de Maple.
- Educación. En los últimos doce años Maple ha sido parte importante como apoyo en el contenido de diversas materias de ciencias e ingeniería impartidas en la Universidad de Queensland. Por otro lado este sistema también ha servido como herramienta, durante ocho años, para un programa de educación a distancia de la Universidad de Stanford en el cual participan alumnos de todo el mundo.

Objetivo de este trabajo

Dada la naturaleza de las carreras impartidas en la Facultad de Ciencias, es conveniente contar con un sistema que pueda auxiliar a los estudiantes en el manejo de problemas de tipo matemático desde un punto de vista científico, pero haciendo uso de las capacidades de cálculo de una computadora. Un software que les permita abordar, aprovechando el poder de los equipos actuales, un amplio rango de problemas en diversas áreas de las matemáticas; desde problemas simples de visualización o solución de ecuaciones lineales, hasta problemas complejos, tales como manejo de sistemas dinámicos, integrales complicadas, manejo de números de precisión grande, visualización de funciones, de sistemas de ecuaciones o de sus soluciones, así como la obtención misma de soluciones simbólicas y numéricas de ecuaciones diversas, cálculo de integrales, derivadas y límites, operaciones de álgebra lineal, simulaciones de fenómenos matemáticos, físicos o biológicos, etc.

Este trabajo es una propuesta para cubrir parte de la necesidad de contar con una formación básica en cómputo para los alumnos de los primeros semestres de la Facultad. Dentro de esta propuesta se plantéa la necesidad de contar con un curso en el cual el alumno obtenga conocimientos de algún sistema de álgebra computacional, particularmente de Maple. Sin embargo, esta tesis también fue desarrollada con el objetivo de que se convierta en un material de autoestudio para todos los estudiantes interesados en el manejo de esta aplicación; razón por la cual se ha incluido al inicio material básico, tanto del manejo de la interfaz como del manejo mismo de Maple.

Existen otras aplicaciones que pueden sugerirse para tales fines; sistemas como: Mathematica, Macsima, Derive, entre otros. Sin embargo, dadas las características antes expuestas de Maple, nuestra sugerencia esta enfocada a este programa. Otro punto a favor del uso de este sistema es la sencillez de la sintaxis, más parecida a la notación matemática que la usada por otros programas. Un detalle más que refuerza esta elección es el hecho de que actualmente se cuenta con una licencia institucional de Maple en la UNAM, con lo cual cualquier estudiante de la Universidad puede tener acceso a este software.

Existen varios sistemas que pueden ofrecer un mejor rendimiento y prestaciones para el usuario, en aplicaciones muy específicas. Por ejemplo, para realizar cálculos numéricos la elección por excelencia para quienes se dedican a esta área es Fortran o Matlab. Algunos otros sistemas también han probado ser más eficientes en otras áreas; por ejemplo el paquete Fermat ha demostrado poder hacer un mejor manejo de problemas que involucran cierto tipo de polinómios y matrices[1]. Sin embargo, se trata de aplicaciones orientadas a un área o a un tipo específico de problemas, y no de un sistema con todas las prestaciones de Maple. Al respecto, en la actualidad el único sistema que se le puede comparar es Mathematica, el cual presenta diversas desventajas con respecto a Maple; en cuanto a la estructura del programa, la complejidad de la sintaxis, así como en el rendimiento de los algoritmos usados.

Por lo anterior, consideramos que esta aplicación resulta ser la mejor elección para los fines antes expuestos, si se le compara con otros sistemas con características similares y con capacidades para la solución de problemas simbólicos, numéricos y gráficos, como los mencionados anteriormente para Maple; por lo cual, el lector con seguridad encontrará de gran utilidad este material como apoyo en el manejo de problemas propios de temas científicos y de áreas afines.

[1] Aunque cabe mencionar que Maple 8 ha introducido mejoras considerables en los tiempos de ejecución en cálculos simbólicos y numéricos, en el manejo de polinómios y matrices, en operaciones de álgebra lineal y de ecuaciones diferenciales, entre otras cosas.

Capítulo 1

Interfaz de Maple 8

Al iniciar un nuevo archivo, Maple nos presenta un documento en blanco en el cual podemos comenzar a ejecutar instrucciones y escribir información. También nos proporciona un conjunto de opciones que podemos aplicar sobre el documento actual, entre las cuales se encuentran varias operaciones de edición de texto, así como algunas otras que nos permiten dar presentación al archivo. Todas éstas son implementadas en los menús con los que cuenta la interfaz gráfica y muchas también pueden ser aplicadas a través de los botones presentes en forma de barra de herramientas.

En este capítulo se describen algunos de los elementos más importantes de la interfaz gráfica de esta aplicación, además de las diferentes ***regiones*** que conforman una hoja de trabajo. Todos los documentos creados con Maple están formados por varias de estas regiones, por lo que es importante poder distinguirlas, así como saber el tipo de operaciones que podemos aplicar sobre cada una de ellas para manipularlas; de tal manera que se obtengan documentos legibles y organizados. Entre estas operaciones se encuentran el uso de diferentes tipos de fuentes y estilos de texto, y la edición y alineación de párrafos. También, en este capítulo se describen algunas de las opciones proporcionadas a los usuarios para dar presentación a sus documentos, como son el uso de secciones y subsecciones, además del manejo de marcadores e hipervínculos.

1.1 Elementos de la Interfaz Gráfica

La versión gráfica de Maple está formada básicamente por los siguientes elementos:

1.1.1 Barra de Menús

La barra de menús (Menu Bar), como su nombre lo indica, está compuesta por los diferentes menús que proporciona Maple; los elementos y opciones que contiene varían dependiendo del contexto. Por ejemplo, cuando estamos trabajando en un documento, introduciendo instrucciones o comentarios, este menú muestra las opciones que aparecen en la Figura 1.1.

Figura 1.1: Barra de menús estandar

A ésta se le conoce como barra de menús estándar (para consultar la hoja de ayuda de estos menús ejecute la instrucción **?worksheet,reference,standard**).

En cambio, si nos encontramos en la ventana de ayuda, la barra que aparece se conoce como barra de menús de la página de ayuda (para obtener información de esta barra ejecute **?worksheet,reference,help**). Los elementos de ésta se muestran en la Figura 1.2

Figura 1.2: Barra de menús de la página de ayuda

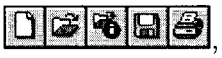
A continuación se enumeran algunas otras barras de menús que están disponibles según el contexto en el que se está trabajando y se proporciona la instrucción para poder consultar su página de ayuda.

- Barra de menús de gráficas en dos dimensiones. Consultese su hoja de ayuda ejecutando:
  ?worksheet,reference,plot2dmenu.
- Barra de menús de gráficas en tres dimensiones. Para consultar su hoja de ayuda ejecutese:
  ?worksheet,reference,plot3dmenu.
- Barra de menús de hojas de cálculo. Para ver su página de ayuda ejecutese:
  ?worksheet,reference,SpreadsheetMenus.

La página de ayuda de todas estas barras también puede consultarse ejecutando, en la línea de comandos: **?worksheet,reference,menus**

1.1.2 Barra de Herramientas

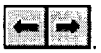
Esta barra (Tool Bar en inglés) está formada por varios conjuntos de botones que implementan diferentes opciones de los menús de Maple. A continuación se describe la función de cada uno de ellos:

- El primer grupo, formado por los botones:  , nos proporciona acceso directo a las opciones **New**, **Open**, **Open URL**, **Save** y **Print**, del menú **File**, respectivamente; los cuales tienen la misma utilidad que en otras aplicaciones.
- El segundo grupo de botones:  , nos permiten aplicar de manera rápida las opciones de **Cut, Copy y Paste** (cortar, copiar y pegar) respectivamente, del menú **Edit**, sobre la información que contiene nuestro documento.
- Los siguientes dos botones:  , implementan las opciones **Undo y Redo** del menú **Edit**. El primer botón nos permite deshacer el último cambio realizado en la hoja de trabajo y el segundo nos permite rehacer este cambio. Maple (al igual que otras aplicaciones como Word o Excel) guarda un registro de los cambios llevados a cabo en la hoja de trabajo, de tal manera que estos pueden eliminarse oprimiendo repetidamente el botón **Undo**. Una vez eliminados pueden ser aplicados nuevamente en el orden en que se encontraban oprimiendo el botón Redo.
- Los tres botones que se encuentran a continuación tienen la siguiente utilidad:
  - El primer botón:  , nos permite insertar expresiones matemáticas dentro de la hoja de trabajo, a partir de expresiones de Maple. Este botón implementa la opción **Math Input** del menú **Insert**. Usando esta opción podemos incluir como parte del documento expresiones tales como: $\int_{-\pi}^{\pi} \sqrt{x_2}\, dx$, o bien: $\frac{\partial^3}{\partial x\, \partial y\, \partial x} \mathrm{f}(x, y)$

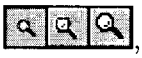
- El siguiente botón:  , nos permite insertar regiones de texto dentro del documento o bien convertir instrucciones de Maple en texto inerte. Este botón implementa la opción **Paragraf - Before** del menú **Insert** y la opción **Text** del mismo menú.

- El tercer botón:  , nos permite insertar una región de **Input** debajo de la posición del cursor, para ejecutar instrucciones de Maple.

- Los siguientes botones:  , nos permiten remover e insertar secciones y subsecciones dentro de la hoja de trabajo (las cuales se tratarán más adelante). Estos botones implementan las opciones **Section** y **Subsection** del menú **Insert,** y las opciones **Indent** y **Outdent** del menú **Format.**

- En seguida tenemos los botones:  . Cuando trabajamos en un documento con hipervínculos, Maple guarda un historial de las vínculos que hemos consultado. Estos botones nos permiten retroceder o avanzar dentro de este historial.

- A continuación aparece el botón:  . Al ejecutar una instrucción de Maple, éste se muestra activado (en color rojo) y nos permite detener esta ejecución haciendo clic sobre él.

- Los siguientes tres botones:  , nos permiten desplegar el documento actual a una escala del 100%, 150% y 200%, respectivamente.

- El siguiente botón que aparece:  , nos permite indicar a Maple cuando debe o no ocultar los caracteres no imprimibles de la hoja de trabajo (nueva línea, tabuladores, etc.).

- El penúltimo botón:  , nos permite maximizar la ventana en la cual se encuentra el documento con el que estamos trabajando actualmente, de tal forma que ocupe todo el espacio posible.

- Finalmente, el último botón:  , nos permite reiniciar maple; esto es equivalente a ejecutar la instrucción **restart** en la línea de comandos (consúltese su página de ayuda, ejecutando la instrucción **?restart**).

La barra de herramientas puede ser desplegada u ocultada usando la opción **Toolbar** del menú **View**.

1.1.3 Barra de contexto

En el caso de la barra de contexto (Context Bar), en realidad está formada por un conjunto de barras, las cuales (como su nombre lo indica) varían según el contexto del documento. A continuación se describen algunas de ellas.

Barra de contexto para regiones de texto

Cuando trabajamos en regiones de texto, la barra de contexto muestra los siguientes elementos:

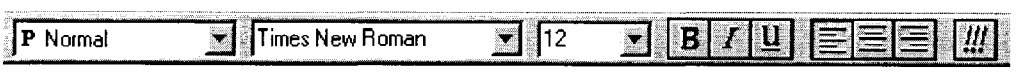
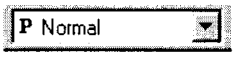


Figura 1.3: Barra de contexto para regiones de texto

Los botones y menús que aparecen en esta barra (véase la Figura 1.3) nos permiten hacer modificaciones sobre el texto que forma nuestro documento. Entre otras cosas nos permiten modificar el estilo, fuente y tamaño, además de hacer alineación del texto. A continuación se decriben cada uno de estos botones y menús.

- **Menú de estilos:** 

  Nos permite seleccionar un estilo predefinido por Maple para aplicarlo sobre el texto[1]. Por ejemplo, si marcamos con el ratón el párrafo: "Maple es un sistema que nos permite realizar cálculos de tipo símbolico", y a continuación en el menú de estilos seleccionamos **Heading 2**, obtendremos:

  "Maple es un sistema que nos permite realizar cálculos de tipo símbolico"

- **Menú de fuentes:** 

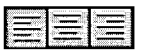
  Nos permite modificar el tipo de fuente utilizado en el documento. Esta opción se puede aplicar sobre porciones de texto ya que no afectan necesariamente a un párrafo completo.

- **Menú de tamaños:** 

  Nos permite modificar el tamaño de la fuente que estamos usando. Esta opción puede ser aplicada a palabras sencillas, no afecta necesariamente a párrafos completos.

- Los tres botones que aparecen a continuación: 

, nos permiten colocar texto en **negritas**, *cursivas* y subrayado respectivamente.

- Los siguientes tres botones:  nos permiten hacer alineaciones sobre el texto. El primero nos permite colocar texto alineado a la izquierda (esta es la alineación predeterminada). El segundo nos permite colocar texto centrado y el último nos permite alinear el texto a la derecha. Nótese que no existe una opción para hacer justificación.

- Finalmente, el botón:  , nos permite ejecutar el documento completo; es decir, al oprimirlo se ejecutarán todas las instrucciones que se encuentren dentro de una región de entrada en la hoja de trabajo. Utilizar este botón es equivalente a invocar la opción **Execute - Worksheet**, del menú **Edit**.

Barra de contexto para regiones de entrada

La Figura 1.4 muestra los elementos presentes en la barra de contexto, cuando trabajamos en regiones de entrada.

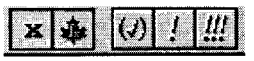


Figura 1.4: Barra de contexto para regiones de entrada

El primer botón:  , nos permite convertir, en una región de entrada, una expresión de notación de Maple a notación matemática y viceversa. Por ejemplo, si nos colocamos en la siguiente región de entrada:

[1]Es importante mencionar que los estilos predefinidos se pueden aplicar solamente sobre párrafos completos. Si se selecciona una parte de un párrafo, el estilo aplicado afectará al párrafo o la sección completa. También cabe mencionar que Maple permite al usuario definir estilos propios.

```
> alpha + beta^2;
```

y oprimimos este botón, obtendremos la siguiente expresión.

$$> \alpha + \beta^2$$

El segundo botón:  , nos permite convertir una expresión de ejecutable a no ejecutable y viceversa. Si nos colocamos en una región de entrada y oprimimos este botón, la expresión se convierte en texto inerte no ejecutable. Esta operación puede realizarse de manera inversa para convertir el texto en una expresión ejecutable.

El tercer botón:  , nos permite corregir de manera automática la sintaxis de la expresión actual. Por ejemplo, consideremos la siguiente expresión:

```
> sqrt(x^3) + cos(x))^3;
```

Como puede notarse, hace falta un paréntesis. Al oprimir el botón mencionado, esta expresión se convierte en:

```
> (sqrt(x^3) + cos(x))^3;
```

El cuarto botón:  , nos permite ejecutar la expresión actual, lo cual también puede hacerse oprimiendo la tecla [**RETURN**].

Finalmente, el último botón:  , nos permite ejecutar la hoja completa; esto es equivalente a solicitar la opción **Execute - Worksheet** , dentro del menú **Edit**. Esta opción ejecuta las instrucciones de todas las regiones de entrada existentes en el documento actual.

Barra de contexto para regiones con expresiones en notación matemática

Cuando trabajamos en una expresión escrita en notación matemática (por ejemplo, una región de salida, una región de entrada convertida a notación matemática, o bien una expresión matématica introducida como parte del documento), la barra de contexto muestra los elementos que se aprecian en la Figura 1.5

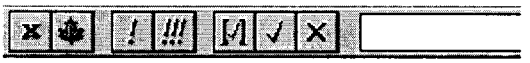


Figura 1.5: Barra de contexto para regiones con expresiones en notación matemática

La utilidad de los primeros cuatro botones es la misma que se explicó en la sección anterior. Para ver la utilidad de los siguientes tres botones considérese la siguiente expresión en notación matemática:

$$> \frac{\partial}{\partial x}\left(2x - \sin(x-5)\right)$$

Si nos colocamos dentro de la expresión y seleccionamos, por ejemplo, la primera ocurrencia de x, al oprimir el botón:  , podemos sustituir todas las ocurrencias de esta variable por el contenido del campo de edición.

En cambio el botón:  , nos permite cambiar únicamente la expresión seleccionada por el contenido del campo de edición, esto también puede hacerse escribiendo la nueva expresión en el campo de edición y oprimiendo la tecla [**RETURN**].

Por último, el botón:  , nos permite cancelar la edición de una expresión de este tipo.

Existen barras de contexto para otro tipo de regiones, estas se tratarán más adelante en los capítulos correspondientes. A continuación se enumeran y se proporciona la instrucción para consultar la página de ayuda de cada una de ellas.

- Barra de contexto para gráficas en dos dimensiones. Para ver su hoja de ayuda ejecútese: **?worksheet,reference,context2dplot.**
- Barra de contexto para gráficas en tres dimensiones. Para consultar su página de ayuda ejecute: **?worksheet,reference,context3dplot.**
- Barra de contexto para animaciones. Su hoja de ayuda se puede consultar con la instrucción: **?worksheet,reference,contextanimate.**
- Barra de contexto para hojas de cálculo. Para ver su hoja de ayuda se debe ejecutar: **?worksheet,reference,contextspread.**

Esta barra también puede ocultarse o desplegarse usando las opciones del menú **View**. Su página de ayuda puede consultarse por medio de la instrucción: **?worksheet,reference,contextbar.**

1.1.4 Barra de estado

La barra de estado (Status Bar) está formada por la pequeña barra que aparece en la parte inferior de la ventana de la aplicación, como su nombre lo indica, esta barra muestra el estado en el cual se encuentra Maple en todo momento. En esta zona, entre otras cosas, se despliegan mensajes de diagnóstico, el tema de las hoja de ayuda que se está consultando, e información del sistema; tal como el tamaño del documento actual y el tiempo de procesador ocupado.

Por ejemplo, al oprimir un botón en la barra de herramientas, se despliega un pequeño mensaje en la línea de estado que indica el tipo de operación ejecutada sobre el documento.

Este elemento también puede ser ocultado o mostrado usando la opción **Status bar** del menú **View**. Ejecútese la instrucción **?worksheet,reference,statusline** para obtener su página de ayuda.

1.1.5 Paletas

Las ***"paletas"*** (Palettes) son elementos de la interfaz de Maple que nos proporcionan un conjunto de botones útiles para introducir símbolos predefinidos, expresiones, operadores, matrices y vectores en el documento actual. Estas paletas son desplegadas en forma de ventanas independientes, mediante la opción **Palettes** del menú **View**; su función es facilitar al usuario la introducción de expresiones matemáticas por medio del uso de botones. Por ejemplo, para calcular una integral definida, nos colocamos en una región de entrada y en la **Paleta de Expresiones** oprimimos el botón:  , a continuación, en la región de entrada donde nos encontramos aparecerá la siguiente expresión:

```
> int(%?, %?=%?..%?);
```

en la cual debemos escribir la función a integrar, la variable de la cual depende y los límites de integración.

Las paletas disponibles en esta versión de **Maple** son:

- **Paleta de símbolos**. Ésta nos permite usar, en las expresiones introducidas, símbolos y letras griegas.
- **Paleta de expresiones**. Nos permite construir expresiones que contengan integrales, derivadas, sumas, productos, límites y algunas funciones tales como exponencial, logaritmo, seno y coseno.

- **Paleta de matrices** . Nos permite construir matrices de diferentes dimensiones.
- **Paleta de vectores.** Nos permite insertar vectores columna y renglón en el documento que estamos trabajando.

1.2 Estructura de la Hoja de Trabajo

El tipo de documento generado durante una sesión de Maple es conocido como ***"hoja de trabajo"*** (***"worksheet"***). Toda hoja de trabajo puede estar formada por varias de las regiones que se describen a continuación.

1.2.1 Regiones de entrada

Las regiones de entrada (Input Region) son aquellas en las cuales se introducen las instrucciones que debe ejecutar Maple. Éstas aparecen delimitadas por los símbolos [>. Los datos que se colocan en estas zonas siempre son interpretados y ejecutados por Maple al oprimir la tecla [**RETURN**], o bien al hacer clic en el botón:  , de la barra contextual. Por ejemplo, realizaremos a continuación una operación aritmética. Para poder ejecutar esta instrucción simplemente colocamos el cursor de texto sobre la instrucción (en cualquier posición) y a continuación oprimimos la tecla [**RETURN**] o hacemos clic en el botón de ejecución mencionado.

```
> 4 + 5; # Oprima enter para ejecutar
                                    9
```

Como se mencionó anteriormente, siempre que oprimimos la tecla [**RETURN**] en una región de entrada (o hacemos clic en el botón) automáticamente Maple interpreta y ejecuta las instrucciones tecleadas en ésta. Sin embargo, es posible teclear una instrucción (o varias) en dos o más líneas, basta con oprimir las teclas [**SHIFT**][**RETURN**]; esta combinación le indica a Maple que debe insertar una nueva línea de entrada inmediatamente abajo, sin ejecutar la expresión. De esta manera se pueden escribir instrucciones de varias líneas y las expresiones contenidas en ellas no se ejecutan hasta que se oprime un [**RETURN**] sencillo en alguna de las líneas (o se hace clic en el botón de ejecución).

Las regiones de entrada pueden ser colocadas antes o después de la posición actual del cursor. Existen varias formas de insertar una de estas regiones:

- **Antes del cursor.** Se puede usar la opción **Execution Group - Before Cursor** del menú **Insert.**
- **Despues del cursor.** Se puede usar la opción **Execution Group - After Cursor** del mismo menú. También se puede usar el botón: , de la barra de herramientas. Al oprimirlo, automáticamente se inserta una región de entrada debajo del cursor.

Este tipo de operaciones pueden ser ejecutadas también desde el teclado usando las combinaciones de teclas conocidas como ***"Hot Keys"***. Por ejemplo, para insertar una región de entrada debajo del cursor se puede usar la combinación de teclas [**CTRL**]-[**J**], y para insertar una región arriba del cursor se puede teclear [**CTRL**]-[**K**]. Para obtener información acerca de todas las ***hot keys*** existentes en esta versión de Maple se puede consultar la hoja de ayuda mediante las instrucciones:

?worksheet,reference,hotmac

?worksheet,reference,hotunix

?worksheet,reference,hotwin

Para las versiones de Macintosh, UNIX y Windows, respectivamente.

En las regiones de entrada también es posible escribir comentarios, los cuales son tratados por Maple como texto inerte. Para poder colocal un comentario simplemente se coloca el símbolo "#" en la región de entrada y a continuación el texto, como se muestra en el siguiente ejemplo:

```
> # Este es un comentario.  Oprima enter
```

Maple no interpreta, en las regiones de entrada, el texto que se encuentra después del símbolo "#", por lo cual se puede colocar cualquier palabra que se desee dentro de los comentarios.

Esta aplicación asigna de manera predeterminada la fuente **COURIER NEW** en tamaño 12, en color rojo, para el texto que se utiliza dentro de las regiones de entrada. Es posible cambiar esta fuente usando la opción **Styles**del menú **Format**. Esta opción nos proporciona una ventana con los distintos estilos usados en los documentos de Maple. En el caso de las regiones de entrada, para hacer alguna modificación al tipo de fuente, color, tamaño, etc., se debe modificar el estilo **Maple input**.

1.2.2 Regiones de salida

Las regiones de salida (Output Region) son aquellas en las cuales se despliega el resultado de las instrucciones ejecutadas por Maple. Por ejemplo, ejecutemos la siguiente instrucción:

```
> 3*4;  # Presione enter para ejecutar
                                    12
```

Como puede verse, el resultado se despliega inmediatamente después de la instrucción ejecutada; ésta es la ***"región de salida"***.

El tipo de fuente, el tamaño y color de la salida también puede ser modificado usando la opción **Styles** del menú **Format**; en este caso, el estilo que debe modificarse es **2D Output**.

Manipulación de las regiones de salida

Existen varios detalles importantes acerca de las regiones de salida. Maple nos permite manipular estas regiones de diferentes formas. Por ejemplo, podemos modificar el aspecto, color, tipo y tamaño de la fuente. Además, también podemos ***"reutilizar"*** las expresiones que se generan como resultado de la ejecución de las instrucciones. Describiremos esto a continuación.

Forma tipográfica de la salida

La forma en la que Maple nos presenta los resultados es controlable en varios aspectos. Anteriormente se habia mencionado que es posible modificar el tamaño, tipo y color de la fuente usada tanto en las regiones de entrada como en las de salida; además, también es posible usar varios formatos para esta última. Para seleccionar uno de estos formatos podemos usar la opción **Preferences** del menú **File**; esto deplegara la ventana de preferencias, en la cual aparecen varias fichas, seleccionamos la que aparece con el nombre **I/O Display** y en la sección **Output Display** podemos indicar el tipo de salida que deseamos. Por ejemplo, si solicitamos **Maple Notation**, obtenemos una salida como la siguiente[2]:

```
> sqrt(25*x);

5*x^(1/2)

> Int(x^2 - 2*x, x);

Int(x^2-2*x,x)
```

[2]**Int** es la forma ***"inerte"*** de la función **int** que calcula la integral de una función. Véase la página de ayuda, ejecutando la instrucción **?int**.

La ventana de preferencias desplegada se muestra en la Figura 1.6.

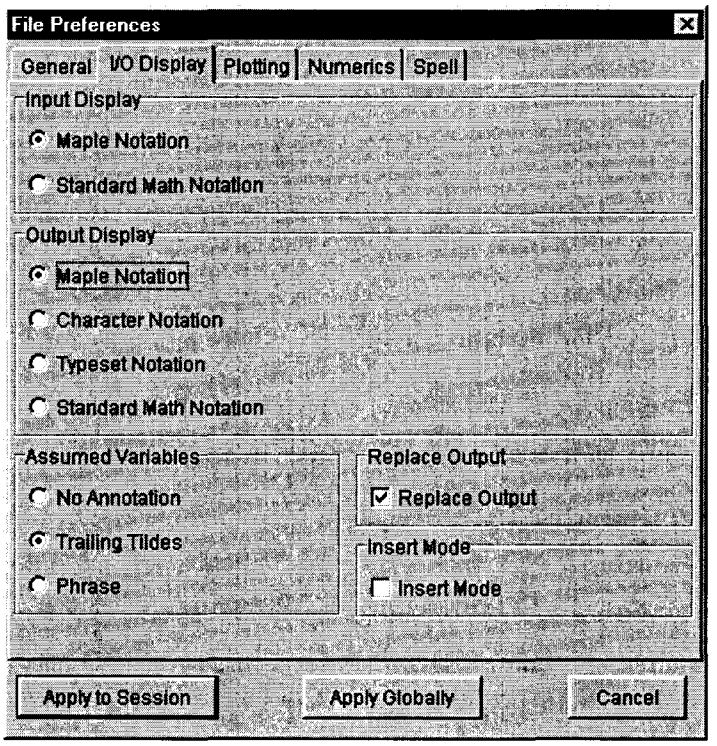


Figura 1.6: Ventana de preferencias

En cambio, si solicitamos **Character Notation**, obtenemos:

```
> sqrt(25*x);
```

```
                                  1/2
                               5 x
```

```
> Int(x^2 - 2*x, x);
```

```
                                 /
                                 |   2
                                 |  x  - 2 x dx
                                 |
                                /
```

Otra opción disponible es **Typeset Notation**, con la cual obtenemos:

```
> sqrt(25*x);
```

$$5\sqrt{x}$$

```
> Int(x^2 - 2*x, x);
```

$$\int x^2 - 2\,x\,dx$$

La opción predeterminada es **Standard Math Notation**:

```
> sqrt(25*x);
```

$$5\sqrt{x}$$

```
> Int(x^2 - 2*x, x);
```

$$\int x^2 - 2\,x\,dx$$

Estos últimos dos estilos nos proporcionan salidas en notación matemática, por lo cual generalmente son los más utilizados; sin embargo, existe una diferencia entre ellos. En ambos casos la salida generada puede ser modificada en la barra contextual; pero en el caso de **Standard Math Notation**, es posible manipular directamente, en la región de salida, cada una de las partes de ésta, para así obtener una nueva expresión. En cambio, en la generada por el estilo **Typeset Notation** no es posible manipular cada una de sus partes por separado directamente en la región de salida, solo se puede modificar toda la expresión completa.

Reutilización de la salida en la entrada

Es posible utilizar las expresiones que Maple genera como resultado de la ejecución de una instrucción, para incluirlas dentro de nuevas instrucciones. La salida (puede ser toda una expresión o solo una parte, dependiendo de como fue generada) puede ser seleccionada, copiada y pegada a una nueva zona de entrada. Por ejemplo, considérese la siguiente instrucción:

```
> (a + b + sqrt(delta + 1))^2;
```

$$(a + b + \sqrt{\delta + 1})^2$$

Seleccionamos con el ratón (en la región de salida) la raíz cuadrada de delta más uno, la copiamos usando las opciones del menú de edición (o bien usando los botones de la barra de herramientas) y finalmente la pegamos en la siguiente zona de entrada:

```
> (delta + 1)^(1/2)
```

Nótese que, a pesar de estar manipulando una región de salida, este sistema convierte la expresión a su correspondiente instrucción de entrada en notación de Maple.

Eliminación de las regiones de salida

Cuando trabajamos en una hoja de Maple, en ocasiones no es deseable para el usuario guardar la hoja con todas las regiones de salida, sobre todo si se considera que generalmente las salidas ocupan un espacio considerablemente más grande que las instrucciones que las generaron (esto es especialmente notable cuando las regiones de salida contienen gráficas o animaciones). Es posible eliminar estas regiones usando la opción **Remove Output** del menú **Edit**. Además esta opción nos permite, ya sea eliminar las salidas de toda la hoja, o bien seleccionar una parte y eliminar solo las salidas de esta selección.

Cambio de estilos y colores en las regiones de salida

Es posible modificar el estilo del texto usado en las regiones de salida. Esto se hace usando la opción **Styles** del menú **Format**. Maple nos proporciona un conjunto de estilos predefinidos que son asignados por defecto a las diferentes partes del documento. Estos mismos estilos pueden ser asignados manualmente por el usuario a cualquier región de la hoja. Simplemente se marca la región a la cual se le desea cambiar de estilo y se aplica uno nuevo, el cual puede ser seleccionado de la barra de contexto. Además de los estilos predefinidos, Maple nos permite crear estilos personalizados. Por ejemplo, si elegimos en la ventana de la opción **Styles** del menú **Format** el botón **Paragraph**, podemos crear un nuevo estilo aplicable a párrafos de texto, en el cual podemos incluir una fuente particular, un color para el texto, guiones al inicio, indentaciones, alineaciones, etc.

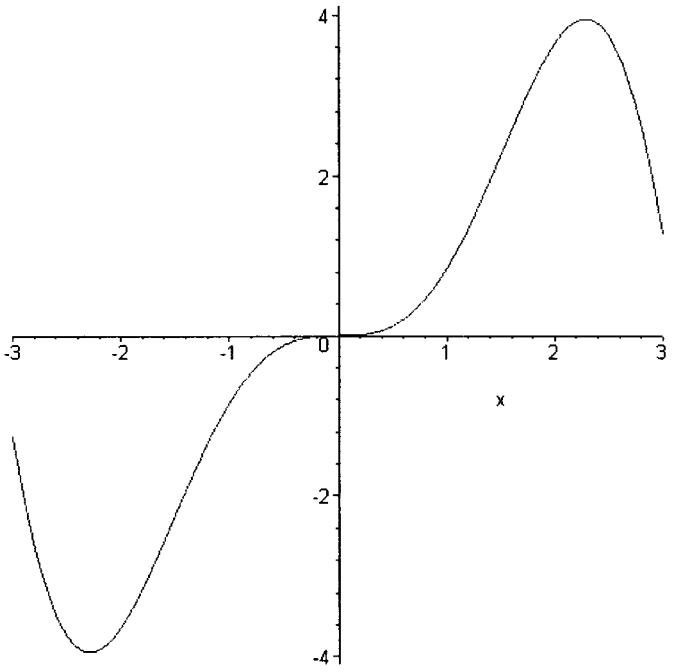
1.2.3 Regiones de gráficas y animaciones

Gráficas

Otro tipo de regiones que pueden aparecer en una hoja de trabajo son aquellas en las cuales Maple despliega las gráficas en 2 y 3 dimensiones.

Por ejemplo, tenemos la siguiente gráfica.

```
> plot(x^2*sin(x), x=-3..3); # Oprima return para ejecutar
```



Nota: plot despliega la gráfica de una función de una variable, este tipo de funciones se tratarán posteriormente, de la misma forma que los detalles acerca de la manipulación de estas regiones.

Por omisión, el despliegue de gráficas y animaciones se hace ***"en línea"***, es decir, inmediatamente abajo de la instrucción que la generó y como parte de la hoja de trabajo. Esto puede ser modificado de tal manera que las gráficas y animaciones no formen parte de la hoja de trabajo sino que se desplieguen en una ventana independiente. Para hacer esto se debe usar la opción **Preferences** del menú **File**; en la ventana desplegada por esta opción, dentro de la etiqueta **Plotting** aparece una zona con el nombre **Plot Display**; en ésta se puede seleccionar **Window** para que las gráficas (y animaciones) aparezcan en una ventana independiente. La opción predeterminada es **Inline**, con la cual estas regiones aparecen como parte de la misma hoja que las instrucciones para generarlas.

Animaciones

Otra facilidad que nos proporciona esta aplicación es la de producir animaciones de funciones, los detalles sobre la generación de éstas se abordarán posteriormente. Al igual que en las gráficas, en el caso de las

animaciones es posible cambiar varios aspectos de la presentación. Por ejemplo, se puede modificar el punto de vista del observador, el color, la iluminación incidente sobre las gráficas, el estilo, los ejes y el número de cuadros utilizados, entre otras cosas. También, con la opción **Preferences** del menú **File**, es posible hacer que la animación aparezca en la misma hoja de trabajo o en una ventana independiente (esto es análogo al caso de las regiones gráficas), en ambos casos es posible manipular el despliegue.

La intención de esta subsección no es entrar en detalles acerca de este otro tipo de regiones manejadas por Maple. Existen varios aspectos importantes acerca de éstas pero no serán tratados ahora, éstos se abordarán posteriormente en el capítulo correspondiente.

1.2.4 Regiones de Texto

Las regiones de texto (Text Region) son áreas de la hoja de trabajo en las cuales se pueden colocar comentarios, explicaciones y en general texto inerte que Maple no ejecuta ni interpreta. Estas regiones pueden ser manejadas de la misma forma que en un editor de texto; es posible colocar acentos, simplemente se oprime la tecla del acento y a continuación la vocal a acentuar: á, é, í, Ó, Ú (en la versión de Maple para UNIX el teclado debe estar configurado para poder escribir letras con acentos). También es posible alinear texto, usando la opción **Paragraph** del menú **Format**, o bien, por medio de los botones de la barra de contexto.

Otra facilidad es que, en estas regiones pueden incluirse expresiones en notación matemática como parte del texto; por ejemplo, podemos colocar la siguiente ecuación: $x^2 + 2\,x = 4\,x^3 - \cos(e^{\sqrt{5}})$. Este tipo de expresiones pueden ser incluidas usando la opción **Standard Math** del menú **Insert**, o bien, usando el botón , de la barra de herramientas descrito con anterioridad. Una vez solicitada esta opción se procede a escribir la expresión que se desea en la notación de Maple y ésta se convertirá a notación matemática. Por ejemplo, para obtener la ecuación anterior se utilizó la expresión de Maple: `x^2 + 2x = 4 x^3 - cos(exp(sqrt(5)))`

El estilo de una región de texto también puede ser modificado; para ello se marca la zona que se desea modificar y a continuación se usa la barra de contexto o bien la opción **Character** del menú **Format**. Esto nos permite cambiar el tipo de fuente, el estilo, el tamaño y el color del texto seleccionado.

1.2.5 Hojas de Cálculo

Otro tipo de elementos que pueden estar presentes en un documento de Maple son las hojas de cálculo. Al respecto cabe mencionar que Maple 8 puede ser usado como un agregado de Excel 2000, lo cual nos da la ventaja de poder hacer llamadas a funciones de Maple desde hojas de cálculo de Excel. Además, es posible copiar y pegar información entre estas dos aplicaciones, con lo cual podemos incluir segmentos de hojas de Excel en hojas de cálculo de Maple y viceversa. Podemos tener acceso a un conjunto de páginas de ayuda de funciones matemáticas de Maple desde Excel, y al realizar llamadas a funciones de Maple, desde Excel, podemos tener acceso a un asistente, el cual nos facilitará la creación de la llamada.
Para insertar una hoja de cálculo en Maple, usamos la opción **Spreadsheet**, del menú **Insert**.

Consúltese la hoja de ayuda de la hoja de cálculo para obtener más información sobre este tipo de elementos, ejecutando la instrucción: **?worksheet,reference,spreadmenu**.

1.3 Secciones y subsecciones

Maple también proporciona al usuario diversas opciones útiles para generar documentos presentables, interactivos y organizados. Además de permitir modificaciones sobre el texto, también es posible dividir el documento en secciones independientes, estructuradas jerárquicamente, con lo cual se pueden crear hojas de Maple bien organizadas. Otra característica interesante de esta aplicación es el soporte que proporciona para el manejo de marcadores e hipervínculos, lo cual facilita la creación de documentos interactivos.

Toda la información de una hoja de trabajo puede ser dividida en secciones independientes, cada una de las cuales se encuentra delimitada por un corchete en la parte izquierda de la hoja de trabajo. Cada sección está siempre marcada al inicio por un pequeño botón y a la derecha de éste generalmente se coloca el nombre de dicha sección. Este botón tiene la particularidad de permitirnos ***"expandir"*** o ***"colapsar"*** la sección a la que pertenece, ocultando o mostrando la información que contiene.

Para poder crear una nueva sección o subsección se pueden utilizar la opciones **Section** y **Subsection** del menú **Insert**. Por ejemplo, si solicitamos la opción **Subsection** del menú **Insert**, obtenemos un pequeño botón como el que se muestra a continuación:

Este botón representa un nueva sección dentro del presente documento, en la cual se puede colocar cualquier tipo de información: regiones de texto, gráficas, animaciones, regiones de entrada, de salida y hojas de cálculo. Es bastante útil colocar un título a cada botón para poder distinguir cada una de las secciones que componen nuestra hoja de trabajo. Para hacer esto simplemente se coloca el cursor a la derecha del botón y se escribe el título.

Una nueva subsección (1)

Ahora, para colocar información dentro de una sección se coloca el cursor a la derecha del botón y se oprime la tecla [**RETURN**], en este momento la sección se expande y podemos comenzar a teclear el contenido de ésta.

Una nueva subsección (1)

"Aqui podemos colocar texto, instrucciones, gráficas, animaciones, etcétera."

Como se mencionó antes, dentro de cada sección es posible insertar regiones de entrada, texto, gráficas, animaciones y hojas de cálculo.

Una vez que nos encontramos dentro de una sección, si solicitamos la opción **Subsection** del menú **Insert**, se inserta una nueva subsección dentro de la sección en la cual estamos. En cambio, si solicitamos la opción **Section** en el menú **Insert**, lo que obtenemos es una nueva subsección, pero no dentro de la que estamos, sino al mismo nivel que la subsección actual:

Una nueva subsección (1)

"Aqui podemos colocar texto, instrucciones, gráficas, animaciones, etcetera."

Una nueva subsección (2)

Una característica importante de las secciones y subsecciones de Maple, es que nos permiten ocultar o mostrar su contenido. Para hacer esto simplemente colocamos el ratón sobre el botón de la sección o subsección y hacemos clic sobre él. Al ***"colapsar"*** una sección aparece sobre el botón un signo "+"; en cambio, cuando la subsección se encuentra ***"expandida"*** aparece el símbolo "-".

Cuando nos encontramos en un documento con varias secciones y/o subsecciones, podemos expandirlas o colapsarlas todas por medio de las opciones **Expand All Sections** y **Collapse All Sections**, respectivamente, del menú **View**.

1.4 Marcadores

Dentro de una hoja de trabajo es posible asignar marcas especiales a elementos del documento, que permiten crear accesos rápidos y directos a éstos. Por medio de ***"marcadores"*** (***"bookmarks"***), es posible posicionar automáticamente el cursor en la línea a la cual fueron asignados, sin importar en que lugar se encuentre la referencia a dicho marcador.

Es importante tener en cuenta que estos marcadores pueden ser definidos en regiones de texto, regiones de entrada, regiones de salida, gráficas en dos y tres dimensiones, animaciones y hojas de cálculo.

1.4.1 Creación de un marcador

Para crear un marcador es necesario seguir los siguientes pasos:

- Colocamos el cursor en el lugar donde deseamos que se encuentre el marcador. Si se trata de una gráfica, una región de salida o una hoja de cálculo, simplemente la seleccionamos.
- A continuación se solicita, dentro del menú **View,** la opción **Bookmarks** y en seguida la opción **Edit Bookmark**, esto desplegará la ventana que se muestra en la Figura 1.7.

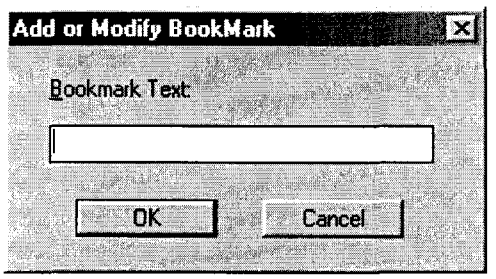


Figura 1.7: Ventana para creación de marcadores

- Como siguiente paso, en el campo **Bookmark Text** se debe colocar una palabra o frase diferente a cada uno de los bookmarks, la cual permanecerá en el submenú **Bookmarks** del menú **View**, para futuras referencias.

Una vez que ya está definido el marcador, para que el cursor se desplace a la región a la cual se asignó, simplemente se solicita la opción **Bookmarks** del menú **View** y a continuación se selecciona la palabra o frase perteneciente al marcador que se desea. Una vez hecho esto, el cursor aparecerá al inicio de la región con la cual está asociado dicho marcador.

Una característica importante de los marcadores es que permiten acceder a las líneas a las cuales fueron asignados, sin importar que éstas esten dentro de una sección o subsección cerrada. Este tipo de elementos son útiles para facilitar la navegación a través de la hoja de trabajo.

1.5 Hipervínculos

Un **"hipervínculo"** (**"hyperlink"**) es una palabra o frase a través de la cual podemos vincular una parte de una hoja de trabajo, ya sea con otra hoja creada por el usuario, con una de las hojas que conforman el sistema de ayuda de Maple, con un marcador definido previamente o bien con una dirección de WWW. Cada hipervínculo está formado por un texto subrayado, el cual al ser pulsado nos lleva directamente al

lugar al que apunta (es decir, al lugar con el cual está vinculado). Esta operación se realiza sin importar que las hojas señaladas se encuentren inactivas o aun no hayan sido abiertas para su uso. Los hipervínculos de Maple funcionan de la misma forma que en las paginas de Web. Para colocar uno de estos elementos seguimos los siguientes pasos:

- Posicionamos el cursor en el lugar donde deseamos colocar el hipervínculo, sin seleccionar ningún elemento del documento.
- Solicitamos la opción **Hiperlink** del menú **Insert**. A continuación aparecerá la ventana que se muestra en la Figura 1.8

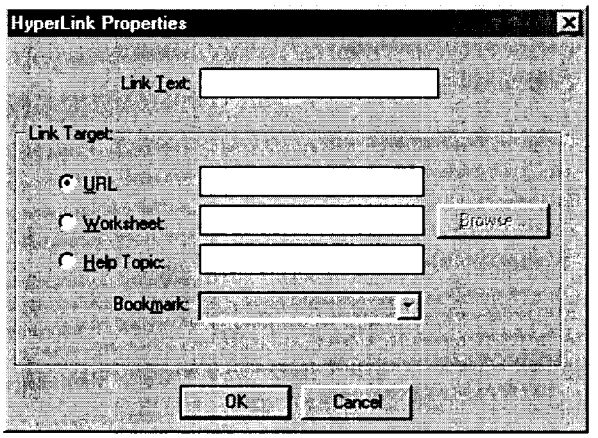


Figura 1.8: Ventana para edición de Hipervínculos

- En el campo **Link text** de esta ventana colocamos la palabra que aparecerá como texto del vínculo.
- A continuación, seleccionamos la opción **URL**, **Worksheet** o **Help Topic**, dependiendo de si queremos colocar un vínculo a un documento de WWW, a un archivo de Maple o a una hoja del sistema de ayuda, respectivamente.
- Colocamos la dirección de WWW, el nombre del archivo o el tema del sistema de ayuda con el cual estará asociado nuestro vínculo.
- Finalmente presionamos el botón **OK** y a continuación aparecerá el vínculo en el sitio indicado de nuestro documento.

Para insertar un hipervínculo que apunte a un marcador se debe proceder de la misma forma. Solamente se selecciona en la ventana de diálogo la opción **Worksheet** y a continuación, en el menú **Book Mark**, se selecciona el marcador al cual debe apuntar el vínculo. En este caso, Maple nos presentará una lista de los marcadores que han sido colocados anteriormente, solamente tenemos que tomar el que nos interesa.

El texto donde aparece el hipervínculo se encuentra resaltado con otro color y además subrayado. Al hacer clic sobre este texto, automáticamente Maple localiza el elemento con el cual está asociado el vínculo y lo muestra al usuario. En caso de que el vínculo apunte a un sitio de WWW, Maple iniciará un navegador para tener acceso a este sitio.

El color en el que se muestran los vínculos puede ser modificado mediante la opción **Styles** del menú **Format**. El estilo que debe modificarse es **Hyperlink**.

Capítulo 2

Elementos Básicos de Maple

Una vez que hemos iniciado una hoja de trabajo podemos comenzar a insertar regiones de entrada e instrucciones para su ejecución. Existe un conjunto de elementos básicos de Maple, en base a los cuales se construyen las instrucciones que ejecuta. Entre ellos se encuentran los números, las constantes, cadenas y nombres asignados a expresiones. Así mismo, existen varias reglas sintácticas que deben tenerse en cuenta al teclear dichas instrucciones. Todo esto es importante ya que son los fundamentos para poder realizar operaciones en este sistema.

A continuación se presentan los principales elementos y reglas que se deben considerar al desarrollar una hoja de trabajo en Maple.

2.1 Mayúsculas y minúsculas

Antes de comenzar a ejecutar instrucciones debemos tener en cuenta que Maple hace distinción entre mayúsculas y minúsculas. Así, la palabra ***"solve"*** es diferente de ***"Solve"*** y de ***"SOLVE"***. Se debe tener esto en cuenta al manipular regiones de entrada, pues es precisamente la información tecleada en estas regiones la que se interpreta.

2.2 Terminadores de instrucciones

Una instrucción en Maple está formada por una o más expresiones terminadas con un símbolo que indica su final. Este sistema utiliza dos símbolos como ***"terminadores"*** de instrucciones de entrada, el punto y coma ";", y los dos puntos ":". Toda instrucción debe ser terminada con uno de los dos símbolos; éstos también pueden ser utilizados para separar dos o más instrucciones que sean introducidas en una misma línea. Ambos terminadores tienen casi la misma función salvo por una diferencia muy importante:

- **Punto y coma** (;). Al utilizar este terminador en una instrucción, una vez que la ejecutamos, Maple nos muestra inmediatamente el resultado de dicha instrucción.
- **Dos puntos** (:). Al utilizar los dos puntos, Maple ejecuta la instrucción pero no nos muestra ningún resultado. Esto puede ser bastante útil cuando se realiza un cálculo cuyo resultado es demasiado grande o simplemente éste no se desea visualizar (por ejemplo, si solo se desea conservar su valor en una variable).

Es importante mencionar que siempre se debe colocar uno de estos caracteres como terminador, de lo contrario Maple nos mostrará un mensaje de error debido a que no pudo determinar el final de la instrucción.

A continuación veremos algunos ejemplos de su uso:

```
> 3*5; # En esta instrucción, Maple nos muestra el resultado
```

$$15$$

```
> 3*5:  # Se ejecuta la instrucción pero no se despliega el resultado

> 3*5  # Aparece un mensaje de error debido a la falta de terminador
Warning, premature end of input

> 3 + 2; 2 - 4; 4*5:  # Estos símbolos también se usan como separadores
> de instrucciones
```

$$5$$
$$-2$$

2.3 Referencia a resultados anteriores

Podemos hacer referencia al resultado de la última instruccion ejecutada usando el símbolo "%". De la misma forma se puede invocar el penúltimo y antepenúltimo resultado con los símbolos "%%" y "%%%", respectivamente. Por ejemplo:

```
> a + b;
```

$$a+b$$

```
> % + 1;     # el último resultado obtenido, más 1
```

$$a+b+1$$

```
> %% + 2;    # el penultimo resultado más 2
```

$$a+b+2$$

```
> %%% + 3;  # el antepenúltimo resultado más 3
```

$$a+b+3$$

```
> %% + %%%; # el penúltimo resultado más el antepenúltimo
```

$$2a+2b+3$$

Nota: Al utilizar el operador "%" se debe tener en cuenta que Maple reevalúa la última expresión ejecutada. De la misma forma, los operadores "%%" y "%%%" reevaluan la penúltima y antepenúltima expresión, respectivamente. Esto debe tomarse en cuenta cuando se ejecutan expresiones complejas ya que al referenciarlas a través de los operadores mencionados se tendrán que volver a evaluar.

Una manera de evitar estos cálculos (muchas veces innecesarios), es haciendo uso de variables para guardar los resultados, ya que al referenciarlas no es necesario volver a calcular el valor que tienen asignado.

Maple no nos permite, mediante el uso de estos operadores, hacer referencia a un resultado antes del antepenúltimo. Esto solo puede hacerse volviendo a ejecutar la instrucción deseada o bien usando variables, las cuales se tratarán posteriormente.

2.4 Tipos de datos básicos

Maple trabaja con diferentes tipos de datos, entre los cuales se encuentran los siguientes:

2.4.1 Números

Cualquier dato formado por un valor numérico. Maple reconoce los siguientes tipos de números:

- Enteros y racionales

```
> 34; 123; 28;
```

$$34$$
$$123$$
$$28$$

```
> 1/4; 2/5; 6/3*3/8;
```

$$\frac{1}{4}$$
$$\frac{2}{5}$$
$$\frac{3}{4}$$

En las operaciones que involucran racionales Maple realiza - de manera predeterminada - todas las operaciones usando aritmética racional. Esto es, si se le pide calcular la siguiente operación:

```
> 23/4*(12/5 - 24);
```

$$\frac{-621}{5}$$

Nótese que en ningún momento se realizan las divisiones, todo el cálculo se hace en forma racional. Esta característica de Maple evita los problemas que puedan surgir al hacer redondeo de los operandos. Por ejemplo, considérese la siguiente instrucción:

```
> 1/3*25!;
```

$$5170403347776995328000000$$

Este resultado se obtiene al operar usando racionales. Ahora introducimos la misma operación pero realizando la operación 1/3 y calculando la diferencia entre los resultados. Para ello usaremos la función **evalf**, la cual recibe una expresión aritmética y nos da su valor como número de punto flotante (véase su hoja de ayuda ejecutando **?evalf**).

```
> evalf(1/3)*25!;
```

$$0.5170403346\,10^{25}$$

```
> %% - %;
```

$$0.2\,10^{16}$$

Nótese que la diferencia que existe entre la operación con aproximación (1/3 = .333333333...) y la operación racional es considerablemente grande. Esta es la razón por la cual Maple no maneja los números como reales, a menos que se le obligue a hacerlo. Posteriormente se tratarán con más detalle este tipo de operaciones.

- Irracionales. Maple tiene la capacidad de trabajar también con este tipo de números, por ejemplo:

```
> sqrt(sqrt(2)*exp(1));
```

$$\sqrt{\sqrt{2}e}$$

En este caso tenemos una operación que incluye a dos números irracionales, $\sqrt{2}$ y e los cuales son manejados de manera simbólica en las operaciones, a menos que se pida una aproximación de punto flotante con **evalf**.

- Números de punto flotante

```
> 18.345^4; 239.734*785.4;
```

113258.5153
188287.0836

Para este tipo de números (y en general para cualquier operación numérica), usando la función **evalf** es posible obtener una aproximación con una precisión arbitraria. Por ejemplo, podemos solicitar el cálculo siguiente con 500 digitos de precisión:

```
> evalf(sqrt(2)*Pi-exp(1),500);
```

1.72460110969932101165559351870803120085737459567573064811842797923235816274592631600394475623959880993594382075322573144254218492171658391947501206441381939254114649112793404235021983\
12205533268766340079036083607903732562279585178253140936050\
22470406362837069166459625837507891323849788988768162241759\
80200003499552716480762867419919927025826673675952134011885\
29558323211481988601723782556998659909661969755755798449514\
69197203997060003531791738757928968615973404616010629217618\
8652706870884120109392

La función **evalf** también puede ser usada para obligar a Maple a devolver el resultado de un cálculo en forma de punto flotante. Otra forma de hacer esto es incluyendo, como parte de la expresión, al menos un número de punto flotante.

```
> 39.0 - 234.0^2 + sqrt(23.8);
```

$$-54712.12148$$

- Números complejos e imaginarios. Para hacer uso de estos números, Maple utiliza la constante simbólica I que representa $\sqrt{-1}$. Esta constante se encuentra ya predefinida como parte del sistema.

```
> (2 + 3*I)*(4 - 2*I)*(I);
```

$$-8 + 14\,I$$

En el caso de los números complejos, estos deben ser encerrados entre paréntesis para poder operarlos correctamente, además la parte imaginaria debe ser expresada como un producto explícitamente.

2.4.2 Constantes

Cualquier tipo de dato formado por un valor constante. Por ejemplo las constantes numéricas o las simbólicas predefinidas, como se muestra a continuación:

- Constantes numéricas. Están formadas por cualquier dato de tipo numérico:

```
> 23; 4.987; (3.5 + 4*I);
```

$$23$$
$$4.987$$
$$3.5 + 4.\,I$$

- Constantes simbólicas. Maple nos proporciona un conjunto de símbolos predefinidos que pueden ser invocados en cualquier expresión, entre los cuales se encuentran:

  - **Pi**. Representa el número π, un valor aproximado para éste es 3.1415926535...
    Este tipo de aproximación puede ser mejorada utilizando la instrucción **evalf**.
  - **I**. Representa la constante $i = \sqrt{-1}$, usada para expresar números complejos e imaginarios.
  - **infinity**. Representa el símbolo *"infinito"* o *"infinito positivo"* denotado en matemáticas por ∞.
  - **-infinity**. Representa el símbolo *"-infinito"* o *"infinito negativo"*, denotado por $-\infty$
  - **gamma**. Representa a la **Constante de Euler**, comúnmente denotada por γ. Su valor aproximado es: .5772156649.
  - **Catalan**. **Constante de Catalan**. Su valor aproximado es .9159655942.
  - **true, false**. Se utilizan para representar los valores booleanos de falso y verdadero.

  Al invocar este tipo de datos debe recordarse que Maple hace distinción entre mayúsculas y minúsculas.

2.4.3 Expresiones

Las ***"expresiones"*** son uno de los elementos básicos utilizados por Maple. Todas las instrucciones están formadas por expresiones. Como definición podemos decir que **"todo conjunto de caracteres que no viole la sintaxis de Maple es considerado una expresión"**. Veamos algunos ejemplos:

Expresiones válidas

En general, las expresiones están formadas por combinaciones alfanuméricas sin espacios o paréntesis sin cerrar (es recomendable no usar símbolos como #, &, %, @, !, \). Veamos los siguientes ejemplos:

- Palabras solas que no contienen signos de operaciones aritméticas. Nótese que es válido usar guiones bajos para unir palabras. También puede incluirse el caracter **"?"**, siempre que no se encuentre en la primera posición.

```
> ab001drl3k879uiy_3?;
```

$$ab001drl3k879uiy\_3?$$

```
> hola_usuarios_de_Maple;
```

$$hola\_usuarios\_de\_Maple$$

  En las expresiones también debe tomarse en cuenta que Maple distingue mayúsculas y minúsculas. Por ejemplo, ejecutemos las siguientes operaciones:

```
> A - A;
```

$$0$$

```
> A - a;
```

$$A - a$$

- Nombres con operaciones aritméticas; es decir, combinaciones de nombres alfanuméricos con signos de operación e instrucciones bien definidas (se debe tener cuidado con la correcta colocación de los paréntesis asociativos).

```
> alpha - beta + delta - 32899;
```

$$\alpha - \beta + \delta - 32899$$

```
> sqrt(alpha - beta + delta) + 32777;
```

$$\sqrt{\alpha - \beta + \delta} + 32777$$

```
> (A + 1)^(A + 1);
```

$$(A+1)^{(A+1)}$$

```
> (A + 1)^((A + 1)^(A + 1));
```

$$(A+1)^{((A+1)^{(A+1)})}$$

```
> (a + hola_usuarios_de_Maple)/(a - hola_usuarios_de_Maple);
```

$$\frac{a + hola\_usuarios\_de\_Maple}{a - hola\_usuarios\_de\_Maple}$$

- Expresiones que contienen signos de relación.

```
> ecuacion = 3*a + b - gamma;
```

$$ecuacion = 3\,a + b - \gamma$$

```
> ddd + 1 > 33331;
```

$$0 < ddd - 33330$$

Todas las instrucciones dadas a, y producidas por Maple, están formadas por expresiones que caen dentro de alguno de los tipos descritos anteriormente.

Expresiones NO válidas

A continuación veremos algunos ejemplos de expresiones no válidas en este sistema.

- Expresiones sintácticamente mal escritas, en las que se incluyen comas, comillas, paréntesis sin cerrar, números y operaciones aritméticas incompletas. Por ejemplo:

```
> 2,; # La coma no debería aparecer
Error, `;` unexpected

> )ae4n; # El parentesis está mal colocado
Error, `)` unexpected

> g%; # El símbolo % está mal colocado
Error, missing operator or `;`
```

```
> ae(11)bcv; # No se pueden colocar parentesis de esta forma
Error, missing operator or ';'
```

- Expresiones que usan incorrectamente símbolos de operaciones y relaciones.

```
> a-;
Error, ';' unexpected

> a^a^a^a; # Se deben colocar parentesis asociativos
Error, '^' unexpected

> a = b = c; # No se pueden expresar relaciones de esta forma
Error, '=' unexpected
```

- Expresiones formadas por símbolos o palabras reservadas. Por ejemplo, a los nombres de funciones predefinidas y las constantes simbólicas no se les puede asignar un valor diferente al que representan, tales como @, #, %, I, Pi, sqrt, etc. Por ejemplo:

```
> Pi := 23;
Error, attempting to assign to 'Pi' which is protected

> sqrt := 23;
Error, attempting to assign to 'sqrt' which is protected
```

Nota: el operador **":="** es utilizado para asignar a una expresión un valor determinado. Este operador será tratado con detalle más adelante.

Expresiones que contienen espacios

Normalmente las expresiones que contienen espacios en blanco no pueden ser usadas como nombres, a menos que se encuentren delimitadas por acentos agudos. Compárese el resultado de las siguientes expresiones:

```
> la solucion general es;
Error, missing operator or ';'

> 'la solucion general es';
```

la solucion general es

En el primer ejemplo, Maple genera un mensaje de error pues tenemos una expresión que contiene espacios en blanco y por lo tanto cada palabra se considera como una expresión independiente. En cambio en la segunda expresión, al colocar los acentos agudos, Maple considera a todas las palabras con sus espacios en blanco como una sola expresión. A este tipo de datos delimitados por acentos agudos se les conoce con el nombre de ***"cadenas"*** (***"strings"***), como se verá más adelante.

Paréntesis asociativos en las expresiones

En todas las expresiones es importante colocar los paréntesis necesarios para asociar de manera adecuada. Siempre se deben usar "( )". Si no existen paréntesis, Maple no hace la asociacion de manera automática. Por ejemplo:

```
> a^3/2 <> a^(3/2);
```

$$\frac{a^3}{2} \neq a^{(3/2)}$$

```
> a/3 + b <> a/(3 + b);
```

$$\frac{a}{3} + b \neq \frac{a}{3+b}$$

```
> A + b^3 <> (A + b)^3;
```

$$A + b^3 \neq (A+b)^3$$

2.4.4 Cadenas

Las cadenas están formadas por expresiones que no tienen ningún significado especial para Maple. Por ejemplo cadenas de texto (siempre y cuando no sean comentarios). Son conjuntos de caracteres que generalmente se encuentran encerrados por acentos agudos.

```
> `esta es una cadena`; `Maple es un programa`;
```

esta es una cadena

Maple es un programa

2.4.5 Nombres de variables

Maple permite asignarle un nombre a cualquier elemento. Por ejemplo a una constante numérica, a una gráfica o al resultado de un cálculo. Este nombre debe estar formado por una expresión válida seguida de los caracteres ":=". Por ejemplo:

```
> nombre1 := 34;
```

$$nombre1 := 34$$

```
> `operación 1` := 23*124 - 75^2;
```

$$operacin\ 1 := -2773$$

```
> grafica_1 := plot(x^2, x=-2..2):
> deriv := diff(sin(x^3)*cos(x^8), x);
```

$$deriv := 3\cos(x^3)\,x^2\cos(x^8) - 8\sin(x^3)\sin(x^8)\,x^7$$

Nota: diff calcula la derivada de una función. Véase su página de ayuda ejecutando **?diff**.

Este tipo de asignación de nombres se le conoce como definición de variables. Los nombres se consideran un tipo de dato básico, sin embargo las variables serán tratadas más adelante.

Además de los tipos de datos descritos anteriormente, existen algunas estructuras manejadas por Maple. Aunque éstas serán tratadas con detalle posteriormente, vale la pena mencionarlas ahora dado que también pueden ser consideradas como tipos básicos y además forman parte de la sintaxis de una gran cantidad de funciones.

2.4.6 Secuencias

Una secuencia está formada por varias expresiones separadas por comas. Por ejemplo:

```
> a, 3, lista, 0, 1;
```

$$a,\ 3,\ lista,\ 0,\ 1$$

Esta secuencia está formada por los elementos **"a"**, **"3"**, **"lista"**, **"0"** y **"1"**.

Estas secuencias pueden ser usadas, por ejemplo, para evaluar varias expresiones y obtener los resultados en forma de secuencia:

```
> evalf(sqrt(Pi)), 23*45, 4.3/8;
```

1.772453851, 1035, 0.5375000000

2.4.7 Listas

Este tipo de estructura está formado por secuencias de datos encerrados entre los símbolos "[ ]". Cualquier expresión de cualquier tipo puede formar parte de una lista. Por ejemplo:

[1, 2, 3, 4, 5, 6]

[a, b, c, d, e, f, g]

['Maple es un sistema para hacer cálculos cientificos', 'hola a todos', Pi, sqrt(34)]

Una lista puede estar formada incluso por otras listas:

[[a, b, c], [1, Pi, exp(23)], ['cadena 1', 123, 'secuencia 3']]

2.4.8 Conjuntos

Están formados por secuencias de expresiones delimitadas por los símbolos "{ }". Al igual que las listas, los conjuntos pueden estar formados por datos de cualquier tipo. Por ejemplo:

{1, 5, 9, exp(34), Pi^2}

{abcd, dfer, 'Maple despliega gráficas', 98698769}

y, de la misma forma que las listas, los conjuntos pueden contener conjuntos:

{{a, g, t, 54325, I}, fkjdshlf, variable1, {grafica1, grafica2, grafica3}}

Todos estos tipos de datos pueden ser usados en combinación para crear, por ejemplo, listas de conjuntos, conjuntos de números, conjuntos de listas, listas formadas por conjuntos, números y cadenas, etc. Esto se tratará con más detalle posteriormente.

A continuación se describen otras estructuras manejadas por Maple.

2.4.9 Tablas

Las tablas están formadas por un grupo de datos, cada uno de los cuales tiene asignado un índice para referenciarlo. Estas estructuras son creadas con la función **table** (véase su página de ayuda ejecutando **?table**). Por ejemplo:

```
> T := table([el1 = 3, el2 = 8, el3 = 6, palabra = hola, num = 987542]);
```

$$T := \text{table}([el2 = 8,\ el3 = 6,\ num = 987542,\ palabra = hola,\ el1 = 3])$$

Las expresiones colocadas a la izquierda de cada relación son los índices. Para acceder a un elemento debemos utilizar su índice, por ejemplo:

```
> T[num];
```

$$987542$$

```
> T[el3];
```

$$6$$

Este tipo de datos también serán tratados posteriormente.

2.4.10 Arreglos

Los arreglos, al igual que las tablas, están formados por datos indexados, pero sus índices solo pueden ser números, no cadenas. Tales estructuras son creadas mediante la instrucción **array** (consúltese su ***página de ayuda*** ejecutando **?array**). Veamos un ejemplo.

```
> A := array(3..5, [4, 8, 25]);
```

$$A := \text{array}(3..5, [$$
$$(3) = 4$$
$$(4) = 8$$
$$(5) = 25$$
$$])$$

Para hacer referencia a cada elemento utilizamos su índice:

```
> A[3];
```

$$4$$

```
> A[5];
```

$$25$$

Estas estructuras también pueden ser manejadas como vectores renglón, como veremos más adelante.

2.4.11 Subíndices

Es posible usar subíndices en expresiones de Maple, éstos se denotan con los símbolos "[ ]". Cualquier nombre (o secuencia de nombres) sintácticamente bien definido puede ser usado como subíndice.

```
> S[1];
```

$$S_1$$

```
> Solucion[`caso a=0`];
```

$$Solucion_{caso\ a=0}$$

```
> S[`solucion general`];
```

$$S_{solucion\ general}$$

```
> S[a, b, c];
```

$$S_{a,b,c}$$

Es posible colocar subíndices a los subíndices en múltiples niveles. Por ejemplo:

```
> S[a][b][c][d]['caso x=0'];
```

$$S_{abcd_{caso\ x=0}}$$

Como puede apreciarse en la instrucción anterior, se colocan los subíndices correspondientes, encerrados entre corchetes, secuencialmente. De esta forma se pueden usar expresiones con subíndices tan complejas como sea necesario.

2.5 Operadores Aritméticos

Los operadores aritméticos soportados por Maple son los siguientes:

- Suma: **a + b**.
- Resta y números negativos: **a - b, -d**.
- Producto: **a*b**.
- Producto no conmutativo: **a&*b** (utilizado en operaciones de matrices y vectores).
- Potencia: **a^b** (también: **a**b**).
- División: **a/b** (también: **a*b^(-1)**).

  Maple 8 permite usar el operador "/" en la forma: **'/'(a,b)** o **'/'(c)**, para expresar las operaciones **a/b** y **1/c**, respectivamente. Nótese que el operador debe estar delimitado por apóstrofes.
- Factorial: **!**.

  También se puede usar la función **factorial**, véase su página de ayuda ejecutando: **?factorial**.

Con estos operadores es posible usar Maple para realizar cálculos aritméticos sencillos:

```
> 4 + 3; 3*24567; 264/4;
```

$$7$$
$$73701$$
$$66$$

Una característica importante (y muy poderosa) de Maple es que no solo realiza cálculos de tipo numérico, como las operaciones anteriores; también nos permite realizar operaciones de tipo simbólico. Por ejemplo, podemos ejecutar las mismas instrucciones anteriores pero usando datos simbólicos.

```
> a + b; 3*a; a/b;
```

$$a + b$$
$$3\,a$$
$$\frac{a}{b}$$

Nótese que estas operaciones se llevan a cabo sin importar que uno o varios de los operandos involucrados no tengan un valor numérico asignado.

La precedencia de estos operadores es la usual, sin embargo, siempre que existan dudas o se crea necesario, se pueden usar los paréntesis "( )" para asociar las operaciones. Un caso en el que en ocasiones es necesario usar paréntesis para asociar las operaciones es la exponenciación:

```
> a^a^a^a; # Hacen falta los parentesis
Error, `^` unexpected
```

```
> a^(a^(a^(a)));
```

$$a^{(a^{(a^a)})}$$

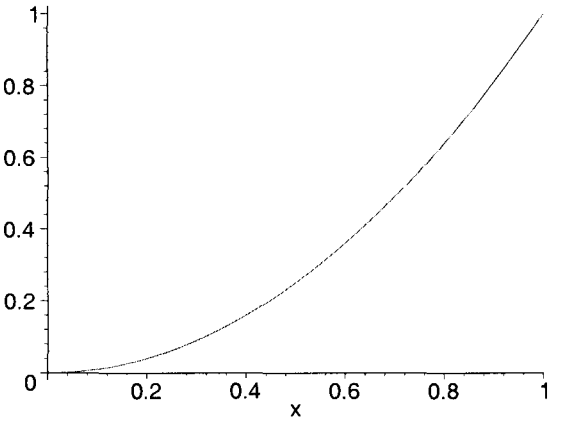
2.6 Paréntesis de funciones, operadores e instrucciones

Los símbolos "( )" indican, ya sea de que variables depende una función, o bien, cuales son los argumentos que recibe. Todos los datos contenidos en estos paréntesis deben ir separados por comas:

```
> f(a, b, c);
```

$$\mathrm{f}(a, b, c)$$

```
> plot(x^2, x=0..1);
```



2.7 Evaluación de expresiones

2.7.1 La función restart

Durante el desarrollo de una hoja de trabajo, al realizar cálculos, definir variables o ejecutar funciones, los resultados generados permanecen en la memoria utilizada por Maple. En las instrucciones ejecutadas posteriormente, estos resultados guardados pueden llegar a causar ciertos problemas al interferir en los nuevos cálculos. Por ejemplo, considérense las siguientes instrucciones:

```
> b := sqrt(23) - 12*Pi;
```

$$b := \sqrt{23} - 12\,\pi$$

```
> evalf(b);
```

$$-32.90328033$$

¿Qué sucede si a continuación deseamos declarar una ecuación en términos de b?

```
> 32^2 + 4*b - b^2 = 0;
```

$$1024 + 4\sqrt{23} - 48\,\pi - (\sqrt{23} - 12\,\pi)^2 = 0$$

Nótese que, al tener ***b*** un valor asignado previamente, Maple considera la instrucción anterior como una ecuación en la que se debe sustituir el valor asignado a esta **"variable"**. De ahí el resultado.

Existe una manera de ***"desasignar"*** una variable, de tal forma que se elimine todo valor previamente asignado (esto se tratará posteriormente en la sección de variables). Otra forma de hacerlo es indicando a Maple que debe eliminar todo resultado anterior de la memoria, antes de realizar el siguiente cálculo. Esto es posible a través de la función **restart**. Por ejemplo, si ejecutamos esta instrucción antes de definir la ecuación anterior obtenemos lo siguiente:

```
> restart;
> 32^2 + 4*b - b^2 = 0;
```

$$1024 + 4\,b - b^2 = 0$$

Nótese que ahora Maple considera a esta expresión como una ecuación que depende de la variable ***b***, la cual, por cierto, es una variable símbolica, sin ningún valor asignado. Esto se debe a que **restart** eliminó todo valor almacenado en la memoria, en particular toda referencia anterior a una variable con este nombre. La instrucción **restart** también puede ser invocada por medio del botón  , de la barra de herramientas.

2.7.2 Evaluación por sustitución

Existen diferentes formas en las que podemos solicitar a Maple que lleve a cabo la evaluación de una expresión, dependiendo del tipo de ésta y del tipo de resultado que deseamos. Una de estas formas de evaluación consiste en solicitar a Maple que sustituya las variables de la expresión por valores particulares; al hacer esto automáticamente se evalúa la expresión como si ésta fuera de tipo aritmético. Para ilustrar esto consideremos el siguiente caso:

La siguiente expresión es conocida como ***"ecuación general de una línea recta"***:

```
> y := m*x + b;
```

$$y := m\,x + b$$

Las variables simbólicas, **m** y **b**, pueden tomar cualquier valor. La cuestión es como evaluamos la expresión ***"y"*** en valores particulares de **m**, **b**, **x** (por ejemplo **m = 2**, **b = 3**, **x = 5**). La manera de hacer esto es sustituyendo, mediante la instrucción **subs**:

subs(m = 2, b = 3, x = 5, y);

En general, esta forma de sustituir es válida para cualquier expresión. Si tenemos una expresión **M** cualquiera y **v1, v2, v3, ..., vn** las variables de las cuales depende; las sustituciones de valores particulares de cada una de ellas pueden hacerse de la siguiente forma:

subs(v1 = valor1, v2 = valor2, . . ., vn = valorn, M);

Es posible sustituir tantas variables como tenga la expresión. Por ejemplo:

```
> y; subs(m = 2, y);
```

$$m\,x + b$$
$$2\,x + b$$

```
> y; subs(m = 2, b = 3, y);
```

$$m\,x + b$$
$$2\,x + 3$$

```
> y; subs(m = 2, b = 3, x = 0.3, y);
```

$$m\,x + b$$
$$3.6$$

Sustitución de variables por otras variables

No solo es posible hacer sustitución de variables simbólicas por números, también se pueden sustituir variables simbólicas por otras variables simbólicas.

Consideremos la siguiente expresión:

```
> M:=(1 + a*x - b*x^2)/((1 + alpha)*(1 + beta));
```

$$M := \frac{1 + a\,x - b\,x^2}{(1+\alpha)\,(1+\beta)}$$

Utilizando la función **subs**, podemos sustituir, por ejemplo **alpha = 1 + x**.

```
> subs(alpha = 1 + x, M);
```

$$\frac{1 + a\,x - b\,x^2}{(2+x)\,(1+\beta)}$$

o bien: **beta = 1 + a + x**.

```
> subs(beta = 1 + a + x, M);
```

$$\frac{1 + a\,x - b\,x^2}{(1+\alpha)\,(2+a+x)}$$

También es posible, en una misma instrucción, sustituir todas las variables de las cuales depende una expresión algebraica. De esta forma, podemos hacer que dicha expresión dependa de cualquier variable. Por ejemplo, podemos hacer que ***M*** dependa solo de ***x***:

```
> subs(a = x, b = x, alpha = x, beta = x, M);
```

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

2.7.3 Evaluación simbólica

A veces es necesario hacer que Maple evalúe una expresión, además de hacer una sustitución. Por ejemplo, consideremos la expresión ***"y"*** definida previamente:

```
> y := m*x + b;
```

$$y := m\,x + b$$

hagamos las siguientes sustituciones:

```
> subs(b = cos(u), y);
```

$$m\,x + \cos(u)$$

```
> subs(u = 0, %);
```

$$m\,x + \cos(0)$$

Pero sabemos que **cos(0) = 1**, sin embargo Maple no realiza en este caso la evaluación de la expresión tomando en cuenta este valor. Para que Maple evalúe está expresión para $u = 0$ es necesario usar la instrucción **eval**.

```
> eval(%);
```

$$m\,x + 1$$

En muchos casos, es suficiente usar **subs**, pero en general, es conveniente usar **subs** y **eval** juntos, de la siguiente forma:

```
> subs(b = cos(u), y);
```

$$m\,x + \cos(u)$$

```
> subs(u = 0, %);
```

$$m\,x + \cos(0)$$

```
> eval(%);
```

$$m\,x + 1$$

También es posible combinar las instrucciones **eval** y **subs** en una sola instrucción:

```
> eval(subs(b = cos(0), y));
```

$$m\,x + 1$$

Este tipo de combinación de funciones (**eval** y **subs**, por ejemplo) son válidas en Maple, siempre y cuando haya compatibilidad en cuanto a los datos que manejan ambas (es obvio que no se le puede pedir a **eval** que realice alguna operación sobre una gráfica, por ejemplo).

La instrucción **eval** se puede utilizar para realizar evaluaciones explícitas en expresiones simbólicas en general, como en el ejemplo anterior.

2.7.4 Evaluación de expresiones pasivas o inertes

Existen varias funciones en Maple que tienen la capacidad de generar, a partir de instrucciones de Maple, expresiones en notación matemática. Este modo de operar se le conoce como forma ***"pasiva"*** o ***"inerte"*** de la función. Al ejecutar las instrucciones en este modo, en lugar de obtener un resultado, lo que se obtiene es una expresión en simbología matemática que representa la operación solicitada. Por ejemplo, una de las funciones que pueden trabajar en forma inerte es **limit**, que calcula el límite de una función alrededor de un punto dado. Su sintaxis es:

limit(función(x) o expresión(x), x=punto);

Por ejemplo, calculemos el límite de $\frac{\sin(h)}{h}$, alrededor de $h = 0$.

```
> limit(sin(h)/h, h = 0);
```

$$1$$

En este caso, estamos usando la función **limit** en su forma normal, lo cual produce como resultado la evaluación de dicha función y por lo tanto el valor del límite estimado. La forma inerte se obtiene al ejecutar una instrucción (que cuente con una forma inerte en Maple), pero escribiendo el nombre con la primera letra mayúscula. En este caso, la forma inerte de esta función es:

Limit(función(x) o expresión(x), x=punto);

Solicitemos nuevamente el mismo cálculo, pero en forma inerte:

```
> Limit(sin(x)/x, x = 0);
```

$$\lim_{x \to 0} \frac{\sin(x)}{x}$$

Como puede apreciarse, en este caso obtenemos una expresión en simbología matemática que representa la instrucción ejecutada. Este tipo de expresiones pueden ser bastante útiles como se verá más adelante.

Cuando obtenemos un resultado a partir de la forma inerte de una función, es posible hacer que Maple evalúe la expresión generada para obtener el valor de dicha expresión. Esto puede hacerse por medio de la función **value**. Así, podemos obtener el valor numérico del límite simbólico anterior con la instrucción:

```
> value(%);
```

$$1$$

En general, **value** nos permite evaluar expresiones que fueron obtenidas en forma ***"inerte"***.

2.7.5 Evaluación numérica

Este sistema evalúa de manera diferente las expresiones aritméticas, dependiendo del tipo de números que las componen. En general, se realizan evaluaciones de este tipo en dos formas diferentes.

Números enteros y racionales

Siempre que se introduce una expresión que involucra datos numéricos, Maple automáticamente trata de evaluarla; dicha evaluación, por cierto, generalmente simplifica la expresión. Sin embargo existen ciertas reglas que se siguen al hacer tales evaluaciones.

Si una expresión esta formada solo por números enteros y/o racionales, el resultado será dado con números enteros o racionales. Por ejemplo:

```
> 5!*46 - 23^3*8/5;
```

$$\frac{-69736}{5}$$

```
> (25*x + 1/4*x^2) + (2*x - 5*x^2);
```

$$27\,x - \frac{19}{4}\,x^2$$

```
> 23*Pi^2 - 18^3*4^2/(3^2 + 6^4) + 4!;
```

$$23\,\pi^2 - \frac{6888}{145}$$

Nótese que Maple realiza todas las evaluaciones posibles de tal forma que, en el resultado, no aparezcan números de punto flotante. Por ejemplo, en la última expresión no se evalúa π^2, ni la división $\frac{6888}{145}$. En general, ésta es la forma en la cual se realizan las evaluaciones con enteros y/o racionales. Este modo de evaluar es diferente al de una calculadora, en la cual los números se convierten automáticamente a punto flotante, pero existe el problema de que en cálculos que involucran números grandes, la conversión a punto flotante puede provocar errores de precisión considerables. En este sentido la forma de evaluar de Maple es más eficiente que la de los programas que trabajan siempre con punto flotante.

Numeros de punto flotante

Como ya se mencionó, Maple por omisión no realiza las operaciones numéricas en punto flotante. Sin embargo, existen varias formas en las cuales podemos obligar a que los resultados se presenten así. Una de ellas es (como vimos anteriormente) incluyendo como operando al menos un número de punto flotante. Por ejemplo:

```
> 4.2/3^4 - 54^2*4^3;
```

$$-186623.9481$$

Otra forma de obtener resultados de esta forma es usando la instrucción **evalf**, la cual recibe como argumento una expresión aritmética y la evalúa devolviendo un resultado de punto flotante. Su sintaxis es:

evalf(N, número de decimales);

También puede ser invocada en la forma:

evalf[número de decimales](N);

Esta función nos permite además obtener aproximaciones de números irracionales, tales como $\sqrt{2}$, Pi, e; así como realizar cálculos de precisión arbitraria. Por ejemplo:

```
> sqrt(2); # No se obtiene un número de punto flotante
```

$$\sqrt{2}$$

```
> evalf(%); # Esto nos da un número de punto flotante
```

$$1.414213562$$

Por omisión, al realizar operaciones de punto flotante, Maple solo despliega un máximo de diez decimales. **evalf** nos permite obtener una precisión mayor, solicitando una cantidad arbitraria de decimales. Por ejemplo, evaluemos la expresión anterior con 100 decimales.

```
> evalf(%%, 100);
```

$$1.41421356237309504880168872420969807856967187537694807317667973799073247846210703885038753432764157 3$$

Esta última instrucción es equivalente a: **evalf[100](%%);**

Veamos otro ejemplo.

```
> 1.1*sqrt(3) + .1;
```

$$1.1\sqrt{3} + 0.1$$

Este resultado es un número irracional (de hecho es un número exacto pero no nos da una idea clara de su valor). Podemos obtener una aproximación con **evalf**:

```
> evalf(%);
```

$$2.005255889$$

2.8 Operadores relacionales

Los símbolos relacionales que soporta Maple son los siguientes:

- Igual: =
- Menor que: <
- Mayor que: >
- Menor o igual: <=
- Mayor o igual: >=
- Desigual o diferente: <>

Con ellos podemos expresar igualdades (ecuaciones) o desigualdades (inecuaciones). Por ejemplo:

```
> a = b;
```

$$a = b$$

```
> 3 < 4; 3 <> 4;
```

$$3 < 4$$
$$3 \neq 4$$

```
> a <= 0; a >= 0;
```

$$a \leq 0$$
$$0 \leq a$$

Es posible escribir, en las regiones de entrada, expresiones con símbolos de relación (un solo símbolo de relación por cada instrucción).

```
> a + b^2 + f - (a + b)^2 = c*d*(1 - a*b);
```

$$a + b^2 + f - (a+b)^2 = c\,d\,(1 - a\,b)$$

```
> (3*b - 2*s)/(a - b)*f >= 0;
```

$$0 \leq \frac{(3\,b - 2\,s)\,f}{a - b}$$

2.9 Operadores lógicos

Los operadores lógicos soportados por Maple son:

- Conjunción: **and**
- Disyunción: **or** y **xor** (**or** exclusivo)
- Negación: **not**
- Implicación: **implies**

Posteriormente veremos algunos ejemplos del uso de estos operadores.

2.10 Manipulación de los miembros de una relación

En la sección anterior vimos como expresar relaciones. Una relación está formada por dos expresiones unidas por un operador relacional. En ocasiones es necesario poder referenciar cada uno de sus componentes. Las instrucciones **lhs** y **rhs** permiten extraer los miembros izquierdo y derecho, respectivamente, de instrucciones en las cuales aparece un signo de relación.

```
> a + b = c + 1/d;
```

$$a + b = c + \frac{1}{d}$$

```
> lhs(%);
```

$$a + b$$

```
> rhs(%%);
```

$$c + \frac{1}{d}$$

La instrucción **lhs** nos permite extraer el miembro izquierdo de la relación $a + b = c + \frac{1}{d}$; es decir, $a + b$. Mientras que el operador **rhs** nos permite extraer el componente derecho, $c + \frac{1}{d}$.

2.11 Utilización de letras griegas

Maple nos permite utilizar letras del alfabeto griego en las instrucciones de entrada. Para poder usar una de estas letras basta con escribir su nombre completo. Por ejemplo, para obtener la letra "α", debemos escribir la palabra ***"alpha"***. Estas palabras (nombres de letras griegas), automáticamente se convierten en el símbolo correspondiente. Así, podemos incluir expresiones como la siguiente:

```
> alpha + beta^2 + epsilon - (a + b)^2 = c*d*(1 - alpha*beta);
```

$$\alpha + \beta^2 + \varepsilon - (a+b)^2 = c\,d\,(1 - \alpha\,\beta)$$

```
> (3*beta - 2*gamma)/(a - b)*Phi >= 0;
```

$$0 \leq \frac{(3\,\beta - 2\,\gamma)\,\Phi}{a - b}$$

Para poder utilizar letras mayúsculas del alfabeto griego solo debemos colocar su nombre con la primera letra en mayúscula. Por ejemplo, para obtener la letra "Γ", debemos escribir la palabra **"Gamma"**, colocando la letra **"G"** mayúscula. Maple interpreta los nombres de letras griegas siempre y cuando éstas sean introducidas en una región de entrada o en regiones de texto a través de la opción **Standard Math** del menú **Insert**.

Recuérdese que esta versión de Maple nos proporciona varias paletas a través de las cuales podemos introducir expresiones, una de ellas es la **"paleta de símbolos"**, precisamente por medio de ésta podemos incluir letras del alfabeto griego. Para visualizar la paleta debemos solicitar la opción **Palettes - Symbol Palette**, del menú **View**.

Capítulo 3

El sistema de ayuda de Maple

Maple proporciona al usuario uno de los sistemas de ayuda más completos dentro de los programas de cálculo científico, el cual está formado por un conjunto de documentos. Una parte de estos documentos contiene información acerca de cada una de las funciones soportadas por Maple, mientras que otros contienen información acerca de la versión actual de este sistema y sus características. También proporcionan un recorrido a los usuarios nuevos de Maple, así como una introducción a esta aplicación, información útil acerca de la manera de usar el mismo sistema de ayuda, ayuda contextual en los diferentes botones de la barra de herramientas y los menús contextuales, además de un sistema de búsqueda por temas o frases. Existen varias formas de utilizar esta ayuda, dependiendo de las necesidades del usuario. En el caso de las hojas de ayuda de las funciones de Maple, además de la información acerca de cada función, cada hoja proporciona un conjunto de ejemplos probados que pueden ser copiados a la hoja de trabajo para su ejecución. De esta forma, un usuario con poca o ninguna experiencia en el uso de alguna función, en la mayoría de los casos solo tiene que modificar alguno de los ejemplos contenidos en esta ayuda para poder sacarle provecho.

Como vimos en secciones anteriores, la sintaxis de esta aplicación es sencilla; además, gracias a su excelente sistema de ayuda, no es necesario memorizar una gran cantidad de funciones ni todos los detalles referentes a éstas. Maple proporciona, para cada tema y cada función, información muy completa acerca de los parámetros, opciones, sintaxis, relación con otras funciones y ejemplos probados.

3.1 Estructura de una hoja de ayuda

Cada una de las hojas que forman parte del sistema interactivo de ayuda de Maple, guardan una cierta estructura que permite al usuario cunsultar todo el documento o solo ciertas partes de su interés. Así, pueden reconocerse varias secciones dentro de cada una de estas hojas.

3.1.1 Secciones que componen una hoja de ayuda

Las hojas de ayuda de Maple están estructuradas por secciones, a continuación describimos cada una de ellas con su respectivo nombre:

- **Nombre de la función**. Esta sección presenta al usuario el nombre de la función a la cual corresponde la hoja de ayuda, además de una breve descripción de su utilidad.
- ***Calling sequence***. En esta sección se presentan de manera general las diferentes formas en que la función puede ser invocada.

- ***Parameters***. Proporciona una descripción del tipo de dato para cada uno de los parámetros aceptados por la función; también nos indica cuales de los argumentos son opcionales.

- ***Description***. Esta sección nos presenta una descripción detallada de la función. Además de mencionar la utilidad de ésta, también se incluyen descripciones de los tipos de datos que pueden ser dados como argumentos, así como las principales opciones que soporta, el número de argumentos que puede recibir, las diferentes formas en que puede invocarse y un conjunto de vínculos que apuntan a funciones relacionadas o bien a opciones de la misma función.

- ***Examples***. Sin duda esta es una de las secciones más útiles para los usuarios, sobre todo a quienes tienen poca o ninguna experiencia en el manejo de Maple, ya que proporciona un conjunto de ejemplos listos para ser ejecutados; solo deben ser copiados y pegados en una hoja de trabajo. En la mayoría de los casos, cuando un usuario se enfrenta a una función desconocida, es suficiente con recurrir a los ejemplos para poder sacarle provecho. Generalmente, es suficiente con copiar y pegar uno de estos ejemplos y hacer algunas modificaciones (basandose también en el sistema de ayuda), para poder resolver problemas sencillos e incluso complicados.

- ***See Also***. Esta sección presenta un conjunto de vínculos a las hojas de ayuda de aquellas funciones que están relacionadas con la que se está consultando.

3.2 Obtención de la ayuda

Al acceder a una página de ayuda, es posible consultar ya sea todo el documento o solo alguna de las secciones que lo componen, dependiendo de las necesidades del usuario; para tal fin existen varias maneras de obtener dicha ayuda, éstas se describen a continuación.

3.2.1 En la línea de comandos

Por medio del operador **"?"** podemos consultar una hoja de ayuda o secciones especificas de ésta, desde la línea de comandos. Este operador puede ser usado de varias formas, las cuales revisaremos a continuación.

Forma "?tema"

La expresión **"?tema"** (donde **tema** puede ser el nombre de una función), nos permite tener acceso a la hoja de ayuda del tema correspondiente. Al ejecutar este operador, Maple nos presenta la hoja de ayuda solicitada con todas las secciones abiertas. Por ejemplo, para obtener la hoja de ayuda de la función **evalf** podemos utilizar la siguiente instrucción:

```
> ?evalf
```

Cabe mencionar que la forma **?tema** también puede ser invocada a través de la función **help**, pasando como argumento el tema acerca del cual requerimos la ayuda. Por ejemplo, la instrucción anterior es equivalente a la siguiente:

```
> help(evalf);
```

Es recomendable utilizar preferentemente **"?"** en lugar de la función **help**, dada su sencilla sintaxis.

Forma "?tema[subtema]"

La forma **"?tema[subtema]"** nos permite acceder a la ayuda correspondiente al subtema indicado. Por ejemplo, para obtener la página de ayuda de las opciones soportadas por la función **plot** podemos utilizar la siguiente instrucción:

```
> ?plot[options]
```

Esta forma también puede ser invocada como **"?tema, subtema"** o bien, a través de la función **help**, como **help("tema, subtema")**. Así, la instrucción anterior es equivalente a:

```
> ?plot, options
```

y también a:

```
> help("plot, options");
```

Nótese que en este último caso el argumento de **help** debe ser encerrado entre comillas. Esta última instrucción también es equivalente a: **help("tema[subtema]")**

Forma "??tema"

Esta forma nos permite acceder a la ayuda del tema correspondiente, pero teniendo visible solamente la sección del nombre de la función y el tipo de los argumentos que recibe (véase la sección 3.1.1). Por ejemplo, ejecutemos la siguiente instrucción:

```
> ??value
```

En este caso la hoja de ayuda que nos muestra Maple solo presenta la primera sección abierta, el resto permanecen cerradas.

Esta forma de invocar la ayuda también puede ser usada de la siguiente forma:

??tema, subtema

??tema[subtema]

para obtener la ayuda correspondiente de un subtema.

Forma "???tema"

Esta forma nos permite visualizar la sección de los ejemplos de una página de ayuda. Por ejemplo, ejecútese la siguiente instrucción:

```
> ???plot
```

Nótese que en este caso, Maple nos despliega la hoja de ayuda y solo aparece abierta la sección que muestra los ejemplos acerca de la función **plot**.

Esta forma también puede ser usada de la siguiente manera:

???tema, subtema

???tema[subtema]

3.2.2 A través de los menús

Las opciones descritas anteriormente, nos permiten acceder a las páginas de ayuda de funciones o temas conocidos. Pero, ¿Qué sucede si deseamos ayuda acerca de un tema pero no sabemos como solicitarselo a Maple?

Para obtener ayuda en línea, es necesario conocer el nombre del tema o función; sin embargo, Maple nos ofrece algunas opciones a través de las cuales podemos hacer búsquedas, por ejemplo de palabras clave o, con ayuda de un índice, navegar entre las hojas que componen el sistema de ayuda. Estas opciones son descritas a continuación y todas ellas pueden ser seleccionadas del menú **Help**.

Introduction

Esta opción nos muestra un documento de Maple con información introductoria acerca de este sistema. Puede ser consultada por usuarios con poca o ninguna experiencia, ya que presenta al usuario información acerca de los diferentes temas que pueden ser abordados con esta aplicación. La información de este documento es presentada en forma temática con hipervínculos que pueden llevar al usuario a otros documentos del programa o a hojas del sistema de ayuda, de tal forma que se puede realizar un ***"recorrido"*** completo por Maple, consultar información acerca de las características nuevas presentes en esta versión, aprender a hacer uso del sistema de ayuda, de los distintos elementos que componen la interfaz gráfica y las opciones disponibles en los diferentes menús. Este documento introductorio puede ser consultado mediante la opción **Introduction** del menú **Help**, haciendo clic aquí , o bien ejecutando la instrucción **?introduction**.

Help on Context

Para ilustrar esta opción, considérese la siguiente instrucción (sin evaluarla):

```
> evalf(Pi^4, 200);
```

En la región de entrada anterior colóquese el cursor de texto en la palabra **"evalf"** (al inicio o dentro de la palabra, pero no al final) o bien seleccione la palabra. Si a continuación, en el menú **Help**, solicitamos la opción **Help on Context**, Maple automáticamente nos despliega la hoja de ayuda de esta función.

En general, si en una región de entrada colocamos el cursor de texto dentro de una palabra (o seleccionamos ésta), al solicitar la opción **Help on Context**, Maple automáticamente tratará de desplegarnos ayuda acerca de dicha palabra (en caso de que exista tal información). Si solicitamos ayuda acerca de una palabra que el sistema no reconoce, obtenemos un mensaje de error.

New users tour

Está opción nos da acceso a una hoja de trabajo, la cual nos permite hacer un recorrido por las características de Maple. Es recomendable sobre todo para usuarios nuevos de este sistema. Este documento puede consultarse con la opción **New users tour**, del menú **Help**, aunque también aparece en forma de vínculo dentro del documento de introducción a Maple.

Using help

Esta opción nos da acceso a un documento que nos muestra, de manera clara y extensa, el uso del sistema de ayuda de Maple. Para consultar este documento haga ***clic aquí*** .

Glossary

A través de esta opción podemos tener acceso a un glosario con los términos más importantes usados por Maple.

Topic Search

La opción **Topic search** nos muestra una ventana en la cual podemos solicitar búsquedas por tema o por palabra clave. Para ejemplificar el uso de esta opción, realizaremos una búsqueda acerca de **limit** (estas palabras deben ser dadas en inglés).

Primero, en el menú **Help**, solicitamos la opción **Topic Search**. A continuación, escribimos la palabra **limit** en la sección **Topic**; automáticamente, en la sección **Matching Topics** Maple nos mostrará un conjunto de vínculos relacionados con esta palabra (vease la Figura 3.1).

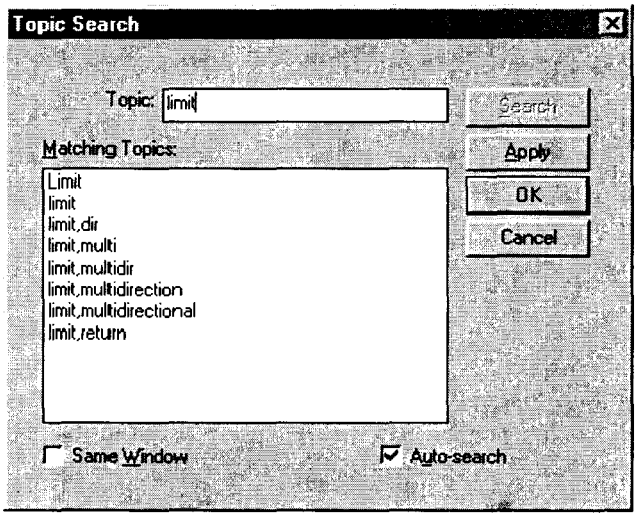


Figura 3.1: Ventana para busquedas por tema

Cada línea dentro de la sección **Matching topics**, representa una hoja de ayuda que podemos consultar acerca de la palabra ***"limit"*** (nótese que algunas de ellas están dadas en la forma ***"tema,subtema"***), para visualizarlas basta con seleccionarlas con el ratón y oprimir el botón **Apply** u **OK**.

Full text search

Esta opción es análoga a **Topic search**, salvo por que nos permite realizar búsquedas de texto libres dentro de las páginas de ayuda. Para relizar esta búsqueda tecleamos, en el campo **Words(s)** de la ventana desplegada, la palabra o palabras que deseamos buscar; a continuación oprimimos el botón **Search** y, en el campo **Matching topics**, se mostrarán todas las páginas de ayuda en las cuales se encontró el texto tecleado. Por ejemplo, la Figura 3.2 nos muestra el resultado al solicitar una búsqueda de la palabra ***"graphic"*** (recuérdese que ésta debe escribirse en inglés).

Para consultar una de las páginas de ayuda listadas en la ventana, simplemente la seleccionamos con el ratón y oprimimos el botón **OK** o **Apply**.

History

Esta opción nos presenta una lista de las últimas páginas del sistema de ayuda que hemos consultado, de tal manera que facilita al usuario volver a consultarlas sin necesidad de buscarlas nuevamente. Para revisar una de las páginas listadas es suficiente con seleccionarla con el ratón y oprimir a continuación el botón **OK** o **Apply**.

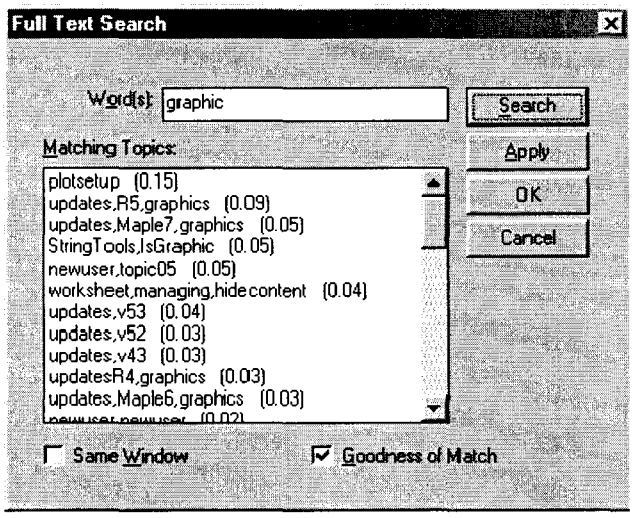


Figura 3.2: Ventana para busquedas de texto

Save to database

Si bien el sistema de ayuda de Maple no puede ser modificado por el usuario, esta aplicación proporciona soporte para una base de datos con documentos creados por el usuario, la cual puede ser modificada y se encuentra accesible de la misma forma que el sistema de ayuda propio de Maple. La opción ***Save to database*** nos permite, una vez que hemos creada una hoja de trabajo, agregarla a la base de datos mencionada, de tal manera que la podamos consultar como cualquier otro documento de la ayuda. Al agregarla a esta base de datos, debemos indicar cual es el tema acerca del cual trata este documento, esto permitirá a Maple el poder referenciarlo durante las búsquedas. Esta opción convierte la hoja de trabajo indicada a un archivo de ayuda de Maple y lo almacena en la base de datos modificable.

Remove topic

Esta opción nos permite eliminar un tema (documento) de la base de datos creado con la opción descrita anteriormente.

Balloon help

Cuando se activa esta opción, cada vez que posicionamos el apuntador del ratón sobre uno de los botones de la barra de herramientas o de la barra contextual, Maple despliega un pequeño rectángulo donde nos muestra información descriptiva de dicho botón.

Por ejemplo, si colocamos el apuntador sobre el botón  , Maple desplegará el cuadro de ayuda que se muestra en la Figura 3.3.

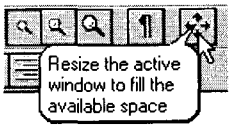


Figura 3.3: Ayuda contextual

Maple on the Web

Esta opción nos proporciona acceso directo a varios sitios de Web relacionados con Maple. Entre ellos se encuentran disponibles el **Maple Aplication Center**, **Maple Powertools** y el **Student Center**; entre otros.

Para hacer uso de estos accesos a los sitios de Web es necesario establecer la ruta de un navegador que pueda ser invocado para tal fin; esto puede hacerse por medio de la opción **Preferences** del menú **File**.

Register Maple 8

Esta opción nos permite acceder directamente a la página de registro de **Waterloo Maple Inc.**, para poder registrar nuestra copia de Maple. Antes de consultar páginas de WWW desde Maple, se debe indicar previamente el navegador a usar. Si aún no se indica, esta opción desplegará una ventana solicitando la ruta de un navegador instalado en el sistema.

About Maple 8

Esta opción muestra una ventana con información acerca de la versión de Maple que usamos, tal como la fecha de instalación, la versión, el tipo de instalación y los derechos de autor bajo los cuales está protegido este programa.

3.2.3 Navegando por el sistema de ayuda

Cada vez que consultamos una página de ayuda, en la parte superior de ésta aparece el **"Navegador del sistema de ayuda"** de Maple. Este Navegador está formado por un área de cinco columnas, con la cual se puede explorar todo el sistema de ayuda (vease la Figura 3.4)

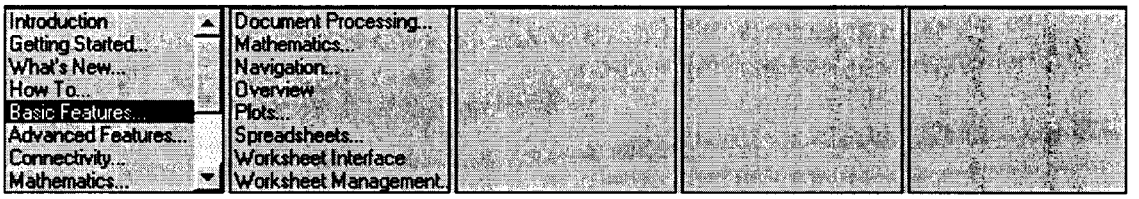


Figura 3.4: Navegador del sistema de ayuda de Maple

Para usar este navegador se deben seguir las siguientes instrucciones:

- Seleccionamos uno de los temas principales en la primera columna a la izquierda. Si este tema contiene subtemas, estos se mostrarán en la siguiente columna a la derecha, de lo contrario se desplegará la página de ayuda correspondiente.
- Los temas que contienen subtemas se pueden reconocer por que terminan con puntos suspensivos **"..."**. Estos subtemas aparecerán en la columna de la derecha, de donde pueden a su vez ser seleccionados para obtener su página de ayuda o sus correspondientes subtemas.
- Si se desea consultar información acerca de como realizar una operación específica sobre una hoja de trabajo, es recomendable consultar primero el tema **"How To"**, en la primera columna del navegador.
- Los temas de la primera columna nos pueden dar acceso también a la Introducción a Maple, al recorrido por Maple (**New Users Tour**), y a diversas hojas de trabajo de ejemplo; además de las hojas de ayuda de las funciones y operadores.

Capítulo 4

Uso de variables

Maple nos permite asignar a cualquier tipo de dato, estructura o resultado, un nombre formado por una palabra sintácticamente válida. Este tipo de expresiones se conocen con el nombre de ***"variables"***. Las variables son útiles ya que nos permiten ***"almacenar"*** datos para su uso posterior. Por ejemplo, considérese que para hacer referencia a resultados anteriores por medio de los operadores "%", "%%" y "%%%", no podemos referenciar más allá del antepenúltimo valor; en cambio, al utilizar variables, podemos almacenar un dato a lo largo de toda una sesión y utilizarlo, modificarlo o eliminarlo en cualquier momento, sin importar cuando fue generado.

4.1 Variables

Existen varios detalles que deben tenerse en cuenta al utilizar variables; respecto a la definición, nombres que se pueden usar, forma de asignar datos y de ***"desasignar"*** una variable previamente creada.

4.1.1 Definición de una variable

Para poder utilizar una expresión como variable y asignarle un dato o alguna otra expresión, se utiliza el operador **":="**, de la siguiente forma:

expresión1 := dato o expresión2;

donde **"expresión1"** es el nombre que le daremos a la variable. Por ejemplo, asignemos a la expresión **"a"** el valor **"34"**.

```
> a := 34;
```

$$a := 34$$

Una vez que se crea una variable, el valor asignado a ésta permanecerá inalterado hasta que se le asigne uno nuevo. Además, cada vez que hacemos referencia a una variable, Maple automáticamente invoca al último valor que se le asignó. Por ejemplo:

```
> a;
```

34

Nótese que Maple nos muestra como resultado el valor asignado. De la misma forma, al incluir el nombre de una variable dentro de una expresión algebraica, Maple evalúa esta expresión tomando en cuenta el valor que contiene la variable.

```
> 3*a + 25*a^2 + 39*a^3;
```

$$1561858$$

Si a esta variable le asignamos otro dato, el valor asignado previamente se pierde:

```
> a := x^2 + alpha + 3;
```

$$a := x^2 + \alpha + 3$$

```
> 3*a + 25*a^2 + 39*a^3;
```

$$3x^2 + 3\alpha + 9 + 25(x^2 + \alpha + 3)^2 + 39(x^2 + \alpha + 3)^3$$

4.1.2 Asignación de valores a una variable

A una variable de Maple se le puede asignar prácticamente cualquier dato válido. Por ejemplo, se pueden asignar números, expresiones aritméticas, relaciones, cadenas, conjuntos, listas, etc. Veamos algunos ejemplos:

```
> var1 := 235^3 + sqrt(29);
```

$$var1 := 12977875 + \sqrt{29}$$

```
> com1 := 23^(1/3) + 8*I;
```

$$com1 := 23^{(1/3)} + 8\,I$$

```
> inq1 := 4*x + 5*x^2 < 10;
```

$$inq1 := 4x + 5x^2 < 10$$

```
> sec1 := dato1, a, 13, 4*25 + 8;
```

$$\mathrm{sec1} := dato1, x^2 + \alpha + 3, 13, 108$$

```
> lis1 := [x^2 + y^2, 8!, 34*29^2, 5.234];
```

$$lis1 := [x^2 + y^2, 40320, 28594, 5.234]$$

```
> cnj1 := {sqrt(23), 'hola a todos', [a, b, c]};
```

$$cnj1 := \{hola\ a\ todos, \sqrt{23}, [x^2 + \alpha + 3, b, c]\}$$

Estas variables pueden ser operadas de la misma forma que el valor que contienen. Por ejemplo:

```
> com1 + 5*com1;
```

$$6\,23^{(1/3)} + 48\,I$$

```
> evalf[30](var1);
```

$$0.129778803851648071345040312507\,10^8$$

También se pueden definir variables a partir de otras variables previamente definidas:

```
> c1 := 92; exp1 := r^3 + p^4 - 35;
```

$$c1 := 92$$

$$\mathrm{exp1} := r^3 + p^4 - 35$$

```
> comb := c1 + c1*exp1 + c1^2*exp1^2;
```

$$comb := -3128 + 92\,r^3 + 92\,p^4 + 8464\,(r^3 + p^4 - 35)^2$$

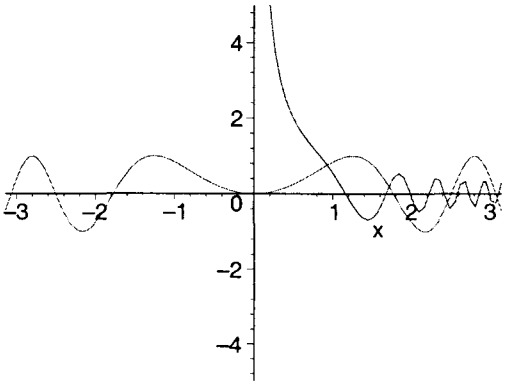
Otro tipo de dato que puede ser asignado a una variable son las gráficas, como veremos a continuación.

```
> grafica1 := plot(sin(x^2), x=-Pi..Pi, numpoints=5):

> grafica2 := plot(cos(x^3)/x, x=0.01..Pi, y=-5..5, color=blue):
```

Al hacer este tipo de asignación Maple muestra todos los datos creados, necesarios para desplegar la gráfica. En este caso se hace evidente la conveniencia de utilizar el caracter *":"* como terminador de instrucción, ya que de esta forma la asignación se lleva a cabo pero no se muestran todos los datos generados.

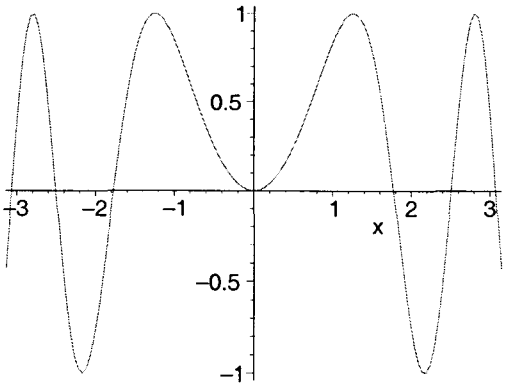
Este tipo de variables, al tener asignadas gráficas de funciones, podemos referenciarlas por medio de una instrucción que pueda manipular los datos de la gráfica. Una de las funciones de Maple que nos permiten hacer esto es **display**, la cual puede tomar tales datos y desplegar la gráfica correspondiente. Esta función forma parte del paquete **plots** y puede usarse de la siguiente forma.

```
> plots[display](grafica1, grafica2);
```



Cuando se crean variables cuyo valor asignado es una gráfica, al evaluarlas en la línea de comandos obtenemos un despliegue de la gráfica. Por ejemplo:

```
> grafica1;
```



Los detalles acerca del uso de la función **display** serán tratados posteriormente en las secciones de gráficas en dos y tres dimensiones.

4.1.3 Nombres de variables

A continuación revisaremos algunas de las reglas que deben seguirse al hacer uso de variables en Maple.

Nombres válidos

En principio, toda expresión de Maple, sintácticamente bien formada, puede ser utilizada como nombre de variable, aunque se deben tener en cuenta algunas restricciones en las palabras utilizadas, como se verá a continuación.

Usualmente los caracteres que se usan para formar nombres de variables son:

- Letras (recuerdese que las mayúsculas son diferentes de las minúsculas).
- Números.
- Guiones de subrayado "_", así como el signo "?", siempre que no aparezca como primer caracter.
- Espacios en blanco, siempre y cuando el nombre sea tratado como una cadena. (Véanse los ejemplos más adelante).
- También es posible incluir subíndices en las variables, aunque se debe tener cuidado con las expresiones usadas dentro de los corchetes.

Otros detalles que deben tenerse en cuenta al asignar un nombre de variable son los siguientes:

- El primer caracter debe ser una letra o un guión bajo, no puede ser un número o el signo "?". Por ejemplo:

```
> variable1?  := 9;
```

$$variable1? := 9$$

```
> _var2 := %*23;
```

$$\_var2 := 207$$

```
> 12var1 := 13; # este no es un nombre válido
Error, missing operator or `;`
```

Nota: aunque son válidos, es recomendable evitar en lo posible el uso de nombres cuyo primer caracter sea un guión de subrayado, ya que Maple utiliza variables internas con esta forma.

- Los corchetes ("[ ]"), solo deben usarse para indicar subíndices, no deben colocarse como caracteres intermedios. Por ejemplo:

```
> num[1] := 23;
```

$$num_1 := 23$$

```
> num[2] := %^2;
```

$$num_2 := 529$$

```
> var[23 := 24; # nombre no valido
Error, `:=` unexpected
```

Nota: las letras o palabras que se usan como subíndices, deben manejarse como cadenas. Esto nos permite evitar efectos colaterales provocados por el uso posterior de la misma expresión. Por ejemplo:

```
> solucion[x=1] := sqrt(29)*5!;
```

$$solucion_{x=1} := 120\sqrt{29}$$

En esta expresión es recomendable encerrar **"x=1"** entre acentos agudos, pues de lo contrario Maple puede considerarla como una asignación, lo cual puede provocar problemas al utilizar posteriormente **"x"**. La forma más recomendable es:

```
> solucion[`x=1`] := sqrt(29)*5!;
```

$$solucion_{x=1} := 120\sqrt{29}$$

De cualquier manera se obtiene el mismo resultado, pero en el segundo caso se evita cualquier problema por el uso posterior de **"x"**.

- Se pueden colocar espacios en blanco en los nombres de variables, siempre y cuando dicho nombre sea tratado como cadena (delimitado por acentos agudos). Por ejemplo:

```
> `solucion 1` := 97;
```

$$solucion\ 1 := 97$$

Al hacer referencia a esta variable también es necesario colocar los acentos.

```
> `solucion 1`^2 + 2*`solucion 1`^3;
```

$$1834755$$

Cuando se manejan este tipo de nombres es más recomendable usar guiones de subrayado en lugar de espacios en blanco:

```
> solucion_1 := 23;
```

$$solucion\_1 := 23$$

```
> solucion_1^2 + 2*solucion_1^3 - 24;
```

$$24839$$

- Otro tipo de nombres válidos son aquellos en los cuales se incluyen paréntesis de la siguiente forma:

```
> dato1(x=0) := 25.35*4.2 + Pi;
```

$$dato1(x = 0) := 106.470 + \pi$$

Al invocar esta variable también deben usarse los paréntesis:

```
> 76*dato1(x=0);
```

$$8091.720 + 76\,\pi$$

Este tipo de variables son válidas, como se mencionó anteriormente; sin embargo, no es recomendable su uso ya que pueden causar confusión con las funciones del sistema y las definidas por el usuario.

Nombres no válidos

En la sección anterior se trató el caso de los tipos de expresiones válidas como nombres de variables en Maple. A continuación se muestran algunas de las expresiones que no deben usarse para este fin:

- No se pueden usar números, constantes o funciones predefinidas por Maple, como nombres de variables. Por ejemplo:

```
> 34 := 6!;
Error, invalid left hand side of assignment
> Pi := 24;
Error, attempting to assign to `Pi` which is protected
> cos := 34;
Error, attempting to assign to `cos` which is protected
> evalf := sin(x)*sqrt(x^2 - 25);
Error, attempting to assign to `evalf` which is protected
```

- Nombres que contengan símbolos de relación, operadores aritméticos, o bien paréntesis o corchetes incompletos. Por ejemplo:

```
> dato<1 := -4.3;
Error, invalid left hand side of assignment
> dato>=5 := 7.9;
Error, invalid left hand side of assignment
> cuatro+5 := 9;
Error, invalid left hand side of assignment
> numero(1 := 34;
Error, `:=` unexpected
> cadena[2a:=`hola a todos`;
Error, missing operator or `;`
```

4.1.4 Variables simbólicas

Generalmente se conceptualiza a una variable como un dato o conjunto de datos que pueden ser referenciados a través de un nombre; sin embargo, Maple también nos permite hacer uso de variables simbólicas, es decir, variables a las cuales no se les ha asignado ningún dato. Por ejemplo, consideremos la siguiente instrucción:

```
> a^2 + 23^3*(5!  - 4) - 5*a^2;
```

$$-4\,(x^2+\alpha+3)^2+1411372$$

Nótese que, la expresión formada por la letra "a" en realidad no tiene asignado ningún valor que pueda ser evaluado, pero Maple de cualquier forma la opera. En este caso, dicha expresión es considerada una variable simbólica, es decir, una variable cuyo valor aun no ha sido definido.

Las variables símbolicas pueden estar dadas por cualquier nombre de variable válido.

4.1.5 Desasignación de una variable.

Cada vez que se define una variable, ésta permanece hasta el fin de la sesión. Por ejemplo, definamos a continuación la variable *"x"*.

```
> x := 34;
```

$$x := 34$$

¿Qué sucede si a continuación deseamos utilizar *"x"* como variable simbólica?

```
> a*x^2 + b*x + c = 0;
```

$$1339804 + 1156\,a + 34\,b + c = 0$$

Obviamente no es posible, pues Maple sustituye la variable por el valor asignado. Para usarla de esta manera, deberíamos poder ***"eliminar"*** o ***"desasignar"*** su valor. Existen dos formas de hacerlo.

Una manera de eliminar una variable es a través de la instrucción **restart** (véase su página de ayuda), la cual también puede ser invocada desde la barra de herramientas por medio del botón . El problema es que esta instrucción elimina toda la información generada desde el inicio de la sesión hasta el momento que es ejecutada. Por lo cual, además de eliminar ***"x"***, eliminamos cualquier otro resultado presente en ese momento.

Una manera más conveniente en este caso, es utilizar lo que se conoce como ***"desasignación"*** de una variable. La sintaxis para esta ***"desasignación"*** es la siguiente:

var:='var';

Es decir, a la variable le asignamos su mismo nombre pero encerrado entre apóstrofes. Por ejemplo, asignemos un nuevo valor a *"x"*.

```
> x := 345;
```

$$x := 345$$

A continuación desasignemos esta variable.

```
> x := 'x';
```

$$x := x$$

Ahora, introduzcamos la siguiente expresión.

```
> d*x^2 + f*x + h = 0;
```

$$d\,x^2 + f\,x + h = 0$$

Como puede verse, después de ***"desasignarla"***, Maple elimina el valor asignado a la variable. Por lo cual, en la instrucción anterior, ***"x"*** es manejada como una variable simbólica.

Esta desasignación de variables también puede llevarse a cabo por medio de la instrucción **unassign**, de la siguiente forma:

unassign('x');

La ventaja de hacer uso de esta función es que nos permite ***"desasignar"*** más de una variable al mismo tiempo. Por ejemplo:

unassign('x', 'y', 'grafica', 'solucion');

Esta instrucción desasigna todas las variables recibidas como argumento.

Capítulo 5

Uso de funciones

Un elemento importantisimo en Maple son las funciones. El sistema por sí mismo ya contiene un conjunto predefinido de éstas, pero también le proporciona al usuario varios mecanismos para definir las suyas.

Al igual que con las variables, las funciones permanecen en la memoria desde su definición y pueden ser invocadas a lo largo de toda la sesión.

5.1 Funciones definidas por el usuario

La manera de crear funciones dadas por el usuario es por medio de la definición de operadores. Esto se hace utilizando la siguiente sintaxis:

nom := var -> regla;

Donde **nom** es el nombre que se le asignará a la función, **var** es la variable de la cual dependerá, y ***regla*** es la regla de correspondencia que se aplicará al invocar dicha función. Entre el nombre de la variable y la regla de correspondencia se debe colocar el operador "->", formado por un signo menos ("-") y un mayor que (">"). Por ejemplo, definamos la función **f**:

```
> f := x -> x + x^2;
```

$$f := x \rightarrow x + x^2$$

Este tipo de funciones pueden ser evaluadas en la forma: **f(n)**, donde **n** es una expresión aritmética, simbólica o una función.

Antes de continuar, definamos la función **g**:

```
> g := x -> 2*x + 3*x^2;
```

$$g := x \rightarrow 2\,x + 3\,x^2$$

Ahora, para evaluar **f** en $x = 25$, $x = \sqrt{23} + 18^2$ y $x = g(34)$, procedemos de la siguiente manera:

```
> f(25);
```

$$650$$

```
> f(sqrt(23) + 18^2);
```

$$\sqrt{23} + 324 + (\sqrt{23} + 324)^2$$

```
> f(g(34));
```

12506832

El procedimiento que debe seguirse para funciones de varias variables es análogo. Para definir la función procedemos de la siguiente forma:

nom := (var1, var2, var3, ...) -> regla;

Nuevamente, **nom** es el nombre de la función, **"(var1, var2, var3, ...)"** son las variables de las cuales depende (deben colocarse entre paréntesis) y **regla** es la regla de correspondencia asignada. Por ejemplo, definamos la siguiente función:

```
> h := (x, y, z) -> x^2 + y^2 + z^2;
```

$$h := (x, y, z) \to x^2 + y^2 + z^2$$

Para poder evaluar h en $x = 3$, $y = 9$, $z = g(x + y)$, procedemos de la siguiente forma:

```
> h(3, 9, g(3 + 9));
```

208026

La función se evaluará tomando el primer dato como el valor de la primer variable definida, el segundo dato como el valor de la segunda variable y así sucesivamente. De esta manera es posible definir funciones que dependan de una o varias variables. Este tipo de funciones también pueden ser utilizadas en instrucciones que reciben funciones como argumento, solamente se debe indicar a Maple que se trata de una función operador y se deben escribir explícitamente las variables de las cuales depende. Por ejemplo, la siguiente instrucción nos da un resultado erróneo:

```
> diff(f, x);
```

0

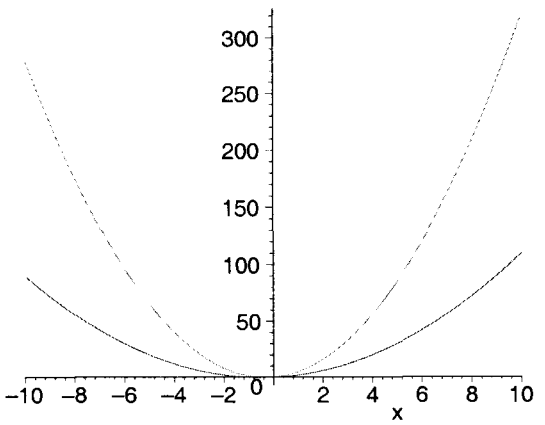
Comparese con la siguiente :

```
> diff(f(x), x);
```

$$1 + 2x$$

A continuación graficaremos **f** y **g** de la siguiente forma:

```
> plot({f(x), g(x)}, x=-10..10);
```



En este caso, al colocar **f** como argumento de **plot** debe expresarse como **f(x)**, de lo contrario Maple no podrá hacer la evaluación correctamente. Lo mismo sucede con funciones de varias variables. Por ejemplo,

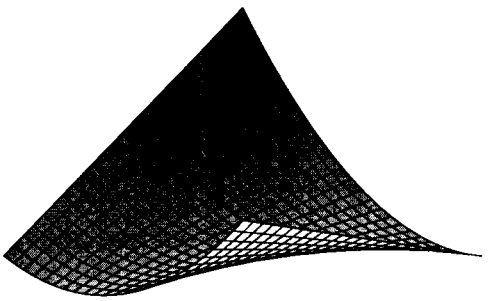
si tenemos una función **d** que depende de ***a***, **b** y ***c***, ésta debe ser siempre invocada como **d(a, b, c)**. Veamos el siguiente ejemplo. Primero definimos una función de dos variables:

```
> p := (e, h) -> e*h + h^2;
```

$$p := (e, h) \rightarrow e\,h + h^2$$

A continuación graficamos **p** para $-5 \leq e \leq 5$, $-4 \leq h \leq 4$

```
> plot3d(p(e, h), e=-5..5, h=-4..4);
```



Existe una forma de evitar tener que escribir explícitamente dichas variables mediante el uso de ***"alias"*** (esto se tratará más adelante). Por último, debemos tener en cuenta que Maple no nos permite hacer una definición recursiva de una función. Por ejemplo, no es válido hacer lo siguiente:

```
> F := x -> sin(x);
```

$$F := \sin$$

```
> F := x -> F(x) + sqrt(25);
```

$$F := x \rightarrow \mathrm{F}(x) + \sqrt{25}$$

Maple nos enviará un mensaje de error al tratar de evaluar esta función :

```
> F(5);
Error, (in F) too many levels of recursion
```

5.2 Composición de funciones

El definir funciones como operadores es bastante útil para poder hacer composición de funciones. Existen varias formas de hacer dicha composición. Una de ellas es simplemente anidar las funciones. Veamos un ejemplo.

Primero definimos las funciones **H** y **K**:

```
> H := a -> sin(a + 1);
```

$$H := a \rightarrow \sin(a + 1)$$

```
> K := a -> exp(1 - a);
```

$$K := a \rightarrow e^{(1-a)}$$

Ahora, calcularemos las composiciones **H(K(w))**, **K(H(w))** y **H(H(H(w)))**.

```
> H(K(w));
```

$$\sin(e^{(1-w)} + 1)$$

```
> K(H(w));
```

$$e^{(1-\sin(w+1))}$$

```
> H(H(H(w)));
```

$$\sin(\sin(\sin(w + 1) + 1) + 1)$$

Otra forma de hacer este último cálculo es utilizando el operador de composición de funciones, de la siguiente forma:

(f1 @ f2 @ f3 @ ... @ fn)(x);

Donde **f1**, **f2**, **f3**, ... , **fn** son las funciones que se desea componer y x es el valor en el cual sera evaluada la composición. Esta expresión nos produce el mismo resultado que si emplearamos lo siguiente:

f1(f2(f3(...(fn(x)))));

Sin embargo, al usar la primera forma nos evitamos el tener que anidar todas las funciones con paréntesis. Utilizando esta notación, calculemos nuevamente las composiciones anteriores:

```
> (H@K)(w);
```

$$\sin(e^{(1-w)} + 1)$$

```
> (K@H)(w);
```

$$e^{(1-\sin(w+1))}$$

```
> (H@H@H)(w);
```

$$\sin(\sin(\sin(w + 1) + 1) + 1)$$

En este último caso, estamos haciendo una composición de la misma función tres veces. Este tipo de operaciones también las podemos realizar utilizando el operador de composición repetida de funciones, cuya notación es:

(f @@ n)(x);

Donde **f** es el nombre de la función, **n** es el número de veces que se desea componer consigo misma y x el valor en el cual se evaluará dicha composición. Por ejemplo, para hacer la composición de **H** consigo misma tres veces:

```
> (H@@3)(w);
```

$$\sin(\sin(\sin(w + 1) + 1) + 1)$$

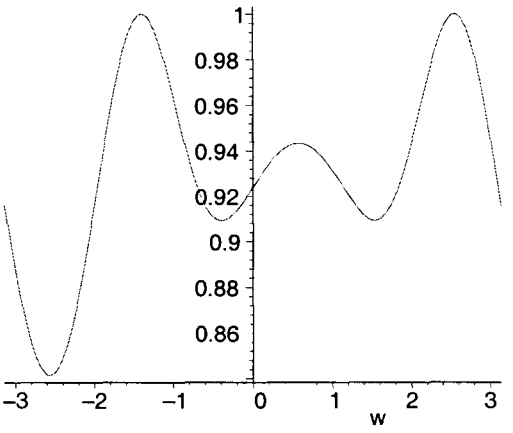
Este último resultado es el mismo que obtenemos al usar **(H@H@H)(w)** o bien **H(H(H(w)))**, solo que se trata de una forma más breve y sencilla. Utilizando esta notación, definiremos la siguiente función:

```
> Q := w -> (H@@3)(w);
```

$$Q := H^{(3)}$$

A continuación graficamos esta función para $-\pi \leq w \leq \pi$.

```
> plot(q, w=-Pi..Pi);
```



De hecho, todas estas formas de composición pueden ser utilizadas con las mismas funciones que proporciona Maple al usuario. Por ejemplo, calculemos el valor numérico de: $\sqrt{\cos(\cos(\cos(\pi^2)))}$.

Una forma de hacerlo es la siguiente:

```
> evalf(sqrt(cos(cos(cos(Pi^2)))));
```

0.9023117120

Otra manera es:

```
> (evalf@sqrt@cos@cos@cos)(Pi^2);
```

0.9023117120

Y existe una opción más:

```
> (evalf@sqrt@cos@@3)(Pi^2);
```

0.9023117120

Nótese que en esta última instrucción utilizamos, además del operador de composición, el operador de composición repetida.

En general, Maple nos permite hacer cualquier composición de funciones, siempre y cuando los datos de salida y entrada de éstas sean compatibles. Por ejemplo, no podemos hacer una composición con las funciones **plot** y **evalf**, pues los datos que manejan son diferentes. Téngase esto en cuenta al hacer composiciones.

5.3 Transformación de expresiones en funciones operador

Considérese la siguiente asignación:

```
> g := x + sin(x);
```

$$g := x + \sin(x)$$

¿Cómo podemos, usando g, obtener una función operador con la misma regla de correspondencia y de tal manera que también dependa de la variable x?

Una opción es hacer lo siguiente:

Asignamos el argumento **x** a la expresión **g** (la regla de correspondencia) de la siguiente forma:

```
> G := x -> g;
```

$$G := x \to g$$

Intentemos evaluar para $x = 5$:

```
> G(5);
```

$$x + \sin(x)$$

Esto nos genera un resultado erróneo, por lo cual se ve inmediatamente que este no es un método apropiado.

Maple nos proporciona un mecanismo a través del cual podemos transformar cualquier expresión en una función operador con la misma regla de correspondencia y que dependa de la misma variable (o variables, según sea el caso), por medio de la instrucción **unapply** (consúltese su página de ayuda). La notación es la siguiente:

unapply(E, x1, x2, x3,...xn);

Donde **E** es la expresión que se desea convertir y **x1, x2, x3, ... ,xn** son las variables de las cuales dependerá la función operador generada (deben ser las mismas variables que aparecen en la expresión **E**). Esta instrucción da como resultado una función operador con las características antes descritas. Por ejemplo, podemos convertir g en operador de la siguiente forma:

```
> G := unapply(g, x);
```

$$G := x \to x + \sin(x)$$

Desde este momento, **G** es considerada una función operador con regla de correspondencia equivalente a la de **g**. Probemos evaluando en **x = Pi/2**:

```
> G(Pi/2);
```

$$\frac{\pi}{2} + 1$$

Incluso, podemos hacer composiciones usando esta función:

```
> (evalf@G@@4)(Pi^3);
```

$$28.31175398$$

El caso de funciones de dos o más variables es equivalente. Veamos el siguiente ejemplo:

```
> d := 4*x + 5*y + x*y;
```

$$d := 4x + 5y + xy$$

Convertimos **d** a operador de la siguiente forma:

```
> funcion := unapply(d, x, y);
```

$$funcion := (x, y) \to 4x + 5y + xy$$

y a continuación la evaluamos para datos particulares:

```
> funcion(6, sqrt(24));
```

$$24 + 22\sqrt{6}$$

También podemos usar esta función para hacer composición con otras funciones:

```
> (evalf@funcion)(6, sqrt(24));
```

$$77.88877435$$

Otra de las ventajas de estas conversiones es que Maple nos permite definir operadores a partir de cualquier expresión, en particular a partir de resultados generados por otras instrucciones. Por ejemplo, calculemos la siguiente integral con respecto a x:

```
> int(1/(a^3 + x^3), x);
```

$$-\frac{1}{6}\frac{\ln(a^2 - a\,x + x^2)}{a^2} + \frac{1}{3}\frac{\sqrt{3}\arctan(\frac{(-a+2\,x)\sqrt{3}}{3\,a})}{a^2} + \frac{1}{3}\frac{\ln(a+x)}{a^2}$$

Este resultado lo asignaremos a la variable **res** pero en forma de operador y evaluaremos en:
a = .32, x = 2.1

```
> res := unapply(%, a, x);
```

$$res := (a, x) \rightarrow -\frac{1}{6}\frac{\ln(a^2 - a\,x + x^2)}{a^2} + \frac{1}{3}\frac{\sqrt{3}\arctan(\frac{1}{3}\frac{(-a+2\,x)\sqrt{3}}{a})}{a^2} + \frac{1}{3}\frac{\ln(a+x)}{a^2}$$

```
> res(.32, 2.1);
```

$$0.686781499 + 3.255208333\,\sqrt{3}\arctan(4.041666667\,\sqrt{3})$$

Esta es una forma en la cual podemos retener expresiones generadas como resultado de alguna operación, para poder evaluarlas o manipularlas posteriormente.

5.4 Funciones predefinidas

Existen varias funciones de Maple que se encuentran disponibles automáticamente cada vez que iniciamos una sesión en este sistema. Todas ellas son conocidas como ***"funciones predefinidas"*** y no requieren de ninguna definición por parte del usuario para poder ser utilizadas, simplemente deben ser invocadas con sus respectivos argumentos. Entre ellas se encuentran las siguientes:

Funciones trigonométricas, trigonométricas inversas e hiperbólicas

A continuación se muestran las funciones trigonométricas soportadas por Maple, así como sus respectivas inversas, también se proporciona una liga a su página de ayuda.

A continuación, la tabla 5.1 muestra las funciones trigonométricas soportadas por Maple, así como sus respectivas inversas.

Nombre	**Función**	**Inversa**
Seno	sin(x)	arcsin(x)
Coseno	cos(x)	arccos(x)
Tangente	tan(x)	arctan(x)
Cotangente	cot(x)	arccot(x)
Secante	sec(x)	arcsec(x)
Cosecante	csc(x)	arccsc(x)

Table 5.1: Funciones trigonométricas

La tabla 5.2 muestra las funciones trogonométricas hiperbólicas y sus respectivas inversas.

Nombre	Función	Inversa
Seno hiperbólico	sinh(x)	arcsinh(x)
Coseno hiperbólico	cosh(x)	arccosh(x)
Tangente hiperbólica	tanh(x)	arctanh(x)
Cotangente hiperbólica	coth(x)	arccoth(x)
Secante hiperbólica	sech(x)	arcsech(x)
Cosecante hiperbólica	csch(x)	arccsch(x)

Table 5.2: Funciones trigonométricas hiperbólicas

Nota: la hoja de ayuda de estas funciones puede consultarse en la forma: **?función**

Todas estas funciones son calculadas en radianes. Algunas de ellas, particularmente el seno y el coseno, ya habían sido usadas con anterioridad. La forma de utilizar las demás es análoga. Por ejemplo:

```
> sin(Pi/8);
```

$$\sin(\frac{\pi}{8})$$

```
> coth(3.1 + 2.5*I);
```

$$1.001144421 + 0.003896610898\,I$$

```
> diff(arctanh(x), x);
```

$$\frac{1}{1-x^2}$$

```
> (sin@cos@arctan)(.2);
```

$$0.8308206799$$

Estas funciones también pueden ser utilizadas para hacer composiciones; incluso es válido usar expresiones como la siguiente para calcular inversas:

```
> sin@@(-1);
```

$$\arcsin$$

Por ejemplo, para evaluar **arctan** en 0.5, podemos utilizar :

```
> (tan@@(-1))(.5);
```

$$0.4636476090$$

Exponencial y logaritmo

La función exponencial está disponible en Maple como: **exp(x)**. Veamos algunos ejemplos de su uso:

```
> exp(5);
```

$$e^5$$

```
> evalf(%);
```

$$148.4131591$$

```
> evalf(cos(exp(44)));
```

$$-0.1618529225$$

Acerca del *logaritmo*, Maple proporciona tres funciones para calcularlo:

- $ln(x)$. Calcula el logaritmo natural de x.
- $log[b](x)$. Calcula el logaritmo en base *"b"*, de *"x"*.
- $log10(x)$. Calcula el logaritmo en base 10.

Estas funciones son evaluadas por Maple de la siguiente forma:

- El logaritmo natural, **ln**, es el logaritmo con base $exp(1) = 2.71828...$
  Para $x > 0$ tenemos que: $ln(x) = y \Longleftrightarrow x = exp(y)$.
- Dada una expresión compleja **x**, su logaritmo natural está dado por: $ln(x) = ln(abs(x)) + I * argument(x)$, donde $-\pi < argument(x) \leq \pi$. La función **abs** calcula el valor absoluto (módulo), de **x**, mientras que **argument** calcula el argumento. En Maple, este cálculo es considerado como la definición de la rama principal del logaritmo.
- La función **log** calcula el logaritmo en general, para cualquier base dada. Para $x > 0$ y $b > 0$, tenemos que $log[b](x) = y \Longleftrightarrow x = b^y$. Esta función es extendida para complejos b y x en general, por medio de la fórmula: $log[b](x) = ln(x)/ln(b)$.
- En el caso de la función **log** , el valor por default para la base **b** es **exp(1)**.
- La función **log10(x)**, es equivalente a **log[10](x)**.

Recuérdese que la función logaritmo solo acepta argumentos mayores que cero.

```
> exp(ln(9));
```

$$9$$

```
> ln(exp(-14));
```

$$-14$$

Veamos ahora algunos ejemplos con complejos:

```
> ln(4 + 9.5*I);
```

$$2.332897404 + 1.172273881\, I$$

```
> ln(3 + 4*I);
```

$$\ln(3 + 4\, I)$$

En este úlimo ejemplo, Maple no hace la evaluación ya que solo estamos incluyendo números enteros como argumento de **ln**. En este caso es necesario aplicar la función **evalc** para evaluación numérica de complejos y después **evalf**:

```
> evalc(ln(3 + 4*I));
```

$$\ln(5) + \arctan(\frac{4}{3})\, I$$

```
> evalf(%);
```

$$1.609437912 + 0.9272952179\, I$$

Veamos otro ejemplo:

```
> log10(10000);
```

$$\frac{\ln(10000)}{\ln(10)}$$

Esta última instrucción nos dá el logaritmo en base 10 de 10000; es equivalente a: **evalf(log[10](10000))**.

Otras funciones básicas

La tabla 5.3 muestra otras funciones comúnmente usadas, disponibles en Maple:

Nombre	**Función**
Raiz cuadrada	sqrt(x)
Valor absoluto	abs(x)
Límite de f en x	limit(f, x)
Derivada de f respecto a x	diff(f, x)
Factorial	fact(x) ó !x
Máximo entero menor o igual	floor(x)
Mínimo entero mayor o igual	ceil(x)
Redondeo	round(x)
Truncamiento	trunc(x)
Máximo y mínimo de una secuencia	max(sec), min(sec)
Parte fraccionaria	frac(x)
Módulo	a mod b
Conjugado de un complejo	conjugate(c)
Argumento de un complejo	argument(c)
Parte real e imaginaria de un complejo	Re(c), Im(c)
Forma polar de un complejo	polar(c)

Table 5.3: Otras funciones disponibles en Maple

Una lista completa de las funciones matemáticas más comúnmente utilizadas, puede ser consultada en la página de ayuda de **inifcn**.

Existen otras funciones predefinidas disponibles en los diversos paquetes de Maple. Por ejemplo, el paquete **LinearAlgebra** nos proporciona varias de ellas, útiles para operaciones de álgebra lineal; mientras que el paquete **stats** nos proporciona una serie de funciones estadísticas. Una lista completa de los paquetes disponibles puede consultarse en la hoja de ayuda **index[package]**. Más adelante se tratarán algunas de estas funciones predefinidas.

5.5 Funciones sin regla de correspondencia

En Maple es posible definir una función sin dar su regla de correspondencia (de la misma forma que podemos usar variables simbólicas). Por ejemplo:

```
> B := x -> b(x);
```

$$B := b$$

Nota: Si la regla de correspondencia no ha sido especificada (como es el caso), no podemos usar el mismo nombre para la función y para la regla de correspondencia (por ejemplo $B := x - > B(x)$).

Para evaluar la función $\boldsymbol{B}$ en $\boldsymbol{x = 3}$, seguimos la regla de evaluación de cualquier función operador:

```
> B(3);
```

$$\mathrm{b}(3)$$

Podemos combinar funciones de este tipo con funciones predefinidas y con aquellas definidas por el usuario, cuya regla de correspondencia haya sido especificada. Por ejemplo:

```
> F := x -> x^2;
```

$$F := x \to x^2$$

```
> C := x -> B(x) + sin(x) + sqrt(x) - F(x);
```

$$C := x \to \mathrm{B}(x) + \sin(x) + \sqrt{x} - \mathrm{F}(x)$$

```
> C(2);
```

$$\mathrm{b}(2) + \sin(2) + \sqrt{2} - 4$$

5.6 Forma inerte de una función

Entre las funciones predefinidas existen algunas que soportan dos modos de operación. Uno de estos modos es el que hemos estado usando hasta aquí, en el cual incluimos un argumento en la función y ésta nos da el resultado de aplicar la regla de correspondencia que tiene asignada. En el otro modo, al incluir un argumento, la función nos da una expresión matemática que representa la operación solicitada. Esta última es conocida como la ***"forma inerte"*** de la función. Por ejemplo, una de las funciones que soporta estos dos modos de operación es **limit**, que nos calcula el límite de una función alrededor de un punto. La forma de invocar esta función para calcular un límite es:

limit(función, punto);

Limit(función, punto);

La segunda instrucción corresponde a la forma inerte.

En general, en las funciones que soportan una forma inerte, ésta puede ser invocada colocando la primer letra del nombre en mayúscula. Para visualizar la diferencia entre estos dos modos conaiderense las siguientes instrucciones:

```
> limit(tan(x), x=Pi/4);
```

$$1$$

```
> Limit(tan(x), x=Pi/4);
```

$$\lim_{x \to (\frac{\pi}{4})} \tan(x)$$

Nótese el resultado de la segunda instrucción, ésta nos da no el límite, sino una expresión no evaluada, en simbología matemática, que representa dicha operación. Tales expresiones pueden ser evaluadas, como vimos anteriormente, utilizando le instrucción **value**.

```
> value(%);
```

$$1$$

Algunas otras funciones que soportan una forma inerte son:

- $diff(f(x), x)$ o $Diff(f(x), x)$. Calcula la diferencial de f con respecto a x.
- $int(f(x), x)$ o $Int(f(x), x)$. Calcula la integral de f respecto a x.
- $sum(expr(x), x = a..b)$ o $Sum(expr(x), x = a..b)$. Calcula la sumatoria para la expresión $expr(x)$, haciendo variar x desde a hasta b, es decir: $expr(a) + expr(a+1) + expr(a+2) + ... + expr(b)$.
- $product(expr(x), x = a..b)$ o $Product(expr(x), x = a..b)$. Calcula el producto de la expresión dada, haciendo variar x desde a hasta b, es decir: $expr(a) * expr(a+1) * expr(a+2) * ... * expr(b)$.

Por ejemplo:

```
> diff(sin(x) + cos(sqrt(x4)), x);
```

$$\cos(x)$$

```
> Diff(sin(x) + cos(sqrt(x4)), x);
```

$$\frac{\partial}{\partial x}(\sin(x) + \cos(\sqrt{x4}))$$

```
> Diff(sin(x) + cos(sqrt(x4)), x) = diff(sin(x) + cos(sqrt(x4)), x);
```

$$\frac{\partial}{\partial x}(\sin(x) + \cos(\sqrt{x4})) = \cos(x)$$

```
> Int(x^2 + y^3 + x^4, x) = int(x^2 + y^3 + x^4, x);
```

$$\int x^2 + y^3 + x^4 \, dx = \frac{1}{3}x^3 + y^3 x + \frac{1}{5}x^5$$

```
> Sum(a*x^2 + x/(x^4 + 1), x=-10..10) = sum(a*x^2 + x/(x^4 + 1),
> x=-10..10);
```

$$\sum_{x=-10}^{10} (a\,x^2 + \frac{x}{x^4+1}) = 770\,a$$

```
> Product(x^(1/x) + x/4, x=2..7) = product(x^(1/x) + x/4, x=2..7);
```

$$\prod_{x=2}^{7} (x^{(\frac{1}{x})} + \frac{x}{4}) = (\sqrt{2} + \frac{1}{2})(3^{(1/3)} + \frac{3}{4})(4^{(1/4)} + 1)(5^{(1/5)} + \frac{5}{4})(6^{(1/6)} + \frac{3}{2})(7^{(1/7)} + \frac{7}{4})$$

```
> Product(x^(1/x) + x/4, x=2..7) = evalf(product(x^(1/x) + x/4, x=2..7));
```

$$\prod_{x=2}^{7} (x^{(\frac{1}{x})} + \frac{x}{4}) = 232.9768924$$

Capítulo 6

Operaciones aritméticas en Maple

Gracias a los diversos tipos de datos disponibles, a las constantes simbólicas (por ejemplo **Pi**), a las funciones aritméticas y de manipulación numérica con las que cuenta, es posible utilizar Maple para realizar cálculos numéricos y evaluar expresiones aritméticas. Esto, aunado a su capacidad para manejar números de precisión infinita y al manejo de aritmética racional, real y compleja, nos permiten considerar a este sistema como un poderoso ambiente para el manejo de operaciones artitméticas en general. Tales características de Maple son tratadas a lo largo de esta sección.

6.1 Funciones aritméticas

Anteriormente presentamos los operadores artiméticos soportados por Maple, así como los operadores relacionales y algunas de las funciones elementales predefinidas en este sistema (vease el capítulo **Elementos Básicos de Maple** y el capítulo **Uso de Funciones**). Además de las ya mencionadas, existen algunas otras funciones, útiles en la manipulación de expresiones aritméticas. Entre ellas tenemos las que se muestran en la tabla 6.1.

Nombre	**Función**
Cociente entero	iquo(x)
Residuo entero	irem(x)
Máximo común divisor entre a y b:	mcd(a, b)
Mínimo común multiplo entre a y b	lcm(a, b)
Signo	signum(x)
Signo complejo	csignum(x)
Factorización en productos de primos	ifactor(x)

Table 6.1: Funciones aritméticas

Otras funciones útiles para realizar cálculos numéricos son las trigonométricas, con sus respectivas formas inversas, hiperbólicas e hiperbólicas inversas, además de los operadores lógicos (todos ellos mencionados anteriormente). Ejemplos sobre el uso de éstos son presentados en las siguientes secciones.

6.2 Uso de Maple como calculadora

Maple tiene la capacidad de realizar cualquier tipo de operación aritmética, por ejemplo:

```
> 4 + 5;
```

$$9$$

```
> 123.45*5423 - 82971;
```

$$586498.35$$

```
> 892/432.98765;
```

$$2.060104948$$

```
> 7/36 + 5/8 + 3/5;
```

$$\frac{511}{360}$$

Además, puede manipular números muy grandes, tanto enteros como de punto flotante (en ocasiones es utilizada la notación decimal).

```
> 100!;
```

$$933262154439441526816992388562667004907159682643816214685929638952175999932299156089414639761565182862536979208272237582511852109168640000000000000000000000$$

```
> exp(245.0);
```

$$0.2524341263\,10^{107}$$

Nota: además del uso tradicional que se le da al operador "/" para expresar cocientes en la forma: **a/b**, Maple 8 también permite usarlo de la siguiente manera:

`/`(a, b)

más aun, puede ser usado como:

`/`(c)

lo cual es equivalente a la expresión:

1/c

Nótese que para hacer uso del operador como se describe aqui, éste debe estar delimitado por acentos agudos.

Por ejemplo:

```
> `/`(7,36) = 7/36;
```

$$\frac{7}{36} = \frac{7}{36}$$

```
> `/`(Pi*3!)  = 1/(Pi*3!);
```

$$\frac{1}{6\,\pi} = \frac{1}{6\,\pi}$$

Este sistema nos proporciona diversas funciones por medio de las cuales podemos hacer manipulaciones sobre expresiones aritméticas o números, por ejemplo:

- Obtención de la parte fraccionaria.

```
> frac(evalf(sqrt(2)));
```

$$0.414213562$$

- Factorización de un entero, en productos de potencias de números primos.

```
> ifactor(-29!);
```

$$-(2)^{25}\ (3)^{13}\ (5)^{6}\ (7)^{4}\ (11)^{2}\ (13)^{2}\ (17)\ (19)\ (23)\ (29)$$

Este tipo de expresiones aritméticas pueden ser expandidas usando la función **expand**:

```
> expand(%);
```

$$-8841761993739701954543616000000$$

- Obtención del máximo común divisor y el mínimo común múltiplo para dos enteros.

```
> gcd(26764, 4686);
```

$$2$$

```
> lcm(49234, 18932);
```

$$466049044$$

- Podemos verificar si una expresión arirmética corresponde a un número primo.

```
> isprime(11);
```

true

```
> isprime(23!);
```

false

```
> isprime(1323);
```

false

- También podemos evaluar expresiones como la conjetura de Fermat (según la cual $2^{(2^n)} + 1$ es un número primo para n natural), para diferentes valores. Probaremos para $n = 1, 2, 3, 4, 5$.

```
> isprime(2^(2^1) + 1);
```

true

```
> isprime(2^(2^2) + 1);
```

true

```
> isprime(2^(2^3) + 1);
```

true

```
> isprime(2^(2^4) + 1);
```

true

```
> isprime(2^(2^5) + 1);
```

false

- Podemos calcular el primer primo mayor que un número cualquiera. Por ejemplo, calcularemos el primer número primo mayor que 4568.

```
> nextprime(4568);
```

$$4583$$

Verifiquemos el resultado:

```
> isprime(%);
```

$$true$$

- También podemos calcular el primer primo menor que un número dado. Calculemos el primero menor que 4568 y verifiquemos el resultado:

```
> prevprime(4568);
```

$$4567$$

```
> isprime(%);
```

$$true$$

Recuérdese que generalmente Maple realiza las operaciones utilizando artimética racional, lo cual quiere decir que las operaciones con enteros o fracciones siempre generan resultados enteros o fracciones. Por ejemplo:

```
> 21!  + 56^2 + 8/5 - 16/9;
```

$$\frac{2299092397726924941112}{45}$$

No obstante, Maple realiza automáticamente algunas simplificaciones sencillas sobre racionales de tal manera que sean expresados en su forma más simple. Por ejemplo:

```
> 4/8;
```

$$\frac{1}{2}$$

```
> 876324/876432;
```

$$\frac{73027}{73036}$$

Cuando trabajamos con expresiones racionales, es necesario aplicar **evalf** para obtener un número de punto flotante, o bien, se debe incluir en la expresión por lo menos un número de punto flotante.

```
> evalf(21!  + 56^2 + 8/5 - 16/9);
```

$$0.5109094217\,10^{20}$$

```
> 21!  + 56^2 + 8/5 - 16/9.0;
```

$$0.5109094217\,10^{20}$$

Este forma de operar de Maple permite obtener resultados más exactos, pues se evitan todos los errores que puedan surgir al hacer redondeos o aproximaciones de números.

- Por otro lado, podemos también calcular el residuo y el cociente de una expresión artitmética racional:

```
> irem(89234, 45);
```

$$44$$

```
> iquo(89234, 45);
```

$$1982$$

- Otras operaciones útiles son las exponenciaciones:

```
> 8^8;
```

$$16777216$$

```
> 25^(2^3);
```

$$152587890625$$

En estas operaciones es necesario utilizar paréntesis para asociar. Por ejemplo:

```
> 2^2^2^2;
```

```
Error, `^` unexpected
```

Nos genera un mensaje de error, la forma correcta es:

```
> 2^(2^(2^2));
```

$$65536$$

Las operaciones de exponenciación pueden ser utilizadas en general para calcular x^p, donde p es un exponente entero, racional o decimal. Por ejemplo:

```
> 4^(1/2);
```

$$\sqrt{4}$$

```
> evalf(%);
```

$$2.000000000$$

```
> evalf(8^(2.8));
```

$$337.7940252$$

```
> evalf(34^Pi);
```

$$64756.43258$$

- También podemos calcular modulos:

```
> 98762 mod 8;
```

$$2$$

```
> 8!  mod 3;
```

$$0$$

```
> 9749867432 mod 7;
```

$$4$$

La expresión "a mod b" nos da como resultado el residuo de la división entera $\frac{a}{b}$. Recuérdese que esta operación únicamente está definida para $\boldsymbol{a}$ y $\boldsymbol{b}$ enteros.

Veamos otro ejemplo. Comprobaremos a continuación que $\sqrt{2}+\sqrt{3}+\sqrt{5}$ es una raíz del polinomio $x^8 - 40\,x^6 + 352\,x^4 - 960\,x^2 + 576$.

```
> subs(x = sqrt(2) + sqrt(3) + sqrt(5), x^8 - 40*x^6 + 352*x^4 -
> 960*x^2 + 576);
```

$$\%1^8 - 40\,\%1^6 + 352\,\%1^4 - 960\,\%1^2 + 576$$
$$\%1 := \sqrt{2}+\sqrt{3}+\sqrt{5}$$

```
> evalf(%);
```

$$0.00007$$

Existen varias funciones más que nos permiten llevar a cabo operaciones sobre números, entre ellas se encuentras las siguientes:

- **add**. Nos permite calcular la suma de una suceción de valores.
- **mul**. Nos permite calcular el producto de una sucesión de valores.
- **Si, Ci**. Estas funciones están definidas de la siguiente forma: Para toda **x** compleja:
  Si(x) = int(sin(t)/t, t=0..x) Ci(x) = gamma + ln(x) + int((cos(t)-1)/t, t=0..x)
- **factorset(n)**. Calcula los factores primos de un entero **n**.
- **ithprime(i)**. Calcula el i-ésimo primo.
- **Randpoly, Randprime**. Permiten calcular polinomios aleatorios sobre un campo finito.
- **rand**. Genera números aleatorios.
- **numer, denom**. Da como resultado el numerador y denominador de una expresión, respectivamente.
- **frem**. Esta función calcula **r = x - y*n**, donde **n** es el entero más cercano a **x/y**.
- **irem, iquo**. Calculan el residuo entero y cociente entero de dos expresiones, respectivamente.
- **rem, quo**. Calculan el residuo y cociente de dos polinomios.

También puede consultarse la ayuda del paquete **numtheory**, para obtener información de diversas funciones propias de Teoría de Números.

6.3 Precisión de una operación y aproximaciones de punto flotante

Maple es capaz de manejar números de punto flotante de precisión arbitraria, para ello es necesario utilizar la función **evalf**; ésta nos permite indicar a Maple con cuantos dígitos debe realizarse un cálculo numérico, con lo cual podemos obtener aproximaciones de diferente orden, para expresiones aritméticas.

Para ejemplificar lo anterior, calcularemos a continuación el cuadrado del número **Pi** con quinientos dígitos de precisión:

```
> evalf(Pi^2, 500);
```

```
9.869604401089358618834490999876151135313699407240790626413349376\
2004482241920524300177340371855223182402591377402314407777\
2\
3481220300467276106176779851976609903998562065756305715060\
4\
1232840328780869352769342164939666571519044538735261779413\
8\
2025826058169341251559204830981887327003307626667110435895\
0\
8715041003257885365952763577528379226833187450864045463541\
2\
5026973729566958334227858150006365227095472490859756072669\
2\
6475277900528533645220666980826415896877105732788929174690\
1\
5455100692544324570363
```

A continuación obtendremos los primeros 200 dígitos de la parte fraccionaria del resultado de la instrucción anterior:

```
> frac(evalf(%, 200));
```

$$0.869604401089358618834490999876151135313699407240790626413349376 2\backslash$$
$$2004482241920524300177340371855223182402591377402314407777 2\backslash$$
$$3481220300467276106176779851976609903998562065756305715060 4\backslash$$
$$12328403287808694$$

De forma predeterminada, Maple solo muestra los primeros 10 dígitos en cualquier operación de punto flotante (tomando en cuenta parte entera y fraccionaria). Por ejemplo:

```
> evalf(exp(4.8));
```

$$121.5104175$$

Este número de dígitos está determinado por una variable global definida por Maple, conocida como **Digits**. (Consúltese la página de ayuda de **Digits** y otras variables de ambiente usando **?envvar**). Veamos el valor que contiene esta variable:

```
> Digits;
```

$$10$$

Al modificar el valor de esta variable, también modificamos el número de dígitos utilizados al hacer este tipo de evaluaciones. Por ejemplo, asignémosle el valor de 25:

```
> Digits := 25;
```

$$Digits := 25$$

Calculemos nuevamente evalf($e^{4.8}$):

```
> evalf(exp(4.8));
```

$$121.5104175187348807570481$$

Nótese como ahora obtenemos 25 dígitos en lugar de 10.

La cantidad de dígitos a desplegar también puede ser ajustado por medio de la opción **Preferences** del menú **File**, seleccionando la ficha **Numerics**.

6.4 Números complejos

Los números complejos son operados de la misma forma que los reales. Estos números son considerados por Maple como polinómios en términos del símbolo **I**, el cual representa la raíz cuadrada de menos uno. Maple utiliza siempre la relación $i^2 + 1 = 0$ para evaluar expresiones que contienen este tipo de números. Veamos un ejemplo:

```
> (4 + 2*I)/(2 - 5*I);
```

$$\frac{-2}{29} + \frac{24}{29} I$$

```
> evalf(%);
```

$$-0.068965517241379310344827590 + 0.82758620689655172413793100\, I$$

Podemos observar que al aplicar **evalf** a un complejo, esta función evalúa tanto la parte real como la parte imaginaria de este número.

```
> (32768 - sin(234)*I)*(9879 + exp(4)*Pi*I);
```

$$(32768 - \sin(234)\, I)\,(9879 + e^4\, \pi\, I)$$

Nótese que este último resultado no está evaluado. Existe una función que nos permite hacer este tipo de evaluaciones para complejos, su nombre es **evalc** (vease su ***página de ayuda***).

```
> evalc(%);
```

$$323715072 + \sin(234)\, e^4 \pi + (-9879 \sin(234) + 32768\, e^4 \pi)\, I$$

Otras funciones útiles en el manejo de complejos son:

- **Re**. Parte real de un complejo.
- **Im**. Parte imaginaria.
- **conjugate**. Conjugado complejo.
- **abs**. Módulo.
- **argument**. Argumento de un complejo.
- **csgn**. Para un número complejo determina en que parte del plano se encuentra éste.
- **signum**. Calcula el signo de una expresión real o compleja.
- **polar**. Calcula la forma polar de un número complejo.

Veamos a continuación algunos ejemplos.

Considérese el siguiente número:

```
> com1 := (12/4746 + 87641*I)*(234/92 - sin(9)*I);
```

$$com1 := \left(\frac{2}{791} + 87641\, I\right)\left(\frac{117}{46} - \sin(9)\, I\right)$$

Calcularemos sus partes real e imaginaria:

```
> Re(com1);
```

$$\frac{117}{18193} + 87641 \sin(9)$$

```
> Im(com1);
```

$$-\frac{2}{791}\sin(9) + \frac{10253997}{46}$$

Calcularemos su conjugado, su módulo, su argumento y su signo como número complejo:

```
> conjugate(com1);
```

$$\left(\frac{2}{791} - 87641\, I\right)\left(\frac{117}{46} + \sin(9)\, I\right)$$

```
> abs(com1);
```

$$\frac{1}{36386}\sqrt{4805821274088965}\,\sqrt{13689 + 2116\sin(9)^2}$$

```
> argument(com1);
```

$$\arctan\left(\frac{-\frac{2}{791}\sin(9) + \frac{10253997}{46}}{\frac{117}{18193} + 87641\sin(9)}\right)$$

```
> csgn(com1);
```

$$1$$

Este último resultado es calculado de la siguiente forma:

- **1**: si la parte real es positiva o si la parte real es cero y la imaginaria positiva.
- **0**: si el complejo es 0, y
- **-1**: en otro caso.

Podemos incluir variables dentro de expresiones dadas en términos de complejos. En este caso, Maple asume que dichas variables representan números reales, por lo cual son manejadas como tales. Por ejemplo:

```
> (a + b*I)*c;
```

$$(a + bI)\,c$$

```
> Re(%);
```

$$\Re((a + bI)\,c)$$

```
> Im(%%);
```

$$\Im((a + bI)\,c)$$

Las expresiones obtenidas representan las partes real e imaginaria del número complejo, se muestran de forma simbólica ya que contienen valores indeterminados.

6.5 Sumatorias

Maple puede efectuar sumas tanto de números reales como de complejos. Una forma de hacer esta operación es a través de la función **sum**. La sintaxis es la siguiente:

sum(expr(x), x=a..b);

Donde **expr(x)** es una expresión (real o compleja), dada en términos de **x**; mientras que **a** y **b** son enteros, tales que **a<b**. Esta función calcula la suma **expr(a) + expr(a + 1) + expr(a + 2) +...+ expr(b)**. Veamos algunos ejemplos:

```
> sum(x + x^2*sin(x), x=1..10);
```

$$55 + \sin(1) + 4\sin(2) + 9\sin(3) + 16\sin(4) + 25\sin(5) + 36\sin(6) + 49\sin(7) + 64\sin(8) + 81\sin(9) + 100\sin(10)$$

```
> evalf(%);
```

$$89.098593377474518314059243$$

```
> sum((4*r + Pi*r*I), r=2..20);
```

$$836 + 209\,I\,\pi$$

```
> evalf(%);
```

$$836. + 656.5928646002667868386924\,I$$

Recuérdese que esta función tiene una forma inerte. Veamos otros ejemplos:

```
> Sum((1 - i)/(1 + i), i=1..15);
```

$$\sum_{i=1}^{15} \frac{1-i}{1+i}$$

```
> value(%);
```

$$\frac{-3689561}{360360}$$

```
> evalf(%);
```

$$-10.23854201354201354201354$$

Esta función también nos permite manipular sumas indefinidas:

```
> Sum((1 - i)/(1 + i), i=1..n);
```

$$\sum_{i=1}^{n} \frac{1-i}{1+i}$$

```
> subs(n = 17, %);
```

$$\sum_{i=1}^{17} \frac{1-i}{1+i}$$

```
> evalf(%);
```

$$-12.00978384360737301913772$$

Veamos algunos ejemplos más con números complejos:

```
> Sum((2 + k*I)/(2 - k*I), k=0..10);
```

$$\sum_{k=0}^{10} \frac{2+kI}{2-kI}$$

```
> evalc(%);
```

$$\left(\sum_{k=0}^{10}\left(\frac{4}{4+k^2}-\frac{k^2}{4+k^2}\right)\right)+\left(\sum_{k=0}^{10}\left(\frac{4k}{4+k^2}\right)\right)I$$

```
> evalf(%);
```

$$-4.46912213661802241541228 9+6.619767013957377155356412\,I$$

```
> sum((234 + sin(t)^2*I), t=1..10);
```

$$2340+\sin(2)^2\,I+\sin(9)^2\,I+\sin(4)^2\,I+\sin(6)^2\,I+\sin(8)^2\,I+\sin(10)^2\,I+\sin(1)^2\,I$$
$$+\sin(3)^2\,I+\sin(5)^2\,I+\sin(7)^2\,I$$

```
> evalf(%);
```

$$2340.+5.00143063348552053566858 8\,I$$

Otra forma de calcular sumas es por medio de la instrucción **add**, la cual nos permite obtener la suma de una sucesión de valores; su sintaxis es similar a la de **sum**. Por ejemplo:

```
> add(sin(i)^2, i=1..20);
```

$$\sin(1)^2+\sin(2)^2+\sin(3)^2+\sin(4)^2+\sin(5)^2+\sin(6)^2+\sin(7)^2+\sin(8)^2+\sin(9)^2$$
$$+\sin(10)^2+\sin(11)^2+\sin(12)^2+\sin(13)^2+\sin(14)^2+\sin(15)^2+\sin(16)^2$$
$$+\sin(17)^2+\sin(18)^2+\sin(19)^2+\sin(20)^2$$

```
> evalf(%);
```

$$10.29712660081824500630002$$

A diferencia de **sum**, la función **add** no tiene una forma inerte y no puede realizar sumas simbólicas, solo puede sumar datos numéricos explícitos.

6.6 Productos

Otro tipo de operaciones válidas son los productos, tanto de números reales como de complejos. Una forma de calcularlos es por medio de la función **product**, su sintaxis es la siguiente:

product(expr(x), x=a..b);

En donde **expr(x)** es una expresión (real o compleja) en términos de **x**, y **a**, **b** son números enteros tales que **a<b**. Esta función también tiene una forma inerte. Consúltese su ***página de ayuda***.Veamos algunos ejemplos:

```
> Product((j + 2)/(j + 3), j=1..10) = product((j + 2)/(j + 3), j=1..10);
```

$$\prod_{j=1}^{10} \frac{j+2}{j+3} = \frac{3}{13}$$

```
> product(d!*sin(d), d=1..20);
```

$$\begin{aligned}&127442031238161076418723266959124572809405357648039968146460747 86\backslash\\ &210860158451928733118473415711895225119341900569911702323 20\backslash\\ &000000000000000000000000000000000\sin(1)\sin(2)\sin(3)\sin(4)\sin(5)\sin(6)\\ &\sin(7)\sin(8)\sin(9)\sin(10)\sin(11)\sin(12)\sin(13)\sin(14)\sin(15)\sin(16)\sin(17)\\ &\sin(18)\sin(19)\sin(20)\end{aligned}$$

```
> evalf(%);
```

$$-0.16796851547078583526236851 \, 10^{152}$$

Esta función también nos permite manipular productos indefinidos:

```
> product((k + 3)/k^2, k=1..n);
```

$$\frac{1}{6}\,\frac{\Gamma(n+4)}{\Gamma(n+1)^2}$$

```
> subs(n = 10, %);
```

$$\frac{1}{6}\,\frac{\Gamma(14)}{\Gamma(11)^2}$$

```
> evalf(%);
```

$$0.00007881393298059964726631393$$

Además, también se pueden calcular productos de complejos:

```
> product(I/(j + 3*I), j=1..10);
```

$$\frac{-7309}{458967717000} - \frac{5401}{1376903151000}\,I$$

```
> evalf(%);
```

$$-0.1592486733\,10^{-7} - 0.3922570731\,10^{-8}\,I$$

```
> Product(n*I/(4 + 2*I), n=6..15) = evalf(product(n*I/(4 + 2*I),
> n=6..15));
```

$$\prod_{n=6}^{15} \left(\frac{1}{10} + \frac{1}{5}\,I\right) n = 258.2656877 - 3395.594442\,I$$

Otra función que nos permite calcular productos es **mul**, la cual calcula el producto de una secuencia explícita de valores. A diferencia de **product**, esta función no tiene una forma inerte. Su sintaxis es similar a la de **product**. Por ejemplo:

```
> datos:=[2, Pi, 4, 5*Pi, 8, 10];
```

$$datos := [2,\ \pi,\ 4,\ 5\,\pi,\ 8,\ 10]$$

```
> mul(x - i, i=datos);
```

$$(x-2)\,(x-\pi)\,(x-4)\,(x-5\,\pi)\,(x-8)\,(x-10)$$

Podemos usar la función **expand** para obtener el resultado del producto

```
> expand(%);
```

$$x^6 - 24\,x^5 + 196\,x^4 - 6\,x^5\,\pi + 144\,x^4\,\pi - 1176\,x^3\,\pi - 624\,x^3 + 3744\,x^2\,\pi + 5\,x^4\,\pi^2 - 120\,x^3\,\pi^2 + 980\,x^2\,\pi^2 - 3120\,x\,\pi^2 + 640\,x^2 - 3840\,x\,\pi + 3200\,\pi^2$$

Capítulo 7

Estructura de Maple y uso de Paquetes

Maple cuenta con un conjunto de funciones predefinidas, algunas de ellas son automáticamente cargadas en la memoria cada vez que iniciamos una sesión; pero, si consideramos que la versión con la cual fue creado este trabajo (Maple 8), está formada por más de tres mil funciones, las cuales ocupan poco más de 100 Mb de espacio en disco, es casi imposible (o al menos poco práctico) tener toda esta información disponible en todo momento en la memoria de la computadora.

Maple fue creado siguiendo una estructura modular que facilita el uso de todos los componentes, sin necesidad de equipos con gran capacidad de memoria. Inicialmente solo proporciona al usuario un conjunto de instrucciones básicas, pero le permite agregar funciones a la memoria para su uso durante una sesión de trabajo. Además, cuenta con un lenguaje de programación con el cual se puede incrementar su capacidad mediante la creación de rutinas propias del usuario, a esto podemos añadir la capacidad para agregar páginas de ayuda creadas también por el usuario.

7.1 Estructura interna de Maple

Maple está formado por dos componentes principales:

7.1.1 El núcleo

El núcleo (también conocido como ***kernel***) es el componente medular de Maple; entre sus funciones se encuentran el realizar los cálculos solicitados por el usuario, la revisión sintáctica de las expresiones introducidas y llevar un registro de las variables definidas, tanto por el sistema mismo como por el usuario. Este núcleo también está formado por un conjunto de instrucciones básicas, tales como: **evalf**, **diff**, **max**, **min**; además de los distintos operadores y las sentencias que forman el lenguaje de programación de este sistema.

Alrededor del núcleo se encuentran definidas un conjunto de funciones, creadas todas ellas a partir del lenguaje de programación mencionado. El núcleo también es el encargado de interpretar todas estas instrucciones que no están contenidas dentro de él, ya que uno de sus componentes es precisamente un interprete para el lenguaje de programación. Para todas estas funciones externas al núcleo, es posible acceder al código con el que están formadas (esto se tratará en las secciones posteriores); en cambio, para las funciones y sentencias pertenecientes al núcleo no es posible acceder a su código ya que el núcleo está escrito en lenguaje **"C"** y en la distribución solo se proporciona el archivo binario (el programa compilado). Este núcleo también tiene la capacidad de cargar en la memoria (de manera automática) algunas de las instrucciones no contenidas dentro de él, como se verá más adelante.

7.1.2 La biblioteca de funciones

La biblioteca está formada por todas las funciones que componen este sistema. Una parte de ellas están contenidas dentro del núcleo, mientras que el resto se encuentran dentro de archivos independientes y están escritas en el lenguaje de programación de Maple. Como se mencionó anteriormente, algunas de las funciones que no pertenecen al núcleo pueden ser cargadas automáticamente por éste, pero la mayoria requieren ser cargadas manualmente por el usuario antes de invocarlas.

Esta biblioteca contiene cuatro tipos diferentes de instrucciones, los cuales son:

Instrucciones y sentencias del núcleo

Entre los componentes del núcleo se encuentran un conjunto de funciones básicas y un conjunto de sentencias que constituyen el lenguaje de programación de Maple. Algunas de estas funciones son: **evalf**, **max**, **min**, **diff**. Mientras que entre las sentencias del lenguaje se encuentran: **if**, **for**, **break**, **goto**, **while**.

Todas estas funciones y sentencias, al estar contenidas en el núcleo, son cargadas automáticamente cuando se inicia Maple, por lo cual están disponibles a lo largo de toda la sesión de trabajo; además, por estar escritas en **"C"**, su código no es accesible al usuario. Existe una forma en la cual podemos visualizar el código de una instrucción de Maple (no de una sentencia), esto es a través de la función **eval**, por ejemplo:

```
> eval(evalf);
```

proc() **option** *builtin*, *remember*; 171 **end proc**

```
> eval(max);
```

proc() **option** *builtin*; 211 **end proc**

```
> eval(diff);
```

proc() **option** *builtin*, *remember*; 162 **end proc**

En el caso de las instrucciones del núcleo, **eval** nos da como resultado la palabra ***builtin*** y un número de identificación, lo cual indica que se trata de una instrucción perteneciente al núcleo y por lo tanto su código no puede ser visualizado.

Instrucciones no pertenecientes al núcleo, cargadas automáticamente

Existe otro conjunto de funciones que no pertenecen al núcleo, pero que son colocadas automáticamente en la memoria durante la primera referencia que se hace a ellas en una sesión de trabajo; éstas no requieren ninguna acción por parte del usuario ya que Maple conoce su localización en el sistema de archivos.

Algunas de las funciones que pertenecen a esta parte de la biblioteca son: **cos**, **sin**, **int**, **ln**, **log**, **solve**. En el caso de estas instrucciones si es posible acceder a su código. Por ejemplo:

```
> eval(cos);
```

proc(x::*algebraic*) ... **end proc**

Nótese que ***eval*** solo nos muestra la cabecera de la función, para visualizar el resto de su código es necesario indicar a Maple que nos despliegue más información. Existe una variable interna conocida como **"verbose-proc"** (consúltese la página de ayuda de **interface**), la cual indica al sistema la cantidad del código de las funciones que debe mostrarse al usuario. Esta variable puede tomar valores enteros entre cero y tres; cuando tiene asignado el valor cero muestra la menor cantidad de información al usuario, mientras que si tiene asignado tres muestra la totalidad del código que compone cada instrucción (siempre y cuando este código este disponible). Una forma de visualizar y modificar el valor que tiene asignada esta variable es a través de la instrucción **interface**:

```
> interface(verboseproc);
                                    1
```

De esta manera podemos visualizar el valor actual (el predeterminado es 1). En este caso, **eval** solo nos muestra los encabezados de las instrucciones, como vimos en la instrucción eval(cos) anterior. Para asignar un valor diferente a **verboseproc**, por ejemplo 2, procedemos de la siguiente forma:

```
> interface(verboseproc = 2);
```

Si ejecutamos nuevamente **eval(cos)**, obtenemos casi todo el código de esta función. Para obtener la totalidad debemos asignar a **verboseproc** el valor de 3.

Por razones de espacio no mostraremos aquí el código de la función **cos**; sin embargo, solo para ejemplificar mostraremos el código de la función **interface**, el cual es más pequeño.

```
> eval(interface);
```

```
proc()
local r, v, na, va;
global patchlevel;
option `Copyright (c) 1999 by the Waterloo Maple Inc. All rights reserved.`;
  if not type([args], 'interfaceargs') then error "invalid arguments" end if;
  r := NULL;
  for v in [args] do
    if type(v, name) then
      if v = 'patchlevel' then
        va := patchlevel(); if va = 'patchlevel'() then r := r, 0 else r := r, va end if
      elif v = 'autoassign' then r := r, `interface/autoassign`
      else r := r, streamcall(INTERFACE_GET(v))
      end if
    else
      na := op(1, v);
      va := op(2, v);
      if na = 'patchlevel' then error "%1 cannot be set", na
      elif na = 'autoassign' then
        if not type(va, symbol) then error "autoassign root must be a symbol" end if;
        if member(va, {'none', false}) then assign("interface/autoassign" = none)
        elif va = true then assign("interface/autoassign" = 'R')
        else assign("interface/autoassign" = va)
        end if
      else streamcall(INTERFACE_SET(na, va))
      end if
    end if
  end do;
  return r
end proc
```

Una de las ventajas de poder acceder al código es que podemos modificar las instrucciones o reutilizar partes de ellas en la creación de nuevas funciones definidas por el usuario.

El código de estas funciones también puede ser visualizado por medio de la función **print**; así, la instrucción anterior es equivalente a **print(interface)**.

La instrucción readlib

En esta parte de la biblioteca también se incluyen un conjunto de instrucciones que no pertenecen al núcleo, las cuales son cargadas automáticamente por Maple al hacer referencia a ellas por primera vez, sin necesidad de la intervención del usuario, pero que en versiones anteriores de Maple debían ser cargadas manualmente a través de la instrucción **readlib**. Por ejemplo, una de estas instrucciones es **realroot**, la cual nos da una lista de intervalos en los cuales se localizan las raíces reales de un polinomio. Veamos un ejemplo.

```
> pol := x^8 - 5*x^6 + 4*x^6 - 20*x^5 + 4*x^4 + 20*x^3;
```

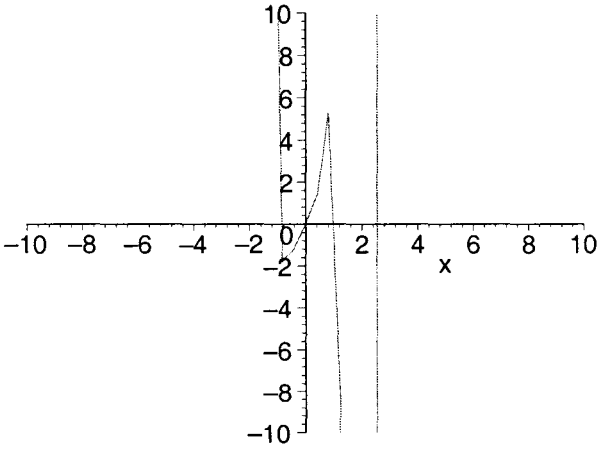
$$pol := x^8 - x^6 - 20\,x^5 + 4\,x^4 + 20\,x^3$$

```
> realroot(pol);
```

$$[[0, 0], [0, 2], [2, 4], [-4, 0]]$$

Lo que obtenemos como resultado es una lista de los intervalos en los que se encuentra cada una de las raices de **pol**. Para corroborar este resultado, desplegaremos la gráfica de este polinomio:

```
> plot(pol, x=-10..10, -10..10);
```



Como puede notarse, no es necesario indicar a Maple que cargue la instrucción, es suficiente con invocarla. De cualquier forma podemos cargarla con **readlib** de la siguiente manera:

```
> readlib(realroot);
```

proc(*poly*, *widthgoal*) ... **end proc**

Aunque dicha función es considerada obsoleta, por ser innecesaria en esta versión, ha sido mantenido por razones de compatibilidad con las versiones anteriores.

Existen muchas instrucciones de este tipo que antes debían ser cargadas mediante **readlib**, todas ellas son rutinas individuales que no pertenecen al núcleo y que son cargadas automáticamente al referenciarlas. Maple también nos permite acceder al código de éstas instrucciones, a través de las funciones **eval** y **print**.

Instrucciones contenidas en paquetes y subpaquetes

Otro de los componentes de la biblioteca de Maple consta de varios grupos de funciones especializadas conocidos con el nombre de **"paquetes"**, los cuales deben ser cargados manualmente en la memoria. Cada

uno de éstos contiene funciones útiles en la solución de problemas de un área específica. Por ejemplo, uno de los paquetes es conocido con el nombre de **plots**, el cual contiene múltiples funciones que nos permiten hacer despliegue y manipulación de estructuras gráficas. Para colocar cualquier paquete en la memoria es necesario utilizar **with** de la siguiente forma:

with(paquete);

Por ejemplo, para tener disponibles las instrucciones contenidas en **plots**, ejecutamos la siguiente instrucción:

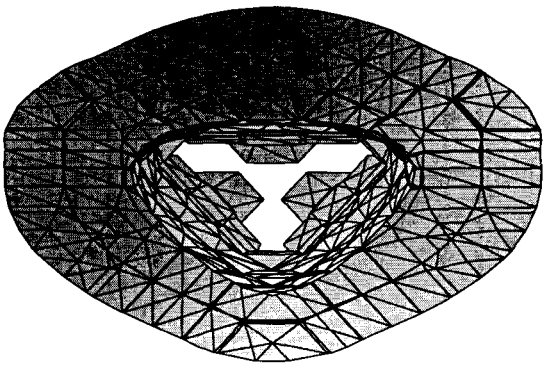
```
> with(plots);
```

```
Warning, the name changecoords has been redefined
```

[*animate*, *animate3d*, *animatecurve*, *arrow*, *changecoords*, *complexplot*, *complexplot3d*, *conformal*, *conformal3d*, *contourplot*, *contourplot3d*, *coordplot*, *coordplot3d*, *cylinderplot*, *densityplot*, *display*, *display3d*, *fieldplot*, *fieldplot3d*, *gradplot*, *gradplot3d*, *graphplot3d*, *implicitplot*, *implicitplot3d*, *inequal*, *interactive*, *listcontplot*, *listcontplot3d*, *listdensityplot*, *listplot*, *listplot3d*, *loglogplot*, *logplot*, *matrixplot*, *odeplot*, *pareto*, *plotcompare*, *pointplot*, *pointplot3d*, *polarplot*, *polygonplot*, *polygonplot3d*, *polyhedra_supported*, *polyhedraplot*, *replot*, *rootlocus*, *semilogplot*, *setoptions*, *setoptions3d*, *spacecurve*, *sparsematrixplot*, *sphereplot*, *surfdata*, *textplot*, *textplot3d*, *tubeplot*]

Nótese que la salida que obtenemos es una lista de palabras; cada una de ellas corresponde al nombre de una de las instrucciones pertenecientes al paquete que acabamos de cargar, y que ahora están disponibles para su uso. Una de las instrucciones de este paquete es **implicitplot3d**, la cual nos permite graficar una función en tres dimensiones, dando su regla de correspondencia de manera implícita. Veamos un ejemplo:

```
> implicitplot3d(x^3 + y^3 + z^3 = (x + y + z)^2, x=-2..2,
> y=-2..2, z=-2..2);
```



Una característica de la instrucción **with(plots)** es que coloca en la memoria todas las funciones que componen el paquete. Si el usuario necesita solo una de ellas, el resto no serán utilizadas pero de cualquier manera se cargan reduciendo la cantidad de memoria disponible. Sin embargo, esta misma instrucción nos proporciona varias formas de resolver este problema, solicitando que se carguen unicamente las instrucciones que el usuario necesita, como veremos a continuación

Otro de los paquetes disponibles en Maple es **student**, el cual contiene algunas instrucciones diseñadas para obtener soluciones, paso a paso, de problemas acerca de integrales, sumas y límites, entre otras cosas. Una de las funciones que contiene este paquete es **intercept**, la cual nos permite encontrar los puntos de intersección de dos curvas. Si ejecutamos la instrucción **with(student)**, esto colocará en la memoria todas las funciones de este paquete. Una forma de cargar solo una o varias funciones (pero no todas) es utilizando **with** de la siguiente manera:

with(paquete, función1, función2, función3, ...);

O bien:

with(paquete, [función1, función2, función3, ...]);

De esta forma podemos cargar, del paquete **student**, unicamente la función que necesitamos:

```
> with(student, intercept);
```

$$[intercept]$$

Ahora podemos utilizar esta función en cualquier momento, pues al cargarla permanece en la memoria durante toda la sesión de trabajo. Por ejemplo:

```
> intercept(y = x + 5, y = 8*x);
```

$$\{x = \frac{5}{7}, y = \frac{40}{7}\}$$

Otra forma de utilizar una función de un paquete, sin necesidad de cargar éste, es referenciándola como se muestra a continuación:

paquete[instrucción](argumentos);

La diferencia que existe entre esta última forma y la anterior es que, en el primer caso, la función es colocada en la memoria y permanece disponible hasta el final de la sesión, mientras que en esta última forma, la función es cargada en la memoria pero solo se puede ejecutar en el momento que es invocada, no permanece disponible en la memoria, por lo cual cualquier referencia posterior a ella no funcionará. Por ejemplo, otra de las funciones del paquete **student** es **integrand**, la cual nos da el integrando de una integral inerte. Ejecutemos la siguiente instrucción:

```
> student[integrand](Int(sin(x)^2 + x^3, x));
```

$$\sin(x)^2 + x^3$$

Ahora, trataremos de ejecutar nuevamente esta instrucción de la siguiente forma:

```
> integrand(Int(sin(x)^2 + x^3, x));
```

$$\text{integrand}(\int \sin(x)^2 + x^3\, dx)$$

En este último caso, Maple no puede ejecutar nuevamente la función, pues al utilizar la forma:

student[integrand](argumento)

ésta solamente se carga una vez para ejecutarse y no permanece en la memoria. Para cualquier referencia posterior es necesario cargar la función nuevamente; sin embargo esta forma puede ser útil para evitar colocar en la memoria paquetes completos o funciones que serán utilizados pocas veces a lo largo de la hoja de trabajo.

Algunos de los paquetes disponibles están divididos a su vez en subpaquetes. Para cargar un subpaquete contenido dentro de un paquete, podemos utilizar alguna de las siguientes formas:

- Ejecutamos primero la siguiente instrucción para tener acceso a todos los subpaquetes:

  with(paquete);

  y a continuación cargamos todas las rutinas de un subpaquete en la memoria:

  with(subpaquete);

- La siguiente instrucción nos permite tener disponible solamente el subpaquete especificado:

  with(paquete, subpaquete);

  y a continuación cargamos todas las instrucciones de este subpaquete:

  with(subpaquete);

- En la siguiente forma cargamos todos los subpaquetes:

  with(paquete);

  y después cargamos solo las funciones necesarias de un subpaquete:

  with(subpaquete, función1, función2, ...);

  Estas instrucciones son equivalentes a:

  with(paquete);

  with(subpaquete, [función1, función2, ...]);

- La siguiente forma nos permite tomar solo la instrucción especificada para ser ejecutada, pero no se almacena en la memoria:

  paquete[subpaquete, instrucción](argumentos);

- Con las siguientes instrucciones ponemos disponibles todos los paquetes pero cargamos una instrucción solo para su ejecución:

  with(paquete);

  subpaquete[instrucción](argumentos);

Uno de los paquetes de Maple que contiene varios subpaquetes es **stats**, el cual está formado por diversas instrucciones propias del área de estadística; uno de los subpaquetes que contiene es **describe**, el cual está formado por varias funciones útiles para realizar cálculos de estadística descriptiva, tales como varianza, covarianza y media, entre otros. A continuación cargaremos este subpaquete completo:

```
> with(stats, describe);
```

[*describe*]

```
> with(describe);
```

[*coefficientofvariation*, *count*, *countmissing*, *covariance*, *decile*, *geometricmean*, *harmonicmean*, *kurtosis*, *linearcorrelation*, *mean*, *meandeviation*, *median*, *mode*, *moment*, *percentile*, *quadraticmean*, *quantile*, *quartile*, *range*, *skewness*, *standarddeviation*, *sumdata*, *variance*]

Una vez hecho esto, podemos utilizar todas las funciones que contiene, por ejemplo, calculemos la varianza y la media de los siguientes datos:

```
> datos := [1, 3, 5, 9, 18, 25];
```

$$datos := [1, 3, 5, 9, 18, 25]$$

```
> variance(datos);
```

$$\frac{2669}{36}$$

```
> mean(datos);
```

$$\frac{61}{6}$$

En el caso de las funciones contenidas en paquetes y subpaquetes, la hoja de ayuda nos indica si es necesario cargarlas con **with**; además, su código puede ser visualizado por medio de **eval** o **print**, de la misma forma en que vimos anteriormente.

Instrucciones de la biblioteca compartida

Otro componente importante de Maple es la **"biblioteca compartida"** (**share library**), la cual está formada por un conjunto de rutinas, paquetes y hojas de trabajo que han sido desarrolladas por usuarios de Maple, quienes han contribuido voluntaria y gratuitamente al desarrollo de dicha biblioteca.

Esta biblioteca contiene también varios archivos creados con la finalidad de corregir errores del sistema y alguna documentación adicional. Actualmente se encuentra disponible a través de internet en la dirección:

http://www.mapleapps.com

En este sitio se pueden encontrar paquetes, hojas de trabajo y funciones que han sido aportados por usuarios de Maple, ejemplos y documentación acerca de este sistema; así como las instrucciones necesarias para acceder y usar este material.

Además, es posible hacer aportaciones nuevas y actualizar los códigos u hojas de trabajo ya enviados, por medio de correo electrónico, en la dirección: **applications@maplesoft.com**

Algunos otros sitios donde se puede encontrar información al respecto están accesibles a través de la opción **Maple on the Web**, del menú **Help**.

7.2 Paquetes disponibles en Maple

Esta versión de Maple contiene diversos paquetes de funciones especificas para realizar cálculos y resolver problemas en diversas temas tales como: curvas algebráicas, cálculo combinatorio, ajuste de curvas, ecuaciones diferenciales, formas diferenciales, cálculos financieros, geometría euclidiana, grupos, álgebra lineal, teoría de números, tensores, manejo de documentos de XML, manejo de maplets, manipulación de procesos y redes entre otros.

Para obtener más información acerca de estos paquetes, así como una lista de todos ellos, consúltese la ayuda, por medio del menú **Help**, o en la línea de comandos por medio de las siguientes instrucciones:

- Para obtener información de los paquetes existentes, ejecútese:

  help(index, library);

  o bien:

  help(packages);

- Para obtener ayuda de un paquete específico:

  ?paquete

- Para obtener información de un subpaquete:

  ?paquete, subpaquete

Las siguientes instrucciones también nos permiten acceder a las hojas de ayuda de funciones pertenecientes a un paquete o subpaquete (las cuales fueron tratadas anteriormente):

- Para obtener la página de ayuda de una función perteneciente a un paquete:

  ?paquete, función

- Para obtener la página de ayuda de una función perteneciente a un subpaquete:

  ?paquete, subpaquete, función

Capítulo 8

Ecuaciones y sistemas de ecuaciones

Maple proporciona al usuario un conjunto de funciones para manipulación y solución de ecuaciones y sistemas de ecuaciones, tanto algebraicas como ecuaciones más complicadas (por ejemplo, aquellas que contienen funciones trascendentales).

8.1 Definición de una ecuación

Una ecuación en Maple puede ser definida de la siguiente forma:

expresión1 = expresión2;

Generalmente es más práctico tener una ecuación asignada a una variable, esta asignación podemos hacerla de la siguiente manera:

nombre := ecuación;

Por ejemplo, definiremos una ecuación de segundo grado:

```
> ec1 := x^2 + 4*x + 5 = 1;
```

$$ec1 := x^2 + 4x + 5 = 1$$

8.2 Solución de una ecuación

Este sistema proporciona diversas funciones para la obtención de soluciones exactas y aproximadas de ecuaciones y sistemas de ecuaciones. A continuación haremos una revisión de estas dos formas de obtener las soluciones.

8.2.1 Obtención de soluciones exactas

Maple tiene la capacidad de resolver simbólicamente ecuaciones de segundo, tercero y cuarto grado en general, incluso casos particulares de ecuaciones polinomiales de orden mayor que cuatro y algunas ecuaciones trascendentales. Una de las instrucciones con la cual podemos resolver ecuaciones es **solve**, su sintaxis es:

solve(ecuación, var);

Donde **var** es la variable con respecto a la cual se desea resolver. Por ejemplo, definamos la siguiente ecuación:

```
> ec := 2*x = 9;
```

$$ec := 2x = 9$$

Resolveremos **ec** con respecto a la variable **x**:

```
> solve(ec, x);
```

$$\frac{9}{2}$$

De la misma forma podemos resolver ecuaciones más complicadas, por ejemplo:

```
> ec2 := 2*x^3 - 8*x^2 + 3*x + 8 = 4*x^3 - 2*x^2;
```

$$ec2 := 2x^3 - 8x^2 + 3x + 8 = 4x^3 - 2x^2$$

```
> solve(ec2, x);
```

$$\frac{(2+2I\sqrt{53})^{(1/3)}}{2} + \frac{3}{(2+2I\sqrt{53})^{(1/3)}} - 1, -\frac{(2+2I\sqrt{53})^{(1/3)}}{4} - \frac{3}{2(2+2I\sqrt{53})^{(1/3)}} - 1$$
$$+ \frac{1}{2}I\sqrt{3}\left(\frac{(2+2I\sqrt{53})^{(1/3)}}{2} - \frac{3}{(2+2I\sqrt{53})^{(1/3)}}\right), -\frac{(2+2I\sqrt{53})^{(1/3)}}{4}$$
$$- \frac{3}{2(2+2I\sqrt{53})^{(1/3)}} - 1 - \frac{1}{2}I\sqrt{3}\left(\frac{(2+2I\sqrt{53})^{(1/3)}}{2} - \frac{3}{(2+2I\sqrt{53})^{(1/3)}}\right)$$

Nótese que las soluciones no están dadas como números de punto flotante; anteriormente se había mencionado que Maple generalmente maneja las expresiones numéricas usando aritmética racional, a menos que se le dé la indicación de usar números de punto flotante. Una forma de obtener las soluciones anteriores con punto flotante es usando la función **map**, su sintaxis es:

map(función, lista de datos);

map recibe como primer argumento una función o instrucción de Maple y la aplica sobre cada uno de los elementos de la lista que aparece como segundo argumento. Podemos utilizar esto para aplicar **evalf** a cada una de las soluciones anteriores:

```
> map(evalf, [%]);
```

$$[1.174833928 - 0.2\,10^{-9}I, -3.063415448 + 0.I, -1.111418480 + 0.I]$$

También podemos utilizar **solve** para resolver ecuaciones que contienen valores indeterminados:

```
> ec3 := a*x^2 + b*x + c = d;
```

$$ec3 := ax^2 + bx + c = d$$

```
> solve(ec3, x);
```

$$\frac{-b + \sqrt{b^2 - 4ac + 4ad}}{2a}, \frac{-b - \sqrt{b^2 - 4ac + 4ad}}{2a}$$

Nótese que obtenemos en este caso las soluciones generales para una ecuación de segundo grado.

Veamos otros ejemplos:

```
> ec4 := a*x^2*b^x*y + c*y^2 = d;
```

$$ec4 := ax^2 b^x y + cy^2 = d$$

```
> solve(ec4);
```

$$\{d = ax^2 b^x y + cy^2, a = a, x = x, y = y, b = b, c = c\}$$

```
> ec5 := 4*x + 5*y = 25;
```

$$ec5 := 4\,x + 5\,y = 25$$

```
> solve(ec5, x);
```

$$-\frac{5\,y}{4} + \frac{25}{4}$$

Una forma de verificar que tales soluciones son las correctas es sustituyendo cada una de ellas en la ecuación original. Por ejemplo, obtengamos las soluciones de la siguiente ecuación:

```
> ec6 := x^2 + 2*x + 3 = 7;
```

$$ec6 := x^2 + 2\,x + 3 = 7$$

```
> solve(ec6, x);
```

$$-1 + \sqrt{5},\ -1 - \sqrt{5}$$

Podemos crear una lista con estas soluciones de la siguiente forma:

```
> lsol := [%];
```

$$lsol := [\,-1 + \sqrt{5},\ -1 - \sqrt{5}\,]$$

Para acceder a cada uno de los elementos de la lista usamos las siguientes instrucciones:

```
> lsol[1]; # la primera solución contenida en la lista
```

$$-1 + \sqrt{5}$$

```
> lsol[2]; # la segunda solución
```

$$-1 - \sqrt{5}$$

Ahora, para comprobar que tales soluciones son correctas usamos la instrucción **subs** y los elementos de las lista de la siguiente forma:

```
> subs(x = lsol[1], ec6);
```

$$(\,-1 + \sqrt{5}\,)^2 + 1 + 2\sqrt{5} = 7$$

Es necesario utilizar la función expand para que lleve a cabo la exponenciación:

```
> expand(%);
```

$$7 = 7$$

Lo mismo con la segunda solución:

```
> expand(subs(x = lsol[2], ec6));
```

$$7 = 7$$

Como podemos ver, en este caso, Maple nos devolvió soluciones exactas, que al sustituirlas nos generan números exactos. En ocasiones, al sustituir tales soluciones se obtienen resultados complicados y es necesario manipular las expresiones de alguna forma (o de varias). Por ejemplo, resolvamos **ec2**:

```
> ec2;
```

$$2\,x^3 - 8\,x^2 + 3\,x + 8 = 4\,x^3 - 2\,x^2$$

```
> lsol2 := [solve(ec2, x)]; # creamos la lista de soluciones
```

$$lsol2 := [\frac{(2+2I\sqrt{53})^{(1/3)}}{2} + \frac{3}{(2+2I\sqrt{53})^{(1/3)}} - 1, -\frac{(2+2I\sqrt{53})^{(1/3)}}{4} - \frac{3}{2(2+2I\sqrt{53})^{(1/3)}}$$
$$-1 + \frac{1}{2}I\sqrt{3}\left(\frac{(2+2I\sqrt{53})^{(1/3)}}{2} - \frac{3}{(2+2I\sqrt{53})^{(1/3)}}\right), -\frac{(2+2I\sqrt{53})^{(1/3)}}{4}$$
$$-\frac{3}{2(2+2I\sqrt{53})^{(1/3)}} - 1 - \frac{1}{2}I\sqrt{3}\left(\frac{(2+2I\sqrt{53})^{(1/3)}}{2} - \frac{3}{(2+2I\sqrt{53})^{(1/3)}}\right)]$$

Ahora, procederemos a sustituir tales soluciones:

```
> subs(x = lsol2[1], ec2);
```

$$2\%1^3 - 8\%1^2 + \frac{3(2+2I\sqrt{53})^{(1/3)}}{2} + \frac{9}{(2+2I\sqrt{53})^{(1/3)}} + 5 = 4\%1^3 - 2\%1^2$$
$$\%1 := \frac{(2+2I\sqrt{53})^{(1/3)}}{2} + \frac{3}{(2+2I\sqrt{53})^{(1/3)}} - 1$$

Aplicamos **expand**:

```
> expand(%);
```

$$-\frac{93}{2} + \frac{1}{2}I\sqrt{53} + 17(2+2I\sqrt{53})^{(1/3)} - \frac{7(2+2I\sqrt{53})^{(2/3)}}{2} + \frac{102}{(2+2I\sqrt{53})^{(1/3)}} + \frac{54}{2+2I\sqrt{53}}$$
$$-\frac{126}{(2+2I\sqrt{53})^{(2/3)}} = -47 + \sqrt{53}I + 17(2+2I\sqrt{53})^{(1/3)} - \frac{7(2+2I\sqrt{53})^{(2/3)}}{2}$$
$$+\frac{102}{(2+2I\sqrt{53})^{(1/3)}} + \frac{108}{2+2I\sqrt{53}} - \frac{126}{(2+2I\sqrt{53})^{(2/3)}}$$

Y a continución aplicamos la instrucción **simplify** para que simplifique esta expresión:

```
> simplify(%);
```

$$-\frac{-34I\sqrt{53} + 46(2+2I\sqrt{53})^{(2/3)} - 95(2+2I\sqrt{53})^{(1/3)} + 7I(2+2I\sqrt{53})^{(1/3)}\sqrt{53} + 92}{(2+2I\sqrt{53})^{(2/3)}} =$$
$$-\frac{-34I\sqrt{53} + 46(2+2I\sqrt{53})^{(2/3)} - 95(2+2I\sqrt{53})^{(1/3)} + 7I(2+2I\sqrt{53})^{(1/3)}\sqrt{53} + 92}{(2+2I\sqrt{53})^{(2/3)}}$$

Y finalmente aplicamos **evalf**, a través de **map**, para obtener un resultado:

```
> map(evalf, [%]);
```

$$[3.725716946 - 0.7316337726\,10^{-8}I = 3.725716946 - 0.7316337726\,10^{-8}I]$$

Una vez hecho esto, podemos ver que ambos elementos de la relación son iguales. Otra forma de hacer esta comprobación es utilizar **rhs** y **lhs** para extraer los elementos derecho e izquierdo respectivamente y restarlos:

```
> lhs(%%) - rhs(%%);
```

$$0$$

De esta manera, comprobamos que la primera solución es correcta. Con la segunda solución procederemos de la siguiente forma:

```
> subs(x = lsol2[2], ec2);
```

$$2\,\%2^3 - 8\,\%2^2 - \frac{3\,(2+2\,I\sqrt{53})^{(1/3)}}{4} - \frac{9}{2\,(2+2\,I\sqrt{53})^{(1/3)}} + 5 + \frac{3}{2}\,I\sqrt{3}\,\%1 =$$
$$4\,\%2^3 - 2\,\%2^2$$
$$\%1 := \frac{(2+2\,I\sqrt{53})^{(1/3)}}{2} - \frac{3}{(2+2\,I\sqrt{53})^{(1/3)}}$$
$$\%2 := -\frac{(2+2\,I\sqrt{53})^{(1/3)}}{4} - \frac{3}{2\,(2+2\,I\sqrt{53})^{(1/3)}} - 1 + \frac{1}{2}\,I\sqrt{3}\,\%1$$

```
> evalc(%); # evaluación compleja
```

$$2\,\%3^3 - 8\,\%3^2 - \frac{3}{2}\sqrt{6}\cos(\%1) + 5 - \frac{3}{2}\,\%2 = 4\,\%3^3 - 2\,\%3^2$$
$$\%1 := \frac{1}{3}\arctan(\sqrt{53})$$
$$\%2 := \sqrt{3}\sqrt{6}\sin(\%1)$$
$$\%3 := -\frac{1}{2}\sqrt{6}\cos(\%1) - 1 - \frac{1}{2}\,\%2$$

```
> lhs(%) - rhs(%);
```

$$-2\left(-\frac{1}{2}\sqrt{6}\cos(\%1) - 1 - \frac{1}{2}\sqrt{3}\sqrt{6}\sin(\%1)\right)^3 - 6\left(-\frac{1}{2}\sqrt{6}\cos(\%1) - 1 - \frac{1}{2}\sqrt{3}\sqrt{6}\sin(\%1)\right)^2$$
$$-\frac{3}{2}\sqrt{6}\cos(\%1) + 5 - \frac{3}{2}\sqrt{3}\sqrt{6}\sin(\%1)$$
$$\%1 := \frac{1}{3}\arctan(\sqrt{53})$$

```
> simplify(%);
```

$$0$$

Ahora, hagamos lo mismo con la tercera solución:

```
> evalc(subs(x = lsol2[3], ec2));
```

$$2\,\%3^3 - 8\,\%3^2 - \frac{3}{2}\sqrt{6}\cos(\%1) + 5 + \frac{3}{2}\,\%2 = 4\,\%3^3 - 2\,\%3^2$$
$$\%1 := \frac{1}{3}\arctan(\sqrt{53})$$
$$\%2 := \sqrt{3}\sqrt{6}\sin(\%1)$$
$$\%3 := -\frac{1}{2}\sqrt{6}\cos(\%1) - 1 + \frac{1}{2}\,\%2$$

```
> simplify(lhs(%) - rhs(%));
```

$$0$$

De esta forma comprobamos que tales soluciones son correctas. Otras instrucciones utilizadas para resolver ecuaciones son:

- **isolve**. Resuelve ecuaciones, desplegando solo las soluciones enteras. Por ejemplo :

```
> isolve(x^3 + x^2 - x + 8 = 9);
```

$$\{x = -1\},\ \{x = 1\}$$

En caso de que la ecuación no tenga soluciones enteras, **isolve** no muestra ningún resultado.

- **rsolve**. Resuelve ecuaciones de recurrencia.
- **msolve**. Resuelve ecuaciones en los enteros modulo m.
- **linsolve**. Resuelve ecuaciones matriciales. Esta función pertenece al paquete **linalg**, para usarla debe cargarse con **with** o bien usar la forma: **linalg[linsolve]**.
- **dsolve**. Resuelve ecuaciones diferenciales.

Consúltense las páginas de ayuda de ***rsolve*** y ***msolve*** para mayores referencias.

8.2.2 Soluciones aproximadas

Maple nos proporciona una forma de obtener soluciones de ecuaciones en forma numérica (aproximaciones de punto flotante). Esto se puede hacer a través de la instrucción **fsolve**. Por ejemplo, definamos la siguiente ecuación:

```
> eq1 := 2*x^2 + 4*x = 11;
```

$$eq1 := 2x^2 + 4x = 11$$

Utilicemos **fsolve** para obtener las soluciones:

```
> fsolve(eq1, x);
```

$$-3.549509757, 1.549509757$$

Veamos que sucede al sustituir estas soluciones en la ecuación:

```
> lista := [fsolve(eq1, x)];
```

$$lista := [-3.549509757, 1.549509757]$$

```
> subs(x = lista[1], eq1);
```

$$11.00000001 = 11$$

Ahora intentemos con la segunda solución:

```
> subs(x = lista[2], eq1);
```

$$11.00000000 = 11$$

Podemos ver que la primera solución tiene un pequeño margen de error, pues en realidad se trata de una aproximación numérica. En general, esta función no puede obtener soluciones exactas de ecuaciones, pero tiene la capacidad de calcular aproximaciones numéricas a estas soluciones. Obviamente no puede ser usada para calcular soluciones de ecuaciones en las cuales aparecen valores indeterminados.

Una de las desventajas de esta función es que no siempre encuentra soluciones complejas. Veamos un ejemplo:

```
> eq2 := 2*x^2 + 4*x = x - 9;
```

$$eq2 := 2x^2 + 4x = x - 9$$

Resolvamos primero utilizando **solve**:

```
> solve(eq2, x);
```

$$-\frac{3}{4} + \frac{3}{4} I\sqrt{7}, -\frac{3}{4} - \frac{3}{4} I\sqrt{7}$$

Nótese que ambas soluciones son complejas, intentemos usando **fsolve**:

```
> fsolve(eq2, x);
```

En este caso la función no nos da ninguna solución. Este problema se puede resolver utilizando la opción **complex** en **fsolve:**

```
> fsolve(eq2, x, complex);
```

$$-0.7500000000 - 1.984313483\,I,\ -0.7500000000 + 1.984313483\,I$$

Existe una opción conocida como **maxsols**, la cual nos permite indicar a **fsolve** cuantas soluciones queremos que nos muestre. Esta opción puede ser utilizada, por ejemplo, al resolver polinomios. Veamos un ejemplo:

```
> fsolve(23*x^4 + 10*x^3 - 28*x^2 + 17*x = 3, x, maxsols=2);
```

$$-1.556547150,\ 0.2933145038$$

Otra de las opciones que nos proporciona esta función es la poder incluir un intervalo; esto nos permite solicitar a **fsolve** unicamente aquellas soluciones que se encuentren dentro de dicho intervalo. Tal intervalo puede ser incluido en la forma **"a..b"**, **"x=a..b"** o bien **"{x=a..b, y=c..d, ...}"**. Estos intervalos se consideran cerrados y pueden incluir **infinity** o **-infinity**.

Por ejemplo, para obtener todas las soluciones positivas de la expresión anterior :

```
> fsolve(23*x^4 + 10*x^3 - 28*x^2 + 17*x = 3, x=0..infinity);
```

$$0.2933145038$$

otra forma de ejecutar esta instrucción es:

```
> fsolve(23*x^4 + 10*x^3 - 28*x^2 + 17*x = 3, x, 0..infinity);
```

$$0.2933145038$$

Otra opción de **fsolve** que puede ser útil en estos casos es **avoid**, con la cual le indicamos a esta función que calcule las soluciones pero sin tomar en cuenta el intervalo especificado por la opción **avoid**.

8.3 Interpretación de la expresión RootOf

Existen casos en los cuales Maple despliega expresiones en términos de una función conocida como **RootOf**. Tal función es utilizada para representar todas las raíces de una ecuación que depende de una variable. Aunque tiene algunos otros usos, Maple utiliza esta función para expresar las soluciones de ecuaciones y sistemas de ecuaciones polinomiales. Por ejemplo:

```
> ecn := x - y = sin(x);
```

$$ecn := x - y = \sin(x)$$

```
> solve(ecn, x);
```

$$\text{RootOf}(\_Z - y - \sin(\_Z))$$

Maple identifica de manera simbólica a la raíz de la ecuación con la expresión **RootOf()**. Veamos que sucede en el siguiente caso:

```
> pol := a*x^5 + b*x^2 - x + 1 = 0;
```

$$pol := a\,x^5 + b\,x^2 - x + 1 = 0$$

```
> solve(pol, x);
```

$$\text{RootOf}(a\,\_Z^5 + b\,\_Z^2 - \_Z + 1)$$

Ahora, sustituimos el resultado obtenido en la ecuación:

```
> subs(x = %, lhs(pol) = rhs(pol));
```

$$a\,\mathrm{RootOf}(a\_Z^5 + b\_Z^2 - \_Z + 1)^5 + b\,\mathrm{RootOf}(a\_Z^5 + b\_Z^2 - \_Z + 1)^2 - \mathrm{RootOf}(a\_Z^5 + b\_Z^2 - \_Z + 1) + 1 = 0$$

```
> simplify(%);
```

$$0 = 0$$

Como podemos ver, la expresión **RootOf()** efectivamente es manejada como una raíz de la ecuación. Es común que esta expresión aparezca en ecuaciones algebraicas que no pueden ser resueltas en forma exacta (o cuya solución es demasiado complicada). Utilizando la instrucción **allvalues**, en algunos casos se pueden obtener todos los valores posibles de una expresión en la cual se tiene involucrada **RootOf**. Por ejemplo:

```
> solve(3*x^4+5*x=2,x);
```

$$\mathrm{RootOf}(\%1,\ index = 1),\ \mathrm{RootOf}(\%1,\ index = 2),\ \mathrm{RootOf}(\%1,\ index = 3),\ \mathrm{RootOf}(\%1,\ index = 4)$$

$$\%1 := 3\_Z^4 + 5\_Z - 2$$

```
> map(allvalues, [%]);
```

$$\left[-\frac{\sqrt{6}\sqrt{\%2}}{12} + \frac{\sqrt{6}\sqrt{\dfrac{-\sqrt{\%2}\,\%1 + 32\sqrt{\%2} + 20\sqrt{6}\,\%3}{(300 + 4\sqrt{7673})^{(1/3)}\sqrt{\%2}}}}{12},\ \frac{\sqrt{6}\sqrt{\%2}}{12} + \frac{1}{12}I\sqrt{\frac{6\sqrt{\%2}\,\%1 - 192\sqrt{\%2} + 120\sqrt{6}\,\%3}{(300 + 4\sqrt{7673})^{(1/3)}\sqrt{\%2}}},\right.$$

$$-\frac{\sqrt{6}\sqrt{\%2}}{12} - \frac{\sqrt{6}\sqrt{\dfrac{-\sqrt{\%2}\,\%1 + 32\sqrt{\%2} + 20\sqrt{6}\,\%3}{(300 + 4\sqrt{7673})^{(1/3)}\sqrt{\%2}}}}{12},$$

$$\left.\frac{\sqrt{6}\sqrt{\%2}}{12} - \frac{1}{12}I\sqrt{\frac{6\sqrt{\%2}\,\%1 - 192\sqrt{\%2} + 120\sqrt{6}\,\%3}{(300 + 4\sqrt{7673})^{(1/3)}\sqrt{\%2}}}\right]$$

$$\%1 := (300 + 4\sqrt{7673})^{(2/3)}$$

$$\%2 := \frac{\%1 - 32}{(300 + 4\sqrt{7673})^{(1/3)}}$$

$$\%3 := (300 + 4\sqrt{7673})^{(1/3)}$$

La función **allvalues** procede de la siguiente forma al obtener los valores de **RootOf**:

- Todas las raíces que puedan ser obtenidas de manera exacta, son calculadas utilizando la función **solve**. Por ejemplo, las raíces de polinómios de grado menor o igual a cuatro pueden obtenerse de esta forma.

- Para aquellas raíces que no puedan determinarse de manera exacta, **allvalues** utiliza **fsolve** para calcularlas en forma numérica. Obviamente, en este caso no se admiten constantes indeterminadas en la expresión.

8.4 Interpretación gráfica de la solución de una ecuación

Gráficamente, las soluciones de una ecuación de la siguiente forma:

```
> ec := 4*x + 5*x^2 - 2 = 0;
```

$$ec := 4x + 5x^2 - 2 = 0$$

corresponden a los valores de **x** en los cuales la ecuación se hace cero (es decir, corresponden a ***"los ceros"*** de la ecuación). Primero, obtendremos dichas soluciones:

```
> solve(ec, x);
```

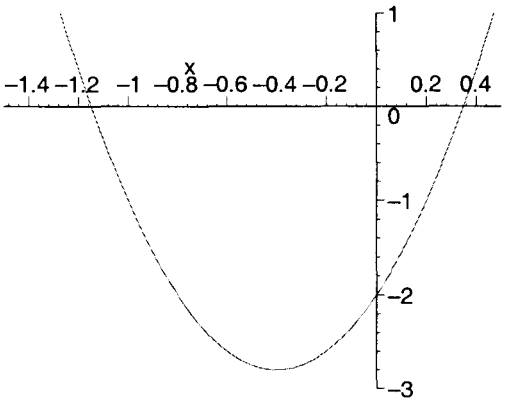
$$-\frac{2}{5} + \frac{\sqrt{14}}{5}, -\frac{2}{5} - \frac{\sqrt{14}}{5}$$

```
> evalf(%);
```

$$0.3483314774, -1.148331477$$

Ahora, graficamos la ecuación:

```
> plot(4*x + 5*x^2 - 2, x=-1.5..0.5, -3..1);
```



Podemos ver que en la gráfica aparecen aproximadas las soluciones. De la misma forma, para la expresión:

```
> ec2 := x^2 + 9 = x - x^3;
```

$$ec2 := x^2 + 9 = x - x^3$$

```
> solve(ec2, x);
```

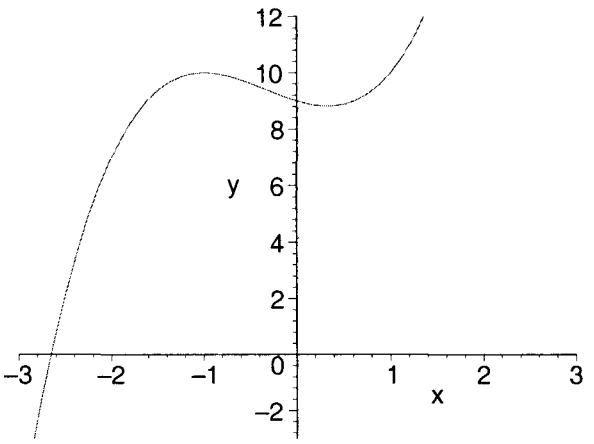
$$-\frac{(127+3\sqrt{1785})^{(1/3)}}{3} - \frac{4}{3(127+3\sqrt{1785})^{(1/3)}} - \frac{1}{3}, \frac{(127+3\sqrt{1785})^{(1/3)}}{6}$$
$$+ \frac{2}{3(127+3\sqrt{1785})^{(1/3)}} - \frac{1}{3}$$
$$+ \frac{1}{2} I\sqrt{3}\left(-\frac{(127+3\sqrt{1785})^{(1/3)}}{3} + \frac{4}{3(127+3\sqrt{1785})^{(1/3)}}\right), \frac{(127+3\sqrt{1785})^{(1/3)}}{6}$$
$$+ \frac{2}{3(127+3\sqrt{1785})^{(1/3)}} - \frac{1}{3}$$
$$- \frac{1}{2} I\sqrt{3}\left(-\frac{(127+3\sqrt{1785})^{(1/3)}}{3} + \frac{4}{3(127+3\sqrt{1785})^{(1/3)}}\right)$$

```
> evalf(%);
```

$$-2.654249157, 0.8271245787 - 1.645191285\,I, 0.8271245787 + 1.645191285\,I$$

Desplegamos la gráfica:

```
> plot(x^2 + 9 - x + x^3, x=-3..3, y=-3..12);
```



En este caso, podemos ver que solo se tiene una solución real, pues la gráfica solo atraviesa una vez el eje (solo existe un cero), mientras que existen también dos raíces complejas.

8.5 Sistemas de ecuaciones

Para resolver sistemas de ecuaciones, utilizamos las mismas funciones **solve** y **fsolve**, pero aplicadas a conjuntos de ecuaciones; con esto obtenemos una solución que satisface todas las ecuaciones del sistema de manera simultanea. Para definir un sistema de ecuaciones podemos proceder de la siguiente forma:

```
> ec1 := x + y + z = 9;
```

$$ec1 := x + y + z = 9$$

```
> ec2 := 4*y + 3*z = 6;
```

$$ec2 := 4\,y + 3\,z = 6$$

```
> ec3 := x - y + 2*z = 4;
```

$$ec3 := x - y + 2\,z = 4$$

Y a continuación colocamos todas las ecuaciones en forma de un conjunto para obtener su solución, por ejemplo a través de **solve**:

solve({ec1, ec2, ec3, ...}, {x1, x2, x3, ...});

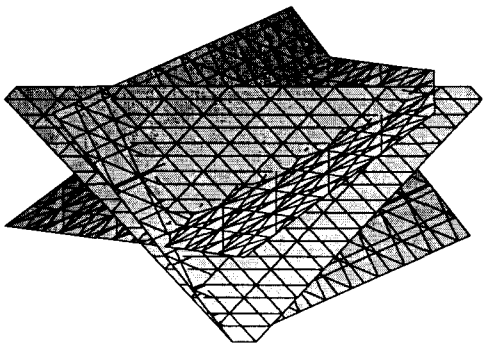
Donde **"ec1, ec2, ec3, ..."** son la ecuaciones que pertenecen al sistema y **"x1, x2, x3, ..."** son las incógnitas con respecto a las que se desea resolver. De hecho, éstas últimas pueden omitirse y entonces Maple tratará de obtener una solución para cada una de las incógnitas presentes. Así, podemos resolver el sistema formado por **"ec1"**, **"ec2"** y **"ec3"** de la siguiente forma:

```
> solve({ec1, ec2, ec3}, {x, y, z});
```

$$\{y = \frac{21}{10}, x = \frac{77}{10}, z = \frac{-4}{5}\}$$

La interpretación gráfica de esta solución es el punto en el cual se intersectan los planos definidos por cada una de las ecuaciones. Podemos utilizar la instrucción **implicitplot3d** del paquete **plots** para desplegar su gráfica:

```
> plots[implicitplot3d]({x + y + z - 9, 2*z + 4*y + z - 6,
> x - y + 2*z - 4}, x=-10..10, y=-10..10, z=-10..10);
```



Para aquellos sistemas en los cuales Maple no puede determinar soluciones exactas, también es posible usar **fsolve** para calcular una aproximación numérica a las soluciones. Por ejemplo:

```
> eq1 := x + y + z = cos(x);
```

$$eq1 := x + y + z = \cos(x)$$

```
> eq2 := 5*x + 4*y + z = 3*x + y;
```

$$eq2 := 5\,x + 4\,y + z = 3\,x + y$$

```
> eq3 := 3*x - y - z = 9;
```

$$eq3 := 3\,x - y - z = 9$$

```
> solve({eq1, eq2, eq3}, {x, y, z});
```

$$\{y = -\frac{5}{2}\,\mathrm{RootOf}(4\_Z - 9 - \cos(\_Z)) + \frac{9}{2},\ x = \mathrm{RootOf}(4\_Z - 9 - \cos(\_Z)),$$
$$z = \frac{11}{2}\,\mathrm{RootOf}(4\_Z - 9 - \cos(\_Z)) - \frac{27}{2}\}$$

En este caso, solve no puede obtener una solución exacta, pero **fsolve** sí puede calcular una aproximación:

```
> fsolve({eq1, eq2, eq3}, {x, y, z});
```

$$\{x = 2.119586203,\ y = -0.7989655084,\ z = -1.842275882\}$$

Nuevamente recuérdese que tales soluciones no son exactas, en realidad se trata de aproximaciones que pueden presentar un cierto grado de error.

8.6 Solución de sistemas de ecuaciones lineales con matrices

En el caso de los sistemas de ecuaciones lineales, estas pueden ser resueltas utilizando operaciones matriciales. El tema de matrices se tratará posteriormente, pero haremos una pequeña introducción para poder resolver sistemas de ecuaciones por este método.

8.6.1 Matrices

Definicion de matrices

Una forma de manejar matrices en Maple es por medio del paquete **LinearAlgebra**, el cual contiene un conjunto de funciones para manejo de vectores y matrices, así como para realizar diversos cálculos de Álgebra Lineal. Este paquete puede cargarse en la memoria por medio de la instrucción **with**.

Por otro lado, una manera de definir una matriz es por medio de la instruccion **Matrix** (la cual no pertenece a ningún paquete). Ésta nos proporciona varias formas de crear matrices, una de ellas es:

Matrix(n, m, [[a11, a12, a13, ...a1m], [a21, ...,a2m], ...,[an1, ..., anm]]);

Donde **n** y **m** indican el número de renglones y columnas de la matriz, respectivamente, mientras que "**[a11, a12, a13,...a1m], [a21,...,a2m], ...,[an1,..., anm]**" son los elementos de cada uno de los renglones (cada uno de éstos debe colocarse como una lista). En caso de que los elementos no sean especificados, la matriz es creada con todos sus elementos iguales a cero. Por ejemplo, la instrucción:

```
> A := Matrix(2, 2, [[1, 2], [3, 4]]);
```

$$A := \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

define una matriz de dos renglones y dos columnas, cuyos elementos son 1, 2, 3, 4. Veamos otro ejemplo:

```
> B := Matrix(1, 2, [3, 4]);
```

$$B := \begin{bmatrix} 3 & 4 \end{bmatrix}$$

Los elementos de una matriz pueden estar dados no solo por números, también pueden incluirse expresiones algebraicas, aritméticas, constantes y variables simbólicas. Por ejemplo:

```
> C := Matrix(4, 4,
> [[1, 2, 3, 4], [sin(x), 4, 9, Pi^2], [8, sqrt(2), 3, 1], [1, 2, 3,
> 4]]);
```

$$C := \begin{bmatrix} 1 & 2 & 3 & 4 \\ \sin(x) & 4 & 9 & \pi^2 \\ 8 & \sqrt{2} & 3 & 1 \\ 1 & 2 & 3 & 4 \end{bmatrix}$$

Para acceder a un elemento de una matriz, utilizamos la siguiente expresión:

M[a, b]

Donde **M** es el nombre de la matriz, **a** es el renglón donde se encuentra el elemento al cual se desea acceder y **b** es la columna del mismo. Por ejemplo para acceder al elemento del segundo renglón y la segunda columna de la matriz **C** utilizamos:

```
> C[2,2];
```

$$4$$

De la misma forma podemos, por ejemplo, modificar los elementos de la matriz **C**:

```
> C[1,1]:=9;
```

$$C_{1,1} := 9$$

```
> C[1,3]:=.8764532;
```

$$C_{1,3} := 0.8764532$$

Evaluación, suma y multiplicación de matrices

Evaluación

En el caso de las matrices creadas con la función **Matrix**, para desplegar sus elementos es suficiente teclear su nombre. Definamos la matriz **"E"**:

```
> E := Matrix(4, 4, [[3, 4, 6, 4], [8, cos(x), .867, Pi^2],
> [5, exp(2), 3, 1], [1, 2, 3, 4]]);
```

$$E := \begin{bmatrix} 3 & 4 & 6 & 4 \\ 8 & \cos(x) & 0.867 & \pi^2 \\ 5 & e^2 & 3 & 1 \\ 1 & 2 & 3 & 4 \end{bmatrix}$$

```
> E;
```

$$\begin{bmatrix} 3 & 4 & 6 & 4 \\ 8 & \cos(x) & 0.867 & \pi^2 \\ 5 & e^2 & 3 & 1 \\ 1 & 2 & 3 & 4 \end{bmatrix}$$

Como podemos ver, al colocar su nombre, Maple nos da automáticamente los elementos.

Suma

La suma de matrices es la matriz formada con la suma de los elementos. Recuérdese que para llevar a cabo esta operación las matrices deben ser del mismo orden. Por ejemplo:

```
> M := Matrix(2, 2, [[3, 4], [78, 2]]);
```

$$M := \begin{bmatrix} 3 & 4 \\ 78 & 2 \end{bmatrix}$$

```
> N := Matrix(2, 2, [[43, 9], [1, 8]]);
```

$$N := \begin{bmatrix} 43 & 9 \\ 1 & 8 \end{bmatrix}$$

```
> P := Matrix(1, 2, [8, 2]);
```

$$P := \begin{bmatrix} 8 & 2 \end{bmatrix}$$

Para obtener la suma podemos utilizar el operador aritmético '+' de la siguiente forma:

```
> M + N;
```

$$\begin{bmatrix} 46 & 13 \\ 79 & 10 \end{bmatrix}$$

Cuando se llevan a cabo operaciones con matrices deben tenerse en consideración las dimensiones éstas. Por ejemplo, a continuación intentemos sumar **M + P** que tienen orden diferente (2x2 y 2x1):

```
> M + P;

Error, (in rtable/Sum) invalid arguments
```

Maple automáticamente verifica las dimensiones antes de ejecutar la operación.

Para calcular este tipo de sumas también puede usarse la función **evalm**, en la forma: **evalm(M + N)**; sin embargo es más sencillo usar directamente el operador aritmético.

Multiplicación

La multiplicación de matrices (por ejemplo **A X**) no se efectúa con * (reservado para multiplicación de números y expresiones), tampoco puede calcularse por medio de **evalm**:

```
> M := Matrix(2, 2, [[3, 4], [78, 2]]);
```

$$M := \begin{bmatrix} 3 & 4 \\ 78 & 2 \end{bmatrix}$$

```
> N := Matrix(2, 2, [[43, 9], [1, 8]]);
```

$$N := \begin{bmatrix} 43 & 9 \\ 1 & 8 \end{bmatrix}$$

```
> P := Matrix(1, 2, [8, 2]);
```

$$P := \begin{bmatrix} 8 & 2 \end{bmatrix}$$

```
> evalm(M*P);
Error, (in rtable/Product) invalid arguments
```

Existen dos formas de multiplicar matrices, utilizando el operador **"."**, o también utilizando la instrucción **MatrixMatrixMultiply** del paquete **LinearAlgebra**:

```
> M.N;
```

$$\begin{bmatrix} 133 & 59 \\ 3356 & 718 \end{bmatrix}$$

```
> LinearAlgebra[MatrixMatrixMultiply](M, N);
```

$$\begin{bmatrix} 133 & 59 \\ 3356 & 718 \end{bmatrix}$$

Recuérdese que la multiplicación de matrices no es conmutativa, es decir, en general se cumple que **M x N** es diferente de **N x M**. Ejemplo:

```
> M.N;
```

$$\begin{bmatrix} 133 & 59 \\ 3356 & 718 \end{bmatrix}$$

```
> N.M;
```

$$\begin{bmatrix} 831 & 190 \\ 627 & 20 \end{bmatrix}$$

Matrices Inversa e Identidad

Maple tiene la capacidad de calcular matrices inversas y de manejar matrices identidad. Dada una matriz ***"Q"***, podemos calcular su inversa por medio de la instrucción **MatrixInverse** (perteneciente al paquete **LinearAlgebra**), de la siguiente forma:

```
> Q := Matrix(2, 2, [[9, 2], [5, 7]]);
```

$$Q := \begin{bmatrix} 9 & 2 \\ 5 & 7 \end{bmatrix}$$

```
> Qinv := LinearAlgebra[MatrixInverse](Q);
```

$$Qinv := \begin{bmatrix} \frac{7}{53} & \frac{-2}{53} \\ \frac{-5}{53} & \frac{9}{53} \end{bmatrix}$$

Esta matriz inversa se representa como: $Q^{(-1)}$ (es decir, la matriz inversa de **Q**), tal que:

$$Q^{(-1)}\,Q = Id$$

donde ***Id*** es la ***"matriz identidad"***, la cual es de la misma dimensión que **Q** en este caso, y contiene únicamente el valor **"1"** en su diagonal y ceros en las otras posiciones. Además, esta matriz tiene la propiedad:

$$Q\,Id = Q$$
$$Id\,Q = Q$$

Esto para cualquier matriz **Q** (recuérdese que son importantes las dimensiones).

Verifiquemos si **Qinv** realmente es la inversa de **Q**:

```
> Qinv.Q;
```

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

```
> Q.Qinv;
```

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Una forma de generar una matriz identidad es por medio de la función **IdentityMatrix de LinearAlgebra**. Por ejemplo, crearemos una matriz identidad de 2 x 2:

```
> Id := LinearAlgebra[IdentityMatrix](2);
```

$$Id := \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Multipliquemos **Id** por **Q**:

```
> Q;
```

$$\begin{bmatrix} 9 & 2 \\ 5 & 7 \end{bmatrix}$$

```
> Id.Q;
```

$$\begin{bmatrix} 9 & 2 \\ 5 & 7 \end{bmatrix}$$

```
> Q.Id;
```

$$\begin{bmatrix} 9 & 2 \\ 5 & 7 \end{bmatrix}$$

8.6.2 Solución de un sistema de dos ecuaciones lineales con dos incógnitas

Dado un sistema de ecuaciones de la forma:

$$a_1\, x + b_1\, y = c_1$$
$$a_2\, x + b_2\, y = c_2$$

donde x, y son las incógnitas, a_1, b_1, a_2, b_2 y c_1, c_2 son los términos independientes. Cada una de las dos ecuaciones puede interpretarse como una expresión o función cuya gráfica es una línea recta (despejando y en terminos de x). Por esta razón originalmente se conocen con el nombre de ***"ecuaciones lineales"***.

La solucion del sistema formado por estas ecuaciones se puede obtener aplicando **solve** (o **fsolve**) al conjunto de ecuaciones, en la forma:

```
solve({ec1, ec2}, {x, y});
```

La interpretacián gráfica de la solución es el punto de intersección de las rectas:

$y := -\frac{a_1\, x}{b_1} + \frac{c_1}{b_1}$, $y := -\frac{a_2\, x}{b_2} + \frac{c_2}{b_2}$

Otra forma de resolver este sistema (sin utilizar **solve** o **fsolve**) es por medio de matrices.

El sistema general de dos ecuaciones lineales inhomogéneas, planteado anteriormente, es equivalente a la ecuación matricial:

$$\mathbf{A\ X = Y}$$

Donde **A** es la matriz formada por los coeficientes de las ecuaciones, **X** es la matrix formada por las incógnitas y **Y** es la matriz formada por los términos independientes. Veamos un ejemplo particular, para el siguiente sistema de ecuaciones lineales:

$3\,x + 2\,y = 1$

$3\,x - 2\,y = 2$

Su ecuación matricial está dada por: $A\,X = Y$, donde:

$$A = \begin{bmatrix} 3 & 2 \\ 3 & -2 \end{bmatrix}$$

$$X = \begin{bmatrix} x \\ y \end{bmatrix}$$

$$Y = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

Para resolver esta ecuación matricial necesitamos encontrar la inversa de **A**, una vez hecho esto multiplicamos cada lado de la ecuación matricial por esta inversa y obtenemos:

$$A^{(-1)}\, A\, X = A^{(-1)}\, Y$$
$$Id\, X = A^{(-1)}\, Y$$

De donde:

$$X = A^{(-1)}\, Y$$

Con lo cual obtenemos la solución. Antes de proceder nos aseguraremos de cargar el paquete **LinearAlgebra**:

```
> with(LinearAlgebra):
```

Ahora definimos **A**, **X**, **Y** y calculamos la inversa de **A**:

```
> A := Matrix(2, 2, [[3, 2], [3, -2]]);
```

$$A := \begin{bmatrix} 3 & 2 \\ 3 & -2 \end{bmatrix}$$

```
> X := Matrix(2, 1, [[x],[y]]);
```

$$X := \begin{bmatrix} x \\ y \end{bmatrix}$$

```
> Y := Matrix(2, 1, [[1], [2]]);
```

$$Y := \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

```
> Inv_A := MatrixInverse(A);
```

$$Inv\_A := \begin{bmatrix} \frac{1}{6} & \frac{1}{6} \\ \frac{1}{4} & \frac{-1}{4} \end{bmatrix}$$

Después definimos la ecuación matricial:

```
> ecmat := A.X = Y;
```

$$ecmat := \begin{bmatrix} 3x + 2y \\ 3x - 2y \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

Y finalmente obtenemos la solución multiplicando ambos lados por la matriz inversa de **A**:

```
> sol := Inv_A.lhs(ecmat) = Inv_A.rhs(ecmat);
```

$$sol := \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \frac{1}{2} \\ \frac{-1}{4} \end{bmatrix}$$

Por lo tanto la solución es: $x = \frac{1}{2}$, $y = -\frac{1}{4}$. Comparemos con el resultado generado por **solve:**

```
> solve({3*x + 2*y = 1, 3*x - 2*y = 2}, {x, y});
```

$$\{y = \frac{-1}{4}, x = \frac{1}{2}\}$$

Esta solución fue calculada llevando a cabo cada uno de las operaciones matriciales; sin embargo, también puede ser obtenida por medio de la instrucción **LinearSolve** del paquete **Linear Algebra**, la cual resuelve ecuaciones matriciales de la forma **A.x = Y**. La sintaxis para realizar esta operación es:

LinearSolve(A, Y);

como se muestra a continuación:

```
> LinearSolve(A, Y);
```

$$\begin{bmatrix} \frac{1}{2} \\ \frac{-1}{4} \end{bmatrix}$$

En el caso de estas ecuaciones, la solución se puede interpretar gráficamente como el punto en el cual se intersectan las rectas definidas por cada una de ellas. Para poder visualizar esto primero despejaremos ambas ecuaciones en términos de **y**:

```
> ec1 := solve(3*x + 2*y = 1, y);
```

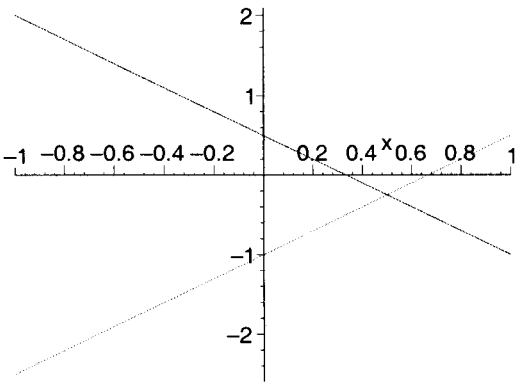
$$ec1 := -\frac{3x}{2} + \frac{1}{2}$$

```
> ec2 := solve(3*x - 2*y = 2, y);
```

$$ec2 := \frac{3x}{2} - 1$$

Ahora deplegaremos las gráficas para poder apreciar el punto de intersección:

```
> plot({ec1, ec2}, x=-1..1);
```



8.6.3 Sistemas de tres ecuaciones lineales con tres incógnitas

Las ecuaciones de la forma:

$$a_1\, x + b_1\, y + c_1\, z = d_1$$

son llamadas ecuaciones lineales. Nótese que las tres incógnitas **x**, **y**, **z** aparecen elevadas a la potencia **"1"** (es decir, no aparecen términos como x^2, $\sqrt{y}$, $\sin(z)$, ni productos como $x\ y$, $z\ x$, etc. En el caso anterior, teníamos:

$$a_1\, x + b_1\, y = c_1$$

Despejando **y** obteníamos una expresión (o función) cuya gráfica era una línea recta, de ahí el nombre ***"lineal"***. En el caso de tres incógnitas, si despejamos una de ellas (por ejemplo **z**), tendriamos:

$$z = -\frac{a_1\, x}{c_1} - \frac{b_1\, y}{c_1} - \frac{d_1}{c_1}$$

cuya gráfica es un plano en el espacio. Aunque el plano es una generalización de una línea recta, a estas ecuaciones las seguimos llamando ***"lineales"*** y no ***"planares"***.

La forma de resolver sistemas de tres ecuaciones con tres incógnitas es equivalente a la solución de sistemas de dos ecuaciones con dos incógnitas.

Veamos un ejemplo con un sistema de tres ecuaciones lineales con tres incógnitas:

```
> ec1 := 3*x + 2*y - z = 0;
```

$$ec1 := 3x + 2y - z = 0$$

```
> ec2 := -2*x - 3*y + z/2 = -1;
```

$$ec2 := -2x - 3y + \frac{z}{2} = -1$$

```
> ec3 := -x - y - z = 1;
```

$$ec3 := -x - y - z = 1$$

Podemos utilizar la instrucción **coeffs** para extraer los coeficientes de la parte izquierda de cada relación. Por ejemplo:

```
> coeffs(lhs(ec1));
```

$$3, 2, -1$$

A continuación definiremos los elementos de la ecuación matricial y resolveremos ésta por medio de la instrucción **LinearSolve**:

```
> A := Matrix(3, 3,
> [[coeffs(lhs(ec1))], [coeffs(lhs(ec2))], [coeffs(lhs(ec3))]]);
```

$$A := \begin{bmatrix} 3 & 2 & -1 \\ -2 & -3 & \frac{1}{2} \\ -1 & -1 & -1 \end{bmatrix}$$

```
> Y := Matrix(3, 1, [[rhs(ec1)], [rhs(ec2)], [rhs(ec3)]]);
```

$$Y := \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix}$$

Ahora obtenemos la solución:

```
> LinearSolve(A, Y);
```

$$\begin{bmatrix} \frac{-10}{13} \\ \frac{9}{13} \\ \frac{-12}{13} \end{bmatrix}$$

Comparemos con la solución obtenida por **solve**:

```
> solve({ec1, ec2, ec3}, {x, y, z});
```

$$\{z = \frac{-12}{13}, y = \frac{9}{13}, x = \frac{-10}{13}\}$$

Para trazar la gráfica de la expresión que define el plano obtenido al despejar **z**, usamos la instrucción para gráficas tridimensionales **plot3d**, cuya sintaxis es:

plot3d(expresion, x=rango, y=rango);

Primero despejamos **z**, por ejemplo en la ecuación 1:

```
> plano_1 := solve(ec1, z);
```

$$plano\_1 := 3x + 2y$$

Y a continuación graficamos:

```
> plot3d(plano_1, x=-5..5, y=-5..5);
```

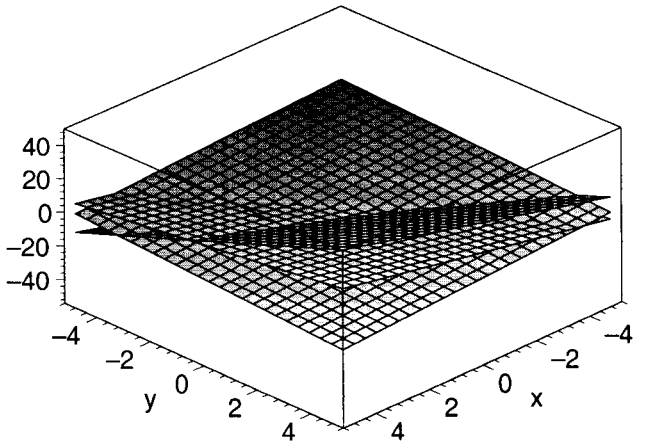
Lo mismo para los otros planos:

```
> plano_2 := solve(ec2, z); plano_3 := solve(ec3, z);
```

$$plano\_2 := 4x + 6y - 2$$

$$plano\_3 := -x - y - 1$$

```
> plot3d({plano_1, plano_2, plano_3}, x=-5..5, y=-5..5,
> axes=BOXED);
```



En este caso, la solución se puede interpretar como el punto en el cual se intersectan los tres planos en el espacio.

8.7 Gráficas de sistemas de ecuaciones

8.7.1 Dos ecuaciones lineales con dos incógnitas

Considérese el siguiente sistema de dos ecuaciones lineales:

```
> sist_lin := 3*x - y = 4, -7*x + 8*y = 1;
```

$$sist\_lin := 3x - y = 4, -7x + 8y = 1$$

Podemos calcular la solución a través de **solve**:

```
> solve({sist_lin}, {x, y});
```

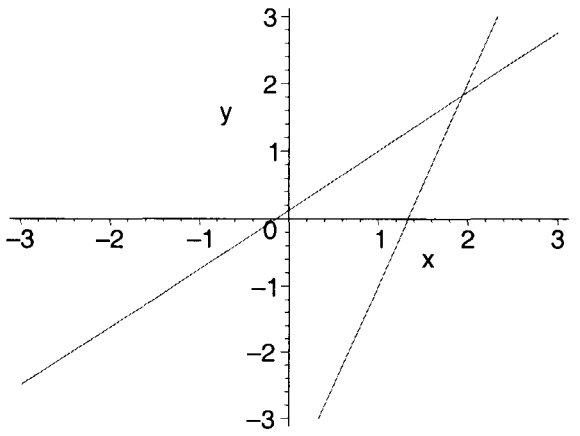
$$\{x = \frac{33}{17}, y = \frac{31}{17}\}$$

La interpretación gráfica de dicha solución es el punto común de las ecuaciones:

$$\begin{aligned} 3x - y &= 4 \\ -7x + 8y &= 1 \end{aligned}$$

Esto puede visualizarse graficando ambas funciones y viendo su punto de intersección en la gráfica. Podemos usar la función **implicitplot** del paquete **plots**.

```
> plots[implicitplot]({3*x - y = 4, -7*x + 8*y = 1}, x=-3..3,
> y=-3..3);
```



Este razonamiento se aplica también a ecuaciones no-lineales, por ejemplo:

```
> sist := 2*x^2 + 2*y - 1 = x, y^2 - 2*x + 1 = y;
```

$$sist := 2x^2 + 2y - 1 = x, y^2 - 2x + 1 = y$$

Al aplicar **solve**, resultan dos soluciones muy complicadas:

```
> solve({sist}, {x, y});
```

$$\{y = -1, x = \frac{3}{2}\}, \{x = \frac{1}{2}\%1^2 + \frac{1}{2} - \frac{1}{2}\%1, y = \%1\}$$
$$\%1 := \text{RootOf}(\_Z^3 - 3\_Z^2 + 5\_Z - 2, label = \_L10)$$

```
> allvalues(%[2]);
```

$$\left\{x = \frac{(-\frac{(108+12\sqrt{177})^{(1/3)}}{6} + \frac{4}{(108+12\sqrt{177})^{(1/3)}} + 1)^2}{2} + \frac{(108+12\sqrt{177})^{(1/3)}}{12} - \frac{2}{(108+12\sqrt{177})^{(1/3)}}, y = -\frac{(108+12\sqrt{177})^{(1/3)}}{6} + \frac{4}{(108+12\sqrt{177})^{(1/3)}} + 1\right\}, \left\{x = \frac{(\frac{(108+12\sqrt{177})^{(1/3)}}{12} - \frac{2}{(108+12\sqrt{177})^{(1/3)}} + 1 + \frac{1}{2}I\sqrt{3}\%1)^2}{2} - \frac{(108+12\sqrt{177})^{(1/3)}}{24} + \frac{1}{(108+12\sqrt{177})^{(1/3)}} - \frac{1}{4}I\sqrt{3}\%1, y = \frac{(108+12\sqrt{177})^{(1/3)}}{12} - \frac{2}{(108+12\sqrt{177})^{(1/3)}} + 1 + \frac{1}{2}I\sqrt{3}\%1\right\}, \left\{x = \frac{(\frac{(108+12\sqrt{177})^{(1/3)}}{12} - \frac{2}{(108+12\sqrt{177})^{(1/3)}} + 1 - \frac{1}{2}I\sqrt{3}\%1)^2}{2} - \frac{(108+12\sqrt{177})^{(1/3)}}{24} + \frac{1}{(108+12\sqrt{177})^{(1/3)}} + \frac{1}{4}I\sqrt{3}\%1, y = \frac{(108+12\sqrt{177})^{(1/3)}}{12} - \frac{2}{(108+12\sqrt{177})^{(1/3)}} + 1 - \frac{1}{2}I\sqrt{3}\%1\right\}$$

$$\%1 := -\frac{(108+12\sqrt{177})^{(1/3)}}{6} - \frac{4}{(108+12\sqrt{177})^{(1/3)}}$$

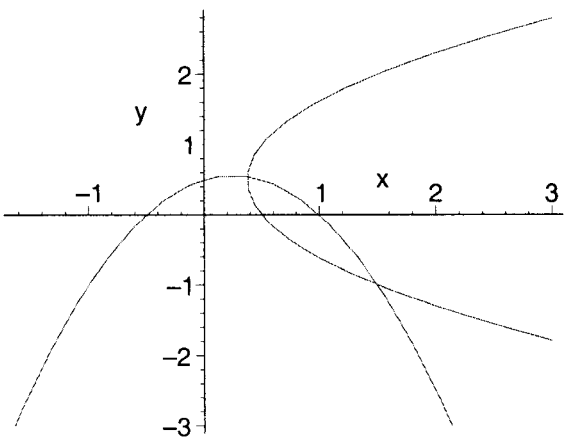
Una solución verdaderamente aparatosa. Por otra parte, **fsolve** solo muestra una solución:

```
> fsolve({sist}, {x, y});
```

$$\{x = 1.500000000, y = -1.000000000\}$$

Veamos el aspecto de las dos soluciones graficamente:

```
> plots[implicitplot]({sist[1], sist[2]}, x=-3..3, y=-3..3);
```



8.7.2 Dos ecuaciones con tres incógnitas

Veamos el siguiente ejemplo con dos ecuaciones lineales:

```
> sist_lin := 3*x - y + 2*z = 4, -7*x + 8*y - z/2 = 1;
```

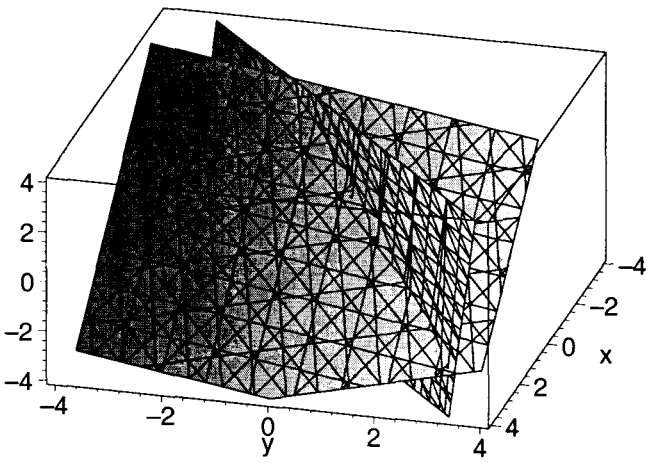
$$sist\_lin := 3x - y + 2z = 4, -7x + 8y - \frac{z}{2} = 1$$

Aplicaremos **solve** y desplegaremos la gráfica del sistema.

```
> solve({sist_lin}, {x, y, z});
```

$$\{z = -\frac{34x}{31} + \frac{66}{31}, x = x, y = \frac{25x}{31} + \frac{8}{31}\}$$

```
> plots[implicitplot3d]({3*x - y + 2*z = 4, -7*x + 8*y - z/2 = 1},
> x=-4..4, y=-4..4, z=-4..4, axes=BOXED, orientation=[15, 50]);
```



Podemos comprobar que existe un número infinito de soluciones (esto se debe a que tenemos más variables que ecuaciones). Por otro lado, en la gráfica podemos ver que las funciones implícitas son planos, por lo que un número infinito de soluciones debe corresponder al hecho de que dos planos se intersectan a lo largo de una recta (dado que en este caso no son paralelos).

Intentaremos conseguir una mejor visualización con la superposición de varias estructuras gráficas, por medio de la función **display3d**.

Nos conviene que los planos definidos por las ecuaciones aparezcan sin ***"parches"***, en color sólido (por ejemplo amarillo o verde). Asimismo, la recta en la cual se intersectan los planos puede ir superpuesta a estos. La ecuación que describe a dicha recta la obtuvimos al aplicar **solve**, y viene dada en forma paramétrica con parámetro **x**.

Primero calculamos la gráfica de la curva: $[x, \frac{25x}{31} + \frac{8}{31}, -\frac{34x}{31} + \frac{66}{31}]$, y la asignamos a una variable. Utilizaremos para ello la función **spacecurve** del paquete **plots**, la cual nos permite graficar una curva en el espacio:

```
> recta_int := plots[spacecurve]([x, (25/31)*x + 8/31, -(34/31)*x +
> 66/31], x=-3..-2, axes=boxed, thickness=2, color=red):
```

A continuación, calculamos la gráfica de: $\frac{4-3x+y}{2}$.

Para ello podemos usar **plot3d**, ya que se trata de una función dada en forma explícita:

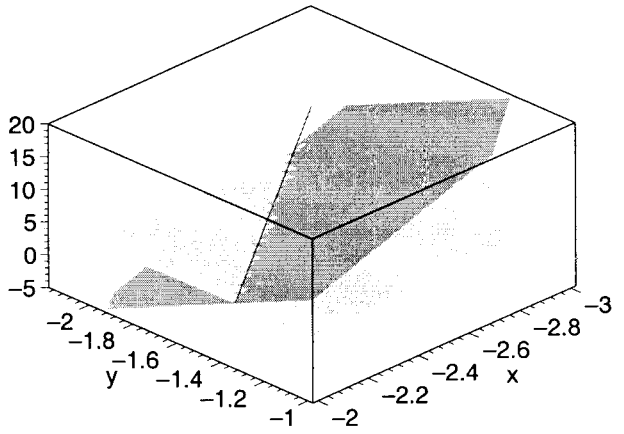
```
> plano_1 := plot3d((4 - 3*x + y)/2, x=-3..-2, y=-2..-1,
> view=-5..20, style=patchnogrid, axes=boxed, color=yellow):
```

Ahora, calculamos la gráfica de: $2(-7x + 8y - 1)$, también con **plot3d**:

```
> plano_2 := plot3d(2*(-7*x + 8*y - 1), x=-3..-2, y=-2..-1,
> view=-5..20, style=patchnogrid, axes=boxed, color=cyan):
```

Y a continuación desplegamos estas gráficas superpuestas:

```
> plots[display3d]({recta_int, plano_1, plano_2});
```



Con lo cual obtenemos una mejor vista de la intersección.

8.7.3 Tres ecuaciones con tres incógnitas

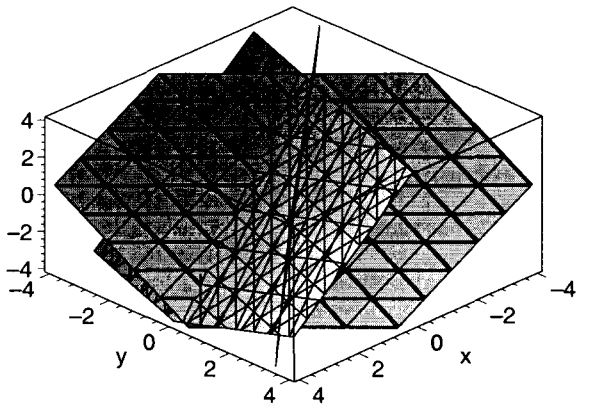
Si al sistema de la sección anterior le agregamos una ecuación más (que no sea multiplo de alguna de las existentes, es decir, debe haber independencia lineal), obtenemos una solución única del sistema:

```
> solve({3*x - y + 2*z = 4, -7*x + 8*y - z/2 = 1, x + y + z = 1/2},
>  {x, y, z});
```

$$\{y = \frac{-83}{44}, z = \frac{111}{22}, x = \frac{-117}{44}\}$$

Gráficamente, esto es consistente con el hecho de que tres planos (no paralelos) se intersectan en un punto, si es que dicha intersección existe. Al igual que en el caso anterior, una sola gráfica no muestra claramente esta intersección en un punto:

```
> plots[implicitplot3d]({3*x - y + 2*z = 4, -7*x + 8*y - z/2 = 1,
> x + y + z = 1/2}, x=-4..4, y=-4..4, z=-4..4, axes=BOXED, style=patch);
```



Para trazar mediante **display3d** la superposicion de los planos y el punto de intersección, definimos la gráfica de cada uno de éstos:

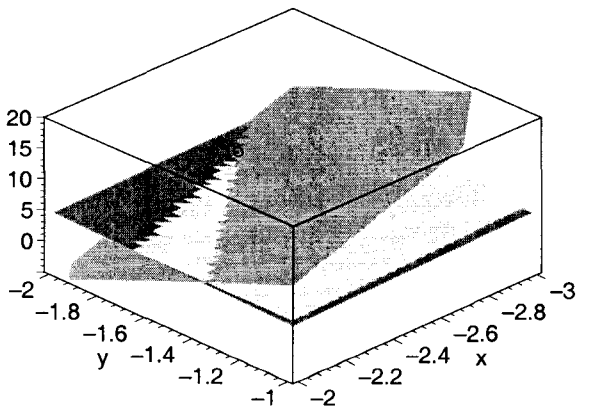
```
> plano_1 := plot3d((4 - 3*x + y)/2, x=-3..-2, y=-2..-1,
> view=-5..20, style=patchnogrid, axes=boxed, color=yellow):

> plano_2 := plot3d(2*(-7*x + 8*y - 1), x=-3..-2, y=-2..-1,
> view=-5..20, style=patchnogrid, axes=boxed, color=cyan):

> plano_3 := plot3d(-x - y + 1/2, x=-3..-2, y=-2..-1, view=-5..20,
> style=patchnogrid, axes=boxed, color=green):
```

A continuación, generamos la gráfica del punto de intersección:

```
> punto_int := plots[pointplot3d]({[-117/44, -83/44, 111/22]}, axes=BOXED,
> style=point, symbol=CIRCLE, symbolsize=25, color=black):
```

Y finalmente desplegamos todas estas estructuras en una misma gráfica:

```
> plots[display3d]({plano_1, plano_2, plano_3, punto_int});
```



De esta forma podemos apreciar el punto en el cual tiene lugar la intersección de los planos.

Capítulo 9

Estructuras de Datos

Muchos de los datos usados en Maple son manejados por medio de varias estructuras soportadas por este sistema. Es importante conocer estas estructuras, ya que son utilizadas por muchas de las instrucciones proporcionadas, así como por el lenguaje de programación soportado. Por ejemplo, para graficar dos funciones de una variable en un mismo despliegue, utilizamos la instrucción **plot** de la siguiente manera:

plot({función1, función2}, rango);

Como puede observarse, las funciones a graficar deben colocarse en la forma "**{función1, función2}**", es decir, como un conjunto. En muchas otras instrucciones los datos deben ser introducidos utilizando una estructura; además, en muchos casos los resultados también son proporcionados al usuario con una de estas estructuras.

9.1 Obtención del tipo de una expresión

Todos los elementos utilizados por Maple son clasificados en diferentes tipos; para poder manipularlos correctamente es útil saber a cual de estos pertenecen. La instruccion que nos muestra esta información es **whattype**. Veamos algunos ejemplos:

```
> whattype(3*x);
```

*

```
> whattype([3*x]);
```

list

```
> whattype({3*x});
```

set

La primera expresión es reconocida por Maple como un producto ("*****"), la segunda como una lista ("**list**") y la tercera como un conjunto ("**set**").

Maple puede reconocer el tipo de cualquier expresión, por ejemplo números o expresiones algebraicas. Los números se clasifican según sean enteros (integer), fracciones o decimales (float). Si son expresiones algebraicas, Maple las clasifica segun la operación algebraica principal. A continuacion algunos ejemplos más:

9.1.1 Números

Por ejemplo un entero, una fracción, un irracional y un número decimal:

```
> whattype(5!);
```

integer

```
> whattype(12/31);
```

fraction

```
> whattype(sqrt(Pi));
```

^

```
> whattype(1.2783);
```

float

Nótese que el tipo del irracional **sqrt(Pi)** es "^" (exponenciación), puesto que: **sqrt(Pi)=Pi^(1/2)**.

Verifiquemos el tipo de los siguientes datos:

```
> whattype(2^(1/2));
```

^

```
> whattype(3*4);
```

integer

```
> whattype(3*4 + 1);
```

integer

Tal vez esperariamos que **"2^(1/2)"** fuera reconocido como ***"float"***, sin embargo, en este caso Maple la reconoce como si se tratara de una expresión algebraica. Para determinar el tipo de estas expresiones, Maple toma en cuenta la precedencia de los operadores que aparecen en ellas.

9.1.2 Expresiones algebraicas

En expresiones simples, el tipo viene dado por la operación. En este caso, suma y resta, multiplicación y división, potencia y raíz, son consideradas como una sola operación, respectivamente:

```
> whattype(x + 3);
```

+

```
> whattype(3/x);
```

*

```
> whattype(x^3);
```

^

```
> whattype(x^(1/3));
```

^

En expresiones que involucran varias operaciones, el tipo viene dado por la operación que describe globalmente a la expresión. Por ejemplo:

```
> (3*x^2 + 3*y - 3*z)/(1 + x*y*z);
```

$$\frac{3x^2 + 3y - 3z}{1 + xyz}$$

```
> whattype(%);
```

*

Ésta última es reconocida como un producto (aunque hay sumas y potencias, estas no describen globalmente a la expresión, sino solo partes de ella).

Finalmente, determinemos el tipo de las siguientes expresiones (podemos identificar en cada una de ellas la operación que las describe):

```
> whattype(3*x^2/(1 + x*y*z) + 3*y - 3*z);
```

+

```
> whattype(3 + ((2*sin(x)^2 - sqrt(2)*cos(x))/(2*sin(x)^2 +
> sqrt(2)*cos(x))));
```

+

9.1.3 Relaciones de equivalencia

Si se trata de una ecuación o desigualdad, el signo de relación determina su tipo:

```
> whattype(3*x^2 - sqrt(y - 3) = log(1 + x*y));
```

=

```
> whattype(3 < x^3);
```

<

```
> whattype(4*x - 5*y >= 23*z);
```

<=

```
> whattype(4*x - 5*y <= 23*z);
```

<=

Nótese que ">=" y "<=" son considerados ambos como "<=". De la misma forma, "<" y ">" son considerados como "<".

9.1.4 Nombres

Si tienen alguna expresión asignada, el tipo es aquel de la expresión (o número) asignado, sin importar la forma que tenga el nombre:

```
> expr_1 := (1 + 3*x*cos(y) - a)/(1 - 3*x*cos(y) - a);
```

$$expr\_1 := \frac{1 + 3\,x\cos(y) - a}{1 - 3\,x\cos(y) - a}$$

```
> whattype(expr_1);
```

*

```
> `masa del objeto` := expr_1 + sin(x)/(1 + y);
```

$$masa\ del\ objeto := \frac{1 + 3\,x\cos(y) - a}{1 - 3\,x\cos(y) - a} + \frac{\sin(x)}{1 + y}$$

```
> whattype(`masa del objeto`);
```

+

```
> phi[0] := 1 + tan(x)^2*y;
```

$$\phi_0 := 1 + \tan(x)^2\, y$$

```
> whattype(phi[0]);
```

$$+$$

Si se trata de una expresión a la cual no se le ha asignado un valor, ésta es reconocida como ***"symbol"***:

```
> whattype(masa);
```

symbol

Una excepción son las expresiones en las cuales aparecen subíndices y aquellas que contienen funciones con valores indeterminados. Las primeras son reconocidas como expresiones ***"indexadas"*** y las segundas como ***"funciones"***.

```
> h[3];
```

$$h_3$$

```
> whattype(h[3]);
```

indexed

```
> whattype(sin(x*y));
```

function

```
> whattype(B(t));
```

function

9.1.5 Funciones

Si se definen como operador, el tipo del nombre es ***"symbol"***:

```
> Fo := (t, a) -> sqrt(1 + a*sin(t/a));
```

$$Fo := (t,\, a) \rightarrow \sqrt{1 + a \sin\left(\frac{t}{a}\right)}$$

```
> whattype(Fo);
```

symbol

Si escribimos la función con sus argumentos el tipo es el de la regla de correspondencia tomada como expresión algebraica:

```
> whattype(Fo(t, a));
```

$$\hat{}$$

Otra de las funciones útiles para verificar el tipo de un dato es **type**, su sintaxis es la siguiente:

type(expresión, tipo);

Esta función da como resultado ***"true"*** si la expresión es del tipo especificado y ***"false"*** en otro caso. Por ejemplo:

```
> type(1 - 2*I, complex);
```

true

```
> type(1/4, fraction);
```

true

```
> type(hola, integer);
```

false

Véase la página de ayuda de type para obtener una lista completa de los tipos reconocidos.

Otra función, útil también para este tipo de operaciones es **hastype**, su sintaxis es:

hastype(expresión, tipo);

Esta función da como resultado ***"true"*** si la expresión dada contiene algún elemento del tipo especificado. Por ejemplo:

```
> hastype(x^2 + 4*x, `+`);
```

true

En este caso el valor devuelto es ***"true"***, pues la expresión involucrada contiene una suma. Los tipos de datos soportados por esta función son los mismos que los de la función **type**.

9.2 Tipos de estructuras

Los principales tipos de estructuras soportados por Maple son: secuencias, conjuntos, listas, funciones, tablas y arreglos. En base a estos se manejan la mayor parte de los datos usados en este sistema. A continuación haremos una revisión de cada uno de ellos.

Nota: existe otro tipo de estructura conocido como **rtable**, el cual es usado por la función **Matrix** y por el paquete **LinearAlgebra** para creación y manejo de matrices y vectores; sin embargo, esta estructura no es utilizada en otros casos, por lo que no la describiremos aquí.

9.2.1 Secuencias de expresiones

Si tecleamos varias expresiones (cualquier combinación de los tipos anteriormente mencionados) separadas por comas, obtenemos una estructura llamada ***"secuencia de expresiones"***, que Maple reconoce con el nombre ***"exprseq"***. Por ejemplo:

```
> 1, 2, 3, sqrt(x + 1), sin(x), S[12], C(x);
```

$$1, 2, 3, \sqrt{x+1}, \sin(x), S_{12}, \mathrm{C}(x)$$

```
> whattype(%);
```

exprseq

Estas secuencias de expresiones pueden ser asignadas a variables:

```
> hola := 1.1, sqrt(23), 3, 7, 89, 23;
```

$$hola := 1.1, \sqrt{23}, 3, 7, 89, 23$$

```
> whattype(hola);
```

exprseq

Existen varias forma de definir una secuencia; una de ellas es la del ejemplo anterior, simplemente colocamos los elementos de manera consecutiva separados por comas. Pero imaginemos el problema que representa formar una estructura de este tipo que contenga los primeros 100 números naturales, tecleando cada uno

de ellos. Maple nos proporciona una forma de tratar con este problema, permitiendonos crear este tipo de estructuras por ***"definición secuencial"***, lo cual puede llevarse a cabo por medio de la instrucción **seq**.

La sintaxis de esta función es:

seq(r(i), i=rango);

Donde **r(i)** es una expresión en términos de la variable **i**, que determina de que manera se generan los elementos de la secuencia, y **rango** determina que valores tomará **i**; es decir, cuantos elementos y con que valores de **i** estarán dentro de esta secuencia. Por ejemplo, para obtener una secuencia que contenga los 100 primeros números naturales podemos utilizar la siguiente instrucción:

```
> seq(i, i=1..100);
```

1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100

El índice de la secuencia (**i** en el ejemplo) puede ser cualquier cadena de caracteres (a la cual no le ha sido asignado previamente un dato), y puede tomar valores enteros negativos. También es posible asignar un nombre a la secuencia obtenida. Por ejemplo, la secuencia de cuadrados de enteros en el rango -10..10 puede ser obtenida por:

```
> cuadr := seq(entero^2, entero=-10..10);
```

cuadr := 100, 81, 64, 49, 36, 25, 16, 9, 4, 1, 0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100

Es posible obtener de esta forma secuencias como las que se muestran a continuación:

- Secuencias a partir de funciones.

```
> C := x -> x + 4;
```

$$C := x \to x + 4$$

```
> sec1 := seq(C(k) + k*sin((k - 1)), k=1..10);
```

sec1 := $5, 6 + 2\sin(1), 7 + 3\sin(2), 8 + 4\sin(3), 9 + 5\sin(4), 10 + 6\sin(5), 11 + 7\sin(6), 12 + 8\sin(7), 13 + 9\sin(8), 14 + 10\sin(9)$

- Secuencias que involucran relaciones de equivalencia.

```
> sec2 := seq(C(n) + 1 = n*sin(n*x + 1), n=-3..4);
```

sec2 := $2 = 3\sin(3x - 1), 3 = 2\sin(2x - 1), 4 = \sin(x - 1), 5 = 0, 6 = \sin(x + 1), 7 = 2\sin(2x + 1), 8 = 3\sin(3x + 1), 9 = 4\sin(4x + 1)$

- Objetos indexados.

```
> sec3 := seq(A[K + 1] - B[K] + C(K), K=-5..5);
```

sec3 := $A_{-4} - B_{-5} - 1, A_{-3} - B_{-4}, A_{-2} - B_{-3} + 1, A_{-1} - B_{-2} + 2, A_0 - B_{-1} + 3, A_1 - B_0 + 4, A_2 - B_1 + 5, A_3 - B_2 + 6, A_4 - B_3 + 7, A_5 - B_4 + 8, A_6 - B_5 + 9$

- Secuencias anidadas: sec(sec()).

```
> seq(alpha*m^2 + beta*n^2, m=1..3);
```

$$\alpha + \beta n^2, 4\alpha + \beta n^2, 9\alpha + \beta n^2$$

```
> seq(%, n=1..3);
```

$$\alpha+\beta,\ 4\alpha+\beta,\ 9\alpha+\beta,\ \alpha+4\beta,\ 4\alpha+4\beta,\ 9\alpha+4\beta,\ \alpha+9\beta,\ 4\alpha+9\beta,\ 9\alpha+9\beta$$

Esta última podemos anidarla en una misma expresión:

```
> seq(seq(alpha*m^2 + beta*n^2, m=1..3), n=1..3);
```

$$\alpha+\beta,\ 4\alpha+\beta,\ 9\alpha+\beta,\ \alpha+4\beta,\ 4\alpha+4\beta,\ 9\alpha+4\beta,\ \alpha+9\beta,\ 4\alpha+9\beta,\ 9\alpha+9\beta$$

```
> seq(seq(P[k, l], k=1..3), l=1..4);
```

$$P_{1,1},\ P_{2,1},\ P_{3,1},\ P_{1,2},\ P_{2,2},\ P_{3,2},\ P_{1,3},\ P_{2,3},\ P_{3,3},\ P_{1,4},\ P_{2,4},\ P_{3,4}$$

- Particiones de intervalos en los reales. Por ejemplo, el intervalo x=0..1 en diez partes iguales.

```
> seq(i/10, i=0..10);
```

$$0,\ \frac{1}{10},\ \frac{1}{5},\ \frac{3}{10},\ \frac{2}{5},\ \frac{1}{2},\ \frac{3}{5},\ \frac{7}{10},\ \frac{4}{5},\ \frac{9}{10},\ 1$$

Otros ejemplos

Mediante una secuencia crearemos las siguientes particiones de intervalos de la recta de los reales. Por razones de espacio no mostraremos todas las salidas, pues algunas de ellas son bastante extensas.

- El intervalo de [0..1] en 20 partes iguales:

```
> seq(i/20, i=0..20);
```

$$0,\ \frac{1}{20},\ \frac{1}{10},\ \frac{3}{20},\ \frac{1}{5},\ \frac{1}{4},\ \frac{3}{10},\ \frac{7}{20},\ \frac{2}{5},\ \frac{9}{20},\ \frac{1}{2},\ \frac{11}{20},\ \frac{3}{5},\ \frac{13}{20},\ \frac{7}{10},\ \frac{3}{4},\ \frac{4}{5},\ \frac{17}{20},\ \frac{9}{10},\ \frac{19}{20},\ 1$$

- El intervalo [-7..5.33] en 59 partes iguales:

```
> seq(-7 + (5.33 - (-7))*i/59, i=0..59);
```

$-7.,\ -6.791016949,\ -6.582033898,\ -6.373050848,\ -6.164067797,\ -5.955084746,$
$-5.746101695,\ -5.537118644,\ -5.328135594,\ -5.119152543,\ -4.910169492,$
$-4.701186441,\ -4.492203390,\ -4.283220340,\ -4.074237289,\ -3.865254238,$
$-3.656271187,\ -3.447288136,\ -3.238305086,\ -3.029322035,\ -2.820338984,$
$-2.611355933,\ -2.402372882,\ -2.193389832,\ -1.984406781,\ -1.775423730,$
$-1.566440679,\ -1.357457628,\ -1.148474578,\ -0.939491527,\ -0.730508476,$
$-0.521525425,\ -0.312542374,\ -0.103559324,\ 0.105423727,\ 0.314406778,$
$0.523389829,\ 0.732372880,\ 0.941355930,\ 1.150338981,\ 1.359322032,$
$1.568305083,\ 1.777288134,\ 1.986271184,\ 2.195254235,\ 2.404237286,$
$2.613220337,\ 2.822203388,\ 3.03118644,\ 3.24016949,\ 3.44915254,\ 3.65813559,$
$3.86711864,\ 4.07610169,\ 4.28508474,\ 4.49406779,\ 4.70305084,\ 4.91203390,$
$5.12101695,\ 5.33000000$

- Definir quince ángulos iguales en un círculo:

```
> seq(2*Pi*i/15, i=0..15);
```

$$0,\ \frac{2\pi}{15},\ \frac{4\pi}{15},\ \frac{2\pi}{5},\ \frac{8\pi}{15},\ \frac{2\pi}{3},\ \frac{4\pi}{5},\ \frac{14\pi}{15},\ \frac{16\pi}{15},\ \frac{6\pi}{5},\ \frac{4\pi}{3},\ \frac{22\pi}{15},\ \frac{8\pi}{5},\ \frac{26\pi}{15},\ \frac{28\pi}{15},\ 2\pi$$

Ahora veamos algunos ejemplos de particiones en el plano:

- Una malla de 10X20 en los rangos: x=-2..2, y=0..3:

```
> seq(seq([-2 + 4*n/9, 3*m/19], m=0..19), n=0..9);
```

$$[-2, 0], [-2, \frac{3}{19}], [-2, \frac{6}{19}], [-2, \frac{9}{19}], [-2, \frac{12}{19}], [-2, \frac{15}{19}], [-2, \frac{18}{19}], [-2, \frac{21}{19}], [-2, \frac{24}{19}], [-2, \frac{27}{19}],$$
$$[-2, \frac{30}{19}], [-2, \frac{33}{19}], [-2, \frac{36}{19}], [-2, \frac{39}{19}], [-2, \frac{42}{19}], [-2, \frac{45}{19}], [-2, \frac{48}{19}], [-2, \frac{51}{19}], [-2, \frac{54}{19}],$$
$$[-2, 3], [\frac{-14}{9}, 0], [\frac{-14}{9}, \frac{3}{19}], [\frac{-14}{9}, \frac{6}{19}], [\frac{-14}{9}, \frac{9}{19}], [\frac{-14}{9}, \frac{12}{19}], [\frac{-14}{9}, \frac{15}{19}], [\frac{-14}{9}, \frac{18}{19}],$$
$$[\frac{-14}{9}, \frac{21}{19}], [\frac{-14}{9}, \frac{24}{19}], [\frac{-14}{9}, \frac{27}{19}], [\frac{-14}{9}, \frac{30}{19}], [\frac{-14}{9}, \frac{33}{19}], [\frac{-14}{9}, \frac{36}{19}], [\frac{-14}{9}, \frac{39}{19}], [\frac{-14}{9}, \frac{42}{19}],$$
$$[\frac{-14}{9}, \frac{45}{19}], [\frac{-14}{9}, \frac{48}{19}], [\frac{-14}{9}, \frac{51}{19}], [\frac{-14}{9}, \frac{54}{19}], [\frac{-14}{9}, 3], [\frac{-10}{9}, 0], [\frac{-10}{9}, \frac{3}{19}], [\frac{-10}{9}, \frac{6}{19}],$$
$$[\frac{-10}{9}, \frac{9}{19}], [\frac{-10}{9}, \frac{12}{19}], [\frac{-10}{9}, \frac{15}{19}], [\frac{-10}{9}, \frac{18}{19}], [\frac{-10}{9}, \frac{21}{19}], [\frac{-10}{9}, \frac{24}{19}], [\frac{-10}{9}, \frac{27}{19}], [\frac{-10}{9}, \frac{30}{19}],$$
$$[\frac{-10}{9}, \frac{33}{19}], [\frac{-10}{9}, \frac{36}{19}], [\frac{-10}{9}, \frac{39}{19}], [\frac{-10}{9}, \frac{42}{19}], [\frac{-10}{9}, \frac{45}{19}], [\frac{-10}{9}, \frac{48}{19}], [\frac{-10}{9}, \frac{51}{19}], [\frac{-10}{9}, \frac{54}{19}],$$
$$[\frac{-10}{9}, 3], [\frac{-2}{3}, 0], [\frac{-2}{3}, \frac{3}{19}], [\frac{-2}{3}, \frac{6}{19}], [\frac{-2}{3}, \frac{9}{19}], [\frac{-2}{3}, \frac{12}{19}], [\frac{-2}{3}, \frac{15}{19}], [\frac{-2}{3}, \frac{18}{19}], [\frac{-2}{3}, \frac{21}{19}],$$
$$[\frac{-2}{3}, \frac{24}{19}], [\frac{-2}{3}, \frac{27}{19}], [\frac{-2}{3}, \frac{30}{19}], [\frac{-2}{3}, \frac{33}{19}], [\frac{-2}{3}, \frac{36}{19}], [\frac{-2}{3}, \frac{39}{19}], [\frac{-2}{3}, \frac{42}{19}], [\frac{-2}{3}, \frac{45}{19}], [\frac{-2}{3}, \frac{48}{19}],$$
$$[\frac{-2}{3}, \frac{51}{19}], [\frac{-2}{3}, \frac{54}{19}], [\frac{-2}{3}, 3], [\frac{-2}{9}, 0], [\frac{-2}{9}, \frac{3}{19}], [\frac{-2}{9}, \frac{6}{19}], [\frac{-2}{9}, \frac{9}{19}], [\frac{-2}{9}, \frac{12}{19}], [\frac{-2}{9}, \frac{15}{19}],$$
$$[\frac{-2}{9}, \frac{18}{19}], [\frac{-2}{9}, \frac{21}{19}], [\frac{-2}{9}, \frac{24}{19}], [\frac{-2}{9}, \frac{27}{19}], [\frac{-2}{9}, \frac{30}{19}], [\frac{-2}{9}, \frac{33}{19}], [\frac{-2}{9}, \frac{36}{19}], [\frac{-2}{9}, \frac{39}{19}], [\frac{-2}{9}, \frac{42}{19}],$$
$$[\frac{-2}{9}, \frac{45}{19}], [\frac{-2}{9}, \frac{48}{19}], [\frac{-2}{9}, \frac{51}{19}], [\frac{-2}{9}, \frac{54}{19}], [\frac{-2}{9}, 3], [\frac{2}{9}, 0], [\frac{2}{9}, \frac{3}{19}], [\frac{2}{9}, \frac{6}{19}], [\frac{2}{9}, \frac{9}{19}], [\frac{2}{9}, \frac{12}{19}],$$
$$[\frac{2}{9}, \frac{15}{19}], [\frac{2}{9}, \frac{18}{19}], [\frac{2}{9}, \frac{21}{19}], [\frac{2}{9}, \frac{24}{19}], [\frac{2}{9}, \frac{27}{19}], [\frac{2}{9}, \frac{30}{19}], [\frac{2}{9}, \frac{33}{19}], [\frac{2}{9}, \frac{36}{19}], [\frac{2}{9}, \frac{39}{19}], [\frac{2}{9}, \frac{42}{19}],$$
$$[\frac{2}{9}, \frac{45}{19}], [\frac{2}{9}, \frac{48}{19}], [\frac{2}{9}, \frac{51}{19}], [\frac{2}{9}, \frac{54}{19}], [\frac{2}{9}, 3], [\frac{2}{3}, 0], [\frac{2}{3}, \frac{3}{19}], [\frac{2}{3}, \frac{6}{19}], [\frac{2}{3}, \frac{9}{19}], [\frac{2}{3}, \frac{12}{19}],$$
$$[\frac{2}{3}, \frac{15}{19}], [\frac{2}{3}, \frac{18}{19}], [\frac{2}{3}, \frac{21}{19}], [\frac{2}{3}, \frac{24}{19}], [\frac{2}{3}, \frac{27}{19}], [\frac{2}{3}, \frac{30}{19}], [\frac{2}{3}, \frac{33}{19}], [\frac{2}{3}, \frac{36}{19}], [\frac{2}{3}, \frac{39}{19}], [\frac{2}{3}, \frac{42}{19}],$$
$$[\frac{2}{3}, \frac{45}{19}], [\frac{2}{3}, \frac{48}{19}], [\frac{2}{3}, \frac{51}{19}], [\frac{2}{3}, \frac{54}{19}], [\frac{2}{3}, 3], [\frac{10}{9}, 0], [\frac{10}{9}, \frac{3}{19}], [\frac{10}{9}, \frac{6}{19}], [\frac{10}{9}, \frac{9}{19}], [\frac{10}{9}, \frac{12}{19}],$$
$$[\frac{10}{9}, \frac{15}{19}], [\frac{10}{9}, \frac{18}{19}], [\frac{10}{9}, \frac{21}{19}], [\frac{10}{9}, \frac{24}{19}], [\frac{10}{9}, \frac{27}{19}], [\frac{10}{9}, \frac{30}{19}], [\frac{10}{9}, \frac{33}{19}], [\frac{10}{9}, \frac{36}{19}], [\frac{10}{9}, \frac{39}{19}],$$
$$[\frac{10}{9}, \frac{42}{19}], [\frac{10}{9}, \frac{45}{19}], [\frac{10}{9}, \frac{48}{19}], [\frac{10}{9}, \frac{51}{19}], [\frac{10}{9}, \frac{54}{19}], [\frac{10}{9}, 3], [\frac{14}{9}, 0], [\frac{14}{9}, \frac{3}{19}], [\frac{14}{9}, \frac{6}{19}],$$
$$[\frac{14}{9}, \frac{9}{19}], [\frac{14}{9}, \frac{12}{19}], [\frac{14}{9}, \frac{15}{19}], [\frac{14}{9}, \frac{18}{19}], [\frac{14}{9}, \frac{21}{19}], [\frac{14}{9}, \frac{24}{19}], [\frac{14}{9}, \frac{27}{19}], [\frac{14}{9}, \frac{30}{19}], [\frac{14}{9}, \frac{33}{19}],$$
$$[\frac{14}{9}, \frac{36}{19}], [\frac{14}{9}, \frac{39}{19}], [\frac{14}{9}, \frac{42}{19}], [\frac{14}{9}, \frac{45}{19}], [\frac{14}{9}, \frac{48}{19}], [\frac{14}{9}, \frac{51}{19}], [\frac{14}{9}, \frac{54}{19}], [\frac{14}{9}, 3], [2, 0],$$
$$[2, \frac{3}{19}], [2, \frac{6}{19}], [2, \frac{9}{19}], [2, \frac{12}{19}], [2, \frac{15}{19}], [2, \frac{18}{19}], [2, \frac{21}{19}], [2, \frac{24}{19}], [2, \frac{27}{19}], [2, \frac{30}{19}],$$
$$[2, \frac{33}{19}], [2, \frac{36}{19}], [2, \frac{39}{19}], [2, \frac{42}{19}], [2, \frac{45}{19}], [2, \frac{48}{19}], [2, \frac{51}{19}], [2, \frac{54}{19}], [2, 3]$$

Podemos verificar la cantidad de elementos generados por esta última instrucción, utilizando **nops**, la cual nos da el número de elementos que contiene una lista.

Para que considere esta última secuencia como lista basta con encerrarla entre corchetes.

```
> nops([%]);
```

200

- Una malla 17X13 para los intervalos x=-Pi..Pi, y=0..2*Pi:

```
> seq(seq([-Pi + 2*Pi*n/9, 2*Pi*m/19], m=0..12), n=0..16):
```

Por razones de espacio no mostramos los elementos generados; sin embargo, podemos comprobar el número de éstos creados:

```
> nops([%]);
```

221

- Finalmente, veamos el siguiente ejemplo:

Considérese la sucesión **S := seq(A[k] + cos(1 + B[k + 1]), k=-3..3)** de 7 elementos.

¿Cómo podemos formar la sucesión **A[k] + cos(1 + B[k + 1])**, de tal manera que **k** tome siete valores diferentes entre -3/2 y 3/2?

```
> S := seq(A[k/2] + cos(1 + B[k/2 + 1]), k=-3..3);
```

$S := A_{-3/2} + \cos(1 + B_{-1/2}), A_{-1} + \cos(1 + B_0), A_{-1/2} + \cos(1 + B_{1/2}), A_0 + \cos(1 + B_1),$
$A_{1/2} + \cos(1 + B_{3/2}), A_1 + \cos(1 + B_2), A_{3/2} + \cos(1 + B_{5/2})$

Acceso a los elementos de una secuencia

Consideremos las siguientes secuencias:

```
> S_a := seq(cos(x + i/5), i=0..5);
```

$$S\_a := \cos(x), \cos\left(x + \frac{1}{5}\right), \cos\left(x + \frac{2}{5}\right), \cos\left(x + \frac{3}{5}\right), \cos\left(x + \frac{4}{5}\right), \cos(x + 1)$$

```
> S_b := seq(y + i/6, i=0..6);
```

$$S\_b := y, y + \frac{1}{6}, y + \frac{1}{3}, y + \frac{1}{2}, y + \frac{2}{3}, y + \frac{5}{6}, 1 + y$$

Si queremos extraer el tercer elemento de **S_a** , usamos subíndices de la siguiente forma:

```
> S_a[3];
```

$$\cos\left(x + \frac{2}{5}\right)$$

Es posible formar subsecuencias a partir de secuencias ya definidas. Para esto hay dos formas equivalentes; por ejemplo, podemos hacerlo utilizando la instrucción **seq** de la siguiente manera:

```
> seq(S_a[j], j=2..5);
```

$$\cos\left(x + \frac{1}{5}\right), \cos\left(x + \frac{2}{5}\right), \cos\left(x + \frac{3}{5}\right), \cos\left(x + \frac{4}{5}\right)$$

Con esto extraemos los elementos del segundo al quinto de **S_a**. Otra forma es la siguiente:

```
> S_a[2..5];
```

$$\cos\left(x + \frac{1}{5}\right), \cos\left(x + \frac{2}{5}\right), \cos\left(x + \frac{3}{5}\right), \cos\left(x + \frac{4}{5}\right)$$

También podemos definir secuencias nuevas en base a las secuencia previamente definidas. Por ejemplo, la secuencia de cosenos de cada elemento de **S_b** está dada por:

```
> S_c := seq(cos(S_b[i]), i=1..7);
```

$$S\_c := \cos(y), \cos\left(y+\frac{1}{6}\right), \cos\left(y+\frac{1}{3}\right), \cos\left(y+\frac{1}{2}\right), \cos\left(y+\frac{2}{3}\right), \cos\left(y+\frac{5}{6}\right), \cos(1+y)$$

Nota: al definir **S_c**, el índice **i** no puede tomar valores negativos ni cero, pues denota un orden de los elementos de **S_b** (no hay elemento ***"cero-ésimo"***, ni ***"menos cinco"***). También, el máximo valor del índice no debe exceder el número de elementos de **S_b** (si tiene 7 elementos, no podemos invocar al octavo elemento).

9.2.2 Listas y conjuntos

Las listas y los conjuntos están formados por expresiones y secuencias de expresiones. Existe una diferencia muy importante entre las listas y los conjuntos. En el caso de las listas el orden de los elementos es importante; en cambio en los conjuntos el orden es irrelevante(al igual que en los conjuntos matemáticos).

Por otro lado, los elementos de listas y conjuntos se pueden definir como en las secuencias, proporcionando cada uno de los elementos separados por coma o bien por definicion secuencial. La sintaxis para crear una de estas estructuras se muestra a continuación.

Para definir una lista:

[e1, e2, e3, ...] , donde **"e1, e2, e3, ..."** son los elementos.

O bien:

[seq(...)] , utilizando la definición secuencial.

El caso de los conjuntos es análogo, pero los elementos deben estar delimitados por llaves (**"{ }"**):

{e1, e2, e3, ...} , donde **"e1, e2, e3, ..."** son los elementos.

O bien,

{seq(...)} , utilizando la definición secuencial.

Maple considera a las listas, los conjuntos y los elementos que los forman como estructuras diferentes. Por ejemplo, definamos la siguiente secuencia:

```
> sq := seq(i/9, i=0..9);
```

$$sq := 0, \frac{1}{9}, \frac{2}{9}, \frac{1}{3}, \frac{4}{9}, \frac{5}{9}, \frac{2}{3}, \frac{7}{9}, \frac{8}{9}, 1$$

Veamos el tipo de **sq** y los tipos de la lista y el conjunto formados por sus elementos:

```
> whattype(sq);
```

exprseq

```
> listasq := [sq];
```

$$listasq := \left[0, \frac{1}{9}, \frac{2}{9}, \frac{1}{3}, \frac{4}{9}, \frac{5}{9}, \frac{2}{3}, \frac{7}{9}, \frac{8}{9}, 1\right]$$

```
> whattype(listasq);
```

list

```
> conjsq := {sq};
```

$$conjsq := \left\{0, 1, \frac{5}{9}, \frac{1}{3}, \frac{2}{3}, \frac{4}{9}, \frac{2}{9}, \frac{8}{9}, \frac{1}{9}, \frac{7}{9}\right\}$$

```
> whattype({conjsq});
```

$$set$$

Nótese que una forma de crear una lista o un conjunto es encerrando entre corchetes o llaves (respectivamente) los elementos de una secuencia.

Para ilustrar la diferencia entre listas y conjuntos, examinemos el ejemplo siguiente.

Definimos la secuencia **elems**:

```
> elems := a, a, a, a, 1+a, sin(a), sin(1 + a);
```

$$elems := a, a, a, a, 1 + a, \sin(a), \sin(1 + a)$$

A continuación utilizamos esta secuencia para crear una lista o un conjunto.

```
> Lista := [elems];
```

$$Lista := [a, a, a, a, 1 + a, \sin(a), \sin(1 + a)]$$

```
> Conjunto := {elems};
```

$$Conjunto := \{a, 1 + a, \sin(a), \sin(1 + a)\}$$

Como puede verse, los componentes de **elems** son colocados íntegramente en la lista; en cambio en el conjunto se eliminan los duplicados y además estos elementos no necesariamente aparecen en el mismo orden.

Por otro lado, las listas y los conjuntos pueden ser manipulados de la misma forma que las secuencias:

```
> Lista;
```

$$[a, a, a, a, 1 + a, \sin(a), \sin(1 + a)]$$

```
> Lista[2];
```

$$a$$

```
> Lista[2..5];
```

$$[a, a, a, 1 + a]$$

```
> Conjunto;
```

$$\{a, 1 + a, \sin(a), \sin(1 + a)\}$$

```
> Conjunto[2];
```

$$1 + a$$

```
> Conjunto[1..3];
```

$$\{a, 1 + a, \sin(a)\}$$

Nota: aunque el orden no es importante en los conjuntos, al hacer referencia a un elemento, por ejemplo **Conjunto[2]**, el elemento que considera Maple es, en este caso, el segundo dependiendo del orden que tenía la salida cuando éste fue definido.

Para invocar a todos los elementos de una lista o conjunto usamos la instrucción **op** y para obtener el número de elementos usamos **nops**, como se muestra a continuación:

```
> Lista;
```

$$[a, a, a, a, 1 + a, \sin(a), \sin(1 + a)]$$

```
> op(Lista);
```

$$a, a, a, a, 1+a, \sin(a), \sin(1+a)$$

```
> nops(Lista);
```

$$7$$

```
> Conjunto;
```

$$\{a, 1+a, \sin(a), \sin(1+a)\}$$

```
> op(Conjunto);
```

$$a, 1+a, \sin(a), \sin(1+a)$$

```
> nops(Conjunto);
```

$$4$$

Por ejemplo, para formar un conjunto con los elementos de una lista podemos proceder de la siguiente forma:

```
> Lis2 := [a, c, d, g, r, t];
```

$$Lis2 := [a, c, d, g, r, t]$$

```
> Conj2 := {op(Lis2)};
```

$$Conj2 := \{g, a, r, c, t, d\}$$

La instrucción **op** también permite invocar uno o más elementos, de la siguiente forma:

```
> Lista;
```

$$[a, a, a, a, 1+a, \sin(a), \sin(1+a)]$$

```
> op(2, Lista);
```

$$a$$

```
> op(2..6, Lista);
```

$$a, a, a, 1+a, \sin(a)$$

```
> Conjunto;
```

$$\{a, 1+a, \sin(a), \sin(1+a)\}$$

```
> op(4, Conjunto);
```

$$\sin(1+a)$$

```
> op(2..4, Conjunto);
```

$$1+a, \sin(a), \sin(1+a)$$

Nota: las instrucciones **op** y **nops** no actuan en secuencias, solo en listas y conjuntos.

La instrucción **ops** es útil para construir listas o conjuntos nuevos, a partir de listas o conjuntos ya definidos. La instrucción **nops** también es muy útil, ya que al hacer uso de ella evitamos tener que contabilizar los elementos de la lista (o el conjunto) ya existente. Por ejemplo, a partir de los elementos de **Lista**, podemos crear fácilmente otra lista nueva, aplicando la función **cos** a cada elemento:

Primero construimos los elementos de la nueva lista:

```
> elems_nuevos := seq(cos(Lista[j]), j=1..nops(Lista));
```

$$elems\_nuevos := \cos(a), \cos(a), \cos(a), \cos(a), \cos(1+a), \cos(\sin(a)), \cos(\sin(1+a))$$

Nótese que esta lista está formada únicamente por elementos simbólicos.

A continuación, utilizando esta secuencia, construimos la lista nueva:

```
> Lista_nueva := [elems_nuevos];
```

$$\textit{Lista\_nueva} := [\cos(a), \cos(a), \cos(a), \cos(a), \cos(1+a), \cos(\sin(a)), \cos(\sin(1+a))]$$

Obviamente, esto lo podemos hacer en un solo paso. Por ejemplo, queremos una lista formada por los elementos de **Conjunto** elevados al cubo:

```
> Otra_Lista := [seq(Conjunto[j]^3, j=1..nops(Conjunto))];
```

$$\textit{Otra\_Lista} := [a^3, (1+a)^3, \sin(a)^3, \sin(1+a)^3]$$

Listas de listas, listas de conjuntos, conjuntos de listas, ...

Es posible construir listas o conjuntos cuyos elementos son a su vez listas o conjuntos. Como ejemplo, formamos la lista cuyos elementos son ***Lista_nueva*** y **Otra_lista**, definidos previamente.

```
> Superlista := [Lista_nueva, Otra_Lista];
```

$$\textit{Superlista} := [[\cos(a), \cos(a), \cos(a), \cos(a), \cos(1+a), \cos(\sin(a)), \cos(\sin(1+a))],$$
$$[a^3, (1+a)^3, \sin(a)^3, \sin(1+a)^3]]$$

Ahora, **Superlista** tiene solo dos elementos (las dos listas):

```
> nops(Superlista);
```

$$2$$

```
> op(Superlista);
```

$$[\cos(a), \cos(a), \cos(a), \cos(a), \cos(1+a), \cos(\sin(a)), \cos(\sin(1+a))],$$
$$[a^3, (1+a)^3, \sin(a)^3, \sin(1+a)^3]$$

El primero y el segundo elementos son las listas a partir de las cuales fue generado:

```
> Superlista[1];
```

$$[\cos(a), \cos(a), \cos(a), \cos(a), \cos(1+a), \cos(\sin(a)), \cos(\sin(1+a))]$$

```
> Superlista[2];
```

$$[a^3, (1+a)^3, \sin(a)^3, \sin(1+a)^3]$$

Para invocar algún(os) elemento(s) de las listas componentes a partir de la lista grande tenemos que hacerlo de manera iterada. Por ejemplo, para invocar a la expresión sin(1+a)^3:

```
> Superlista[2][4];
```

$$\sin(1+a)^3$$

Es decir, el cuarto elemento del segundo componente. O bien, podemos hacerlo de las siguientes dos formas:

```
> op(4, Superlista[2]);
```

$$\sin(1+a)^3$$

```
> op(4, op(2, Superlista));
```

$$\sin(1+a)^3$$

Un ejemplo interesante de extracción de elementos es el que se muestra a continuación:

```
> CC := [seq([i/10, f[i] = cos(i/10), g[i] = exp(-i/10)], i=0..10)];
```

$$CC := [[0, f_0 = 1, g_0 = 1], [\frac{1}{10}, f_1 = \cos(\frac{1}{10}), g_1 = e^{(\frac{-1}{10})}], [\frac{1}{5}, f_2 = \cos(\frac{1}{5}), g_2 = e^{(-1/5)}],$$
$$[\frac{3}{10}, f_3 = \cos(\frac{3}{10}), g_3 = e^{(\frac{-3}{10})}], [\frac{2}{5}, f_4 = \cos(\frac{2}{5}), g_4 = e^{(-2/5)}], [\frac{1}{2}, f_5 = \cos(\frac{1}{2}), g_5 = e^{(-1/2)}],$$
$$[\frac{3}{5}, f_6 = \cos(\frac{3}{5}), g_6 = e^{(-3/5)}], [\frac{7}{10}, f_7 = \cos(\frac{7}{10}), g_7 = e^{(\frac{-7}{10})}], [\frac{4}{5}, f_8 = \cos(\frac{4}{5}), g_8 = e^{(-4/5)}],$$
$$[\frac{9}{10}, f_9 = \cos(\frac{9}{10}), g_9 = e^{(\frac{-9}{10})}], [1, f_{10} = \cos(1), g_{10} = e^{(-1)}]]$$

Para extraer la expresión *"cos(7/10)"* tenemos que invocar, del octavo elemento de la lista (es decir, de la sublista número 8), el lado derecho del segundo elemento:

```
> CC[8];
```

$$\left[\frac{7}{10}, f_7 = \cos\left(\frac{7}{10}\right), g_7 = e^{(\frac{-7}{10})}\right]$$

```
> CC[8][2];
```

$$f_7 = \cos\left(\frac{7}{10}\right)$$

```
> rhs(CC[8][2]);
```

$$\cos\left(\frac{7}{10}\right)$$

Otra forma de hacerlo es:

```
> rhs(op(2, CC[8]));
```

$$\cos\left(\frac{7}{10}\right)$$

O bien:

```
> rhs(op(2, op(8, CC)));
```

$$\cos\left(\frac{7}{10}\right)$$

9.2.3 Funciones

Otra de las estructuras soportadas por Maple son las funciones, las cuales ya han sido tratadas previamente. Veamos algunos ejemplos:

```
> f1 := x -> x^2 + 4;
```

$$f1 := x \rightarrow x^2 + 4$$

```
> f2 := (a, b) -> a^b;
```

$$f2 := (a, b) \rightarrow a^b$$

```
> f3 := x - y^3;
```

$$f3 := x - y^3$$

Aplicamos la función **whattype** a cada una de ellas:

```
> whattype(f1(x));
```

+

```
> whattype(f2(a, b));
```

^

```
> whattype(f3);
```

+

Como habiamos visto previamente, el tipo de cada una de las funciones está dado por el tipo de la regla de correspondencia.

9.2.4 Tablas

Las tablas son un tipo de estructura cuyos elementos están dados por un grupo de datos, cada uno de los cuales tiene asignado un índice. La forma de crear estas estructuras es a través de la función **table**, dando los elementos como una lista de relaciones de la siguiente forma:

table([el1 = dato1, el2 = dato2, el3 = dato3, ...]);

donde **"el1, el2, el3, ..."** son los índices de los elementos y **"dato1, dato2, dato3, ..."** son los elementos asignados a cada uno de ellos. Por ejemplo, definamos la siguiente tabla:

```
> valores := table([A=2, B=4331, C=.3382, D=sqrt(Pi^2), E=hola]);
```

$$valores := \text{table}([A = 2, E = (1.1, \sqrt{23}, 3, 7, 89, 23), \mathrm{D} = \pi, C = 0.3382, B = 4331])$$

El acceso a un elemento de una tabla se hace a través de los índices. Por ejemplo, para acceder a los elementos indexados por **D** y **B**, usamos la siguiente instrucción:

```
> valores[D];
```

$$\pi$$

```
> valores[B];
```

$$4331$$

Los índices utilizados en una tabla pueden estar formados por cualquier expresión válida, incluso pueden incluirse expresiones con espacios, siempre que sean colocadas como cadenas. Por ejemplo:

```
> t := table(['dato 1' = 'cadena de caracteres', 'dato 2' = 75452]);
```

$$t := \text{table}([\textit{dato 2} = 75452, \textit{dato 1} = \textit{cadena de caracteres}])$$

Para acceder a los elementos también es necesario expresar los índices como cadenas (en este caso).

```
> t['dato 1'];
```

cadena de caracteres

```
> t['dato 2'];
```

$$75452$$

También pueden usarse números como índices.

```
> t2 := table([1 = 67832, 2 = 'cadena x', 5 = Pi^2]);
```

$$t2 := \text{table}([1 = 67832, 2 = \textit{cadena } x, 5 = \pi^2])$$

De cualquier forma, sus elementos deben ser accesados a través de los índices asignados.

```
> t2[1];
```

$$67832$$

```
> t2[5];
```

$$\pi^2$$

```
> t2[2];
```

$$cadena\ x$$

Nótese que, en el caso de las tablas, el índice no representa la posición en la cual se encuentra el elemento, es decir, las tablas no son manejadas como listas. Dicho índice es usado únicamente como identificador para cada uno de los elementos. En caso de que al definir la tabla no se especifiquen índices, la función asigna números enteros, tomando en cuenta el orden en que fueron proporcionados los elementos. Por ejemplo:

```
> t3 := table([x, marzo, 98.431, cuatro + seis]);
```

$$t3 := \text{table}([1 = x, 2 = marzo, 3 = 98.431, 4 = cuatro + seis])$$

En este caso podemos acceder a los elementos usando como índice su posición dentro de la tabla:

```
> t3[1];
```

$$x$$

```
> t3[2];
```

$$marzo$$

```
> t3[3];
```

$$98.431$$

```
> t3[4];
```

$$cuatro + seis$$

Los elementos de una tabla pueden ser proporcionados también en forma de conjunto, pero en caso de que no se proporcionen índices (como en el ejemplo anterior), el índice de los elementos no necesariamente corresponde a la posición en la cual fueron colocados (tómese en cuenta que se está creando la tabla a partir de un conjunto y en este el orden es irrelevante). Por ejemplo:

```
> t3 := table({x, marzo, 98.431, cuatro + seis});
```

$$t3 := \text{table}([1 = x, 2 = marzo, 3 = cuatro + seis, 4 = 98.431])$$

Una vez creada una tabla podemos agregarle un elemento usando una instrucción como la siguiente:

```
> t3[5] := 24;
```

$$t3_5 := 24$$

Obviamente el elemento 5 no debe existir, en caso contrario modificariamos uno de los ya existentes.

En este tipo de estructuras, para poder visualizar todos los elementos es necesario utilizar la función **eval**:

```
> eval(t3);
```

$$\text{table}([1 = x, 2 = marzo, 3 = cuatro + seis, 4 = 98.431, 5 = 24])$$

Podemos modificar los elementos de una tabla de la misma forma que en el caso de las listas. Por ejemplo, modifiquemos el elemento indexado por **"2"**:

```
> t3[2] := junio;
```

$$t3_2 := junio$$

```
> eval(t3);
```

$$\text{table}([1 = x, 2 = junio, 3 = cuatro + seis, 4 = 98.431, 5 = 24])$$

Una forma de remover uno de los elementos es usando una ***"desasignación"***. Por ejemplo, eliminemos el elemento indexado por **"4"** en la tabla **t3**:

```
> t3[4] := 't3[4]';
```

$$t3_4 := t3_4$$

```
> eval(t3);
```

$$\text{table}([1 = x, 2 = junio, 3 = cuatro + seis, 5 = 24])$$

De esta manera podemos modificar una tabla previamente creada, eliminando o agregando elementos a los ya existentes.

9.2.5 Arreglos

Los arreglos son estructuras cuyos elementos se encuentran indexados de la misma forma que las tablas, pero en este caso los índices siempre son números enteros. Para este tipo de estructuras no es posible utilizar cadenas, u otro tipo de expresiones diferentes de números positivos, como índices. La forma en la cual se crean los arreglos es a través de la función **array**. Su sintaxis es:

array(dimensiones, elementos);

Donde dimensiones indica la cantidad de elementos que formarán el arreglo. Los elementos pueden o no proporcionarse en el momento de la creación del arreglo. Por ejemplo:

```
> arg1 := array(3..4, [4, Sistema]);
```

$$\begin{array}{l} arg1 := \text{array}(3..4, [\\ (3) = 4 \\ (4) = Sistema \\ ]) \end{array}$$

Esta instrucción crea un arreglo de dos datos, cuyos índices serán **"3"** y **"4"**, y sus elementos estarán dados por el número **"4"** y la cadena **"Sistema"**. Estos elementos pueden ser accesados utilizando sus índices:

```
> arg1[3];
```

$$4$$

```
> arg1[4];
```

$$Sistema$$

```
> arg1[2]; # obviamente este no es un elemento válido
Error, 1st index, 2, smaller than lower array bound 3
```

Nótese que, al igual que las tablas, los índices sirven como identificadores de los datos del arreglo, no necesariamente existe una relación entre el índice y la posición del elemento (como puede verse en la última instrucción). Veamos otro ejemplo:

```
> arg2 := array(2..5);
```

$$arg2 := \text{array}(2..5, [])$$

En este caso creamos un arreglo pero sin asignarle elementos, esta asignación podemos hacerla posteriormente:

```
> arg2[2] := `cadena 1`^2;
```

$$arg2_2 := cadena\ 1^2$$

Para visualizar los elementos de un arreglo es necesario utilizar la función **eval**.

```
> eval(arg2);
```

$$\begin{array}{l}\text{array}(2..5, [\\ (2) = \textit{cadena } 1^2\\ (3) = ?_3\\ (4) = ?_4\\ (5) = ?_5\\ ])\end{array}$$

Podemos ver que los últimos tres elementos aparecen indeterminados. Asignemos el resto de los datos:

```
> arg2[3] := sin(sqrt(2));
```

$$\textit{arg2}_3 := \sin(\sqrt{2})$$

```
> arg2[4] := 785;
```

$$\textit{arg2}_4 := 785$$

```
> arg2[5] := 87.32;
```

$$\textit{arg2}_5 := 87.32$$

```
> eval(arg2);
```

$$\begin{array}{l}\text{array}(2..5, [\\ (2) = \textit{cadena } 1^2\\ (3) = \sin(\sqrt{2})\\ (4) = 785\\ (5) = 87.32\\ ])\end{array}$$

En estas estructuras es posible desasignar un elemento pero éste no es eliminado del arreglo, solo su valor asignado. Por ejemplo:

```
> arg2[2] := 'arg2[2]';
```

$$\textit{arg2}_2 := \textit{arg2}_2$$

```
> eval(arg2);
```

$$\begin{array}{l}\text{array}(2..5, [\\ (2) = ?_2\\ (3) = \sin(\sqrt{2})\\ (4) = 785\\ (5) = 87.32\\ ])\end{array}$$

Al igual que en las tablas, si no se proporcionan índices, la función **array** automáticamente asigna a cada elemento un número entero que corresponde a su posición, iniciando en uno. Por ejemplo:

```
> arg3 := array([exp(f(x)), 8*x, 3 - cos(w), 8754]);
```

$$\textit{arg3} := \left[e^{f(x)}, 8\,x, 3 - \cos(w), 8754\right]$$

```
> arg3[1]; arg3[2]; arg3[3]; arg3[4];
```

$$e^{f(x)}$$

$$8\,x$$

$$3 - \cos(w)$$

$$8754$$

Los arreglos también pueden ser creados con índices multiples, de la misma forma que una matriz de varias dimensiones (de hecho pueden usarse arreglos para crear matrices). Por ejemplo:

```
> arg4 := array(2..4, 2..4, [[1, 9, 8] , [7, 4, 93], [28, 5, 18]]);
```

$$\begin{array}{l} arg4 := \text{array}(2..4,\ 2..4,\ [\\ (2,\ 2) = 1 \\ (2,\ 3) = 9 \\ (2,\ 4) = 8 \\ (3,\ 2) = 7 \\ (3,\ 3) = 4 \\ (3,\ 4) = 93 \\ (4,\ 2) = 28 \\ (4,\ 3) = 5 \\ (4,\ 4) = 18 \\ ]) \end{array}$$

En este caso, los elementos deben ser accesados por medio de dos índices.

```
> arg4[2, 2];
```

$$1$$

```
> arg4[3, 2];
```

$$7$$

```
> arg4[4, 4];
```

$$18$$

Este tipo de datos también pueden ser usados para crear matrices o vectores, de tal manera que puedan ser operados con matrices o vectores creados con la función **Matrix** o **Vector**, respectivamente, o bien por funciones del paquete **LinearAlgebra**. Para poder usarlos de esta forma debemos convertirlos previamente por medio de la instrucción **convert**. Por ejemplo:

```
> Ma := Matrix(3, 3, [[5, 4, 9], [6, 18, 2], [8, 23, 16]]);
```

$$Mat := \begin{bmatrix} 5 & 4 & 9 \\ 6 & 18 & 2 \\ 8 & 23 & 16 \end{bmatrix}$$

```
> Ar := array([[1, 2, 3], [Pi, 2, sin(Pi/2)], [-3, 2, 8]]);
```

$$Ar := \begin{bmatrix} 1 & 2 & 3 \\ \pi & 2 & 1 \\ -3 & 2 & 8 \end{bmatrix}$$

```
> Mb := convert(Ar, Matrix);
```

$$Mb := \begin{bmatrix} 1 & 2 & 3 \\ \pi & 2 & 1 \\ -3 & 2 & 8 \end{bmatrix}$$

```
> Ma.Mb;
```

$$Ma \,.\, \begin{bmatrix} 1 & 2 & 3 \\ \pi & 2 & 1 \\ -3 & 2 & 8 \end{bmatrix}$$

```
> Ma + Mb;
```

$$Ma + \begin{bmatrix} 1 & 2 & 3 \\ \pi & 2 & 1 \\ -3 & 2 & 8 \end{bmatrix}$$

Capítulo 10

Manipulación de expresiones algebraicas

Maple proporciona diversas funciones que nos permiten realizar manipulaciones sobre expresiones algebraicas, tales como simplificaciones, factorizaciones, expansión de expresiones con exponentes, agrupación de términos comunes, normalización de expresiones racionales, manipulación del numerador y denominador de una expresión racional; además de conversión de tipos de expresiones, entre otras. A lo largo de este capítulo describiremos varias de las funciones que nos permiten realizar este tipo de manipulaciones.

10.1 Simplificación

Al manipular una expresión, Maple lleva a cabo simplificaciones automáticas sobre ésta. El sistema fue diseñado para mantener las expresiones en la forma que fueron introducidas; sin embargo, en el caso de expresiones aritméticas, se llevan a cabo algunas simplificaciones sencillas en sumas, productos, cocientes y potencias de números enteros y racionales, números racionales expresados en forma fraccionaria, reducción de monomios semejantes y expresiones en las que puede aplicarse la propiedad asociativa, entre otras. Por ejemplo:

- Normalización de fracciones:

```
> -54/9*x^2 + 16/128*x + 32/4 = 0;
```

$$-6\,x^2+\frac{1}{8}\,x+8=0$$

- Expresiones con operaciones aritméticas y racionales:

```
> 2^3*x^3 - 4.2/8*x^2 + x/(3.5*4);
```

$$8\,x^3-0.5250000000\,x^2+0.07142857143\,x$$

- Agrupación de términos idénticos en sumas y productos:

```
> a + b + c + 2*a - 4*c;
```

$$3\,a+b-3\,c$$

- Reordenamiento de productos, colocando el factor constante al inicio de la expresión:

```
> a*b*3^2*x;
```

$$9\,a\,b\,x$$

- Eliminación de factores sintácticamente idénticos en el numerador y el denominador de fracciones algebraicas:

```
> (a*b)/(a*x^2*b);
```

$$\frac{1}{x^2}$$

- Eliminación de elementos repetidos en conjuntos:

```
> {x, a, b, a, f, g, x};
```

$$\{f, g, a, x, b\}$$

- Combinación de potencias de acuerdo a la regla $(x^r)^s = x^{(r\,s)}$, cuando r y s son racionales y además $-1 < r \leq 1$:

```
> (x^(3/5))^(2/3), (x^(1/3))^(-3/4);
```

$$x^{(2/5)}, \frac{1}{x^{(1/4)}}$$

- Distribución de potencias según la regla $(x\,y)^r = x^r\,y^r$, cuando r es un número racional, y x o y son positivos:

```
> (4*x)^(2/3), (3/x)^(2/3);
```

$$4^{(2/3)}\,x^{(2/3)}, 3^{(2/3)}\,(\frac{1}{x})^{(2/3)}$$

También se llevan a cabo otras simplificaciones, como producto de la evaluación de ciertas expresiones, por ejemplo:

```
> ln(1/2);
```

$$-\ln(2)$$

```
> cos(Pi/4);
```

$$\frac{\sqrt{2}}{2}$$

```
> abs(abs(x));
```

$$|x|$$

Cabe aclarar que, en estos últimos casos en realidad no se lleva a cabo una simplificación automática, ésta es resultado de la evaluación.

Cuando se usan expresiones más complicadas, las reglas de simplificación automática no las reducen a su forma más simple. En estos casos es necesario solicitar explícitamente la simplificación; esto puede hacerse por medio dela función **simplify**; su sintaxis es:

simplify(expr);

simplify(expr, regla1, regla2, ..., reglan);

Donde **expr** es la expresión a simplificar. También se pueden especificar, opcionalmente, las reglas de simplificación que deseamos aplicar a la expresión; aunque esta instrucción puede aplicar reglas de simplificación apropiadas para cada tipo de expresión, de tal forma que generalmente obtenemos una forma más simple. Por ejemplo, consideremos los siguientes ejemplos:

```
> 9^(1/2) - 3;
```

$$\sqrt{9} - 3$$

```
> 9^n/(3^n*3^n) - 1;
```

$$\frac{9^n}{(3^n)^2} - 1$$

Nótese que en ambos casos no se realiza la simplificación. Intentemos usando **simplify**:

```
> simplify(9^(1/2) - 3);
```

$$0$$

```
> simplify(9^n/(3^n*3^n) - 1);
```

$$0$$

Veamos otro ejemplo:

```
> F1 := exp(-ln(x) + x)*(x^2 + 2*sin(x)^2 + 2*cos(x)^2 - 3*x - 2);
```

$$F1 := e^{(-\ln(x)+x)}\,(x^2 + 2\sin(x)^2 + 2\cos(x)^2 - 3\,x - 2)$$

```
> simplify(F1);
```

$$e^x\,(x - 3)$$

Algunas de las opciones soportadas por simplify nos permiten aplicas reglas específicas, por ejemplo cuando las expresiones involucran funciones trigonométricas, logaritmos o radicales; como veremos a continuación.

10.1.1 Simplificación de expresiones con funciones trigonométricas

Para llevar a cabo este tipo de simplificaciones debemos incluir la opción **trig** como argumento de la función **simplify**.

```
> sin(x)^3;
```

$$\sin(x)^3$$

```
> simplify(%, trig);
```

$$\sin(x) - \sin(x)\cos(x)^2$$

```
> 1 + tan(x)^2, simplify(%, trig);
```

$$1 + \tan(x)^2,\ \sin(x) - \sin(x)\cos(x)^2$$

```
> cos(2*x) + sin(x)^2 = simplify(cos(2*x) + sin(x)^2, trig);
```

$$\cos(2\,x) + \sin(x)^2 = \cos(x)^2$$

10.1.2 Simplificación de expresiones con las opciones radical y symbolic

Consideremos la siguiente expresión:

```
> e := [(x^3 + 3*x^2*a + 3*x*a^2 + a^3)^(1/3) + (8*y)^(1/3),
> (y^3)^(1/2) + (-27)^(1/3)];
```

$$e := [(x^3 + 3\,x^2\,a + 3\,x\,a^2 + a^3)^{(1/3)} + 8^{(1/3)}\,y^{(1/3)},\ \sqrt{y^3} + (-27)^{(1/3)}]$$

Nótese que Maple no considera automáticamente **sqrt(x^2)** igual a **x**, pues ésta última podría ser negativa o compleja. Para forzar la evaluación a **x** (asumiendo que es un real positivo) necesitamos incluir las opciones **radical** y **symbolic**.

```
> simplify(e, radical);
```

$$[((a + x)^3)^{(1/3)} + 2\,y^{(1/3)},\ \sqrt{y^3} + \frac{3}{2} + \frac{3}{2}\,I\,\sqrt{3}]$$

```
> simplify(e, radical, symbolic);
```

$$[a + x + 2\,y^{(1/3)},\ y^{(3/2)} + \frac{3}{2} + \frac{3}{2}\,I\sqrt{3}\,]$$

Otra forma en la que se puede forzar la simplificación en este tipo de expresiones es indicando a Maple que suponga una cierta propiedad para las incógnitas. Por ejemplo, consideremos la siguiente expresión:

```
> f := (-8*n^8*c)^(1/4);
```

$$f := (-8\,n^8\,c)^{(1/4)}$$

Primero intentemos simplificarla solo con **simplify**:

```
> simplify(f);
```

$$2^{(3/4)}\,(-n^8\,c)^{(1/4)}$$

Puede notarse que solo se aplica la regla de simplificación para radicales sobre **"8"** y sobre **"(-n^3*c)"**, sin embargo este último producto no puede ser simplificado pues no se sabe nada acerca de n y de c. Volvamos a simplificar, indicandole a **simplify** que asuma las incognitas como reales, esto podemos hacerlo por medio de la opción **assume=real**:

```
> simplify(f, assume=real);
```

$$n^2\,2^{(3/4)}\,(-c)^{(1/4)}$$

Otra opción más que puede aplicarse es **symbolic**, con la cual **simplify** considera las variables como simbólicas:

```
> simplify(f, symbolic);
```

$$(1 + I)\,n^2\,2^{(1/4)}\,c^{(1/4)}$$

También podemos usar la función **assume** para indicar que la constante **c** debe ser considerada, por ejemplo, mayor que cero:

```
> assume(c > 0);
> simplify(f);
```

$$2^{(3/4)}\,c^{\sim(1/4)}\,(-n^8)^{(1/4)}$$

10.1.3 Simplificación de acuerdo a reglas definidas por el usuario

Otra forma de llevar a cabo una simplificación es especificando una igualdad para que se aplique al hacer esta operación. Por ejemplo, consideremos la siguiente relación:

```
> rel_lateral := {sin(x)^2 + cos(x)^2 = 1};
```

$$rel\_lateral := \{\sin(x)^2 + \cos(x)^2 = 1\}$$

Simplificaremos la siguiente expresión, usando para ello **rel_lateral**:

```
> expr1 := sin(x)^3 - 11*sin(x)^2*cos(x) + 3*cos(x)^3 - sin(x)*cos(x) + 2;
```

$$expr1 := \sin(x)^3 - 11\sin(x)^2\cos(x) + 3\cos(x)^3 - \sin(x)\cos(x) + 2$$

```
> simplify(expr1, rel_lateral);
```

$$14\cos(x)^3 - \sin(x)\cos(x) + 2 - \sin(x)\cos(x)^2 + \sin(x) - 11\cos(x)$$

También es posible especificar varias relaciones para que la función **simplify** las aplique al realizar la simplificación, veamos un ejemplo:

```
> rels := {z^3 - z^2 - z*y + 2*y^2 = 1, z^3 + y^2 = 1,
> z^2 + z*y - y^2 = 0, x + y = z};
```

$$rels := \{x + y = z,\ z^2 + z\,y - y^2 = 0,\ z^3 + y^2 = 1,\ z^3 - z^2 - z\,y + 2\,y^2 = 1\}$$

```
> h1 := 36*z^4*y^2 + 36*z*y^4 - 36*z*y^2 - 1/2*z^2 + z*y - 1/2*y^2 +
> 1/2*x*z - 1/2*x*y + 2/3*z^4 + 4/3*z^3*y - 2/3*z^2*y^2 - 4/3*z*y^3 +
> 2/3*y^4;
```

$$h1 := 36\,z^4\,y^2 + 36\,z\,y^4 - 36\,z\,y^2 - \frac{1}{2}\,z^2 + z\,y - \frac{1}{2}\,y^2 + \frac{1}{2}\,x\,z - \frac{1}{2}\,x\,y + \frac{2}{3}\,z^4 + \frac{4}{3}\,z^3\,y - \frac{2}{3}\,z^2\,y^2 - \frac{4}{3}\,z\,y^3 + \frac{2}{3}\,y^4$$

```
> simplify(h1, rels);
```

$$0$$

10.1.4 Simplificación de expresiones con radicales anidados

La función **simplify** puede simplificar expresiones con radicales simples, pero no puede hacerlo cuando los radicales se encuentran anidados, por ejemplo:

```
> s := sqrt(2*(3 - sqrt(2) - sqrt(3) + sqrt(6)));
```

$$s := \sqrt{6 - 2\sqrt{2} - 2\sqrt{3} + 2\sqrt{6}}$$

```
> simplify(s);
```

$$\sqrt{6 - 2\sqrt{2} - 2\sqrt{3} + 2\sqrt{2}\sqrt{3}}$$

Este tipo de expresiones deben ser simplificadas usando la función **radnormal**, de la siguiente forma:

```
> radnormal(s);
```

$$-1 + \sqrt{2} + \sqrt{3}$$

10.2 Factorización

Maple permite llevar a cabo factorizaciones por medio de la función **factor**. Por ejemplo:

```
> factor(2*x^5 + 9*x^4 - 5*x^3 - 49*x^2 - 57*x - 20);
```

$$(x + 4)\,(2\,x - 5)\,(x + 1)^3$$

```
> factor(x^3 - x/3);
```

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

Por omisión, este función lleva a cabo la factorización de acuerdo al tipo de coeficientes que aparecen en el polinomio, a menos que se indique otra cosa. Por ejemplo:

```
> factor(2*I*x^3 - 14*I*x^2 + 80*I*x + 4*I);
```

$$2\,I\,(x^3 - 7\,x^2 + 40\,x + 2)$$

Podemos indicar que la factorización se lleve a cabo en terminos de números complejos, de la siguiente forma:

```
> factor(x^4 + b^4, I);
```

$$(x^2 + b^2\,I)\,(x^2 - b^2\,I)$$

Esta función también puede ser aplicada a integrales, en este caso se lleva a cabo la factorización sobre el integrando:

```
> factor(Int(x^2 + 2*b*x + b^2, x));
```

$$\int (x+b)^2\,dx$$

También es posible llevar a cabo factorizaciones en expresiones que involucran varias incógnitas:

```
> factor(x^2*cos(y)^5 - x^2*cos(y)^2 + 2*x^2*cos(y)^3 - 2*x^2);
```

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

10.3 Expansión de expresiones

La función ***expand*** nos permite expandir expresiones en las que aparecen exponentes y productos de polinomios. Esta expansión se lleva a cabo principalmente distribuyendo productos sobre sumas, aunque también tiene la capacidad para desarrollar expresiones en las que aparecen funciones trigonométricas, logaritmos, binomiales, entre muchas otras. Por ejemplo:

```
> expand((x + 1)^2*(x - 1));
```

$$x^3 + x^2 - x - 1$$

```
> expand([sin(2*x), sin(3*x)*cos(x)]);
```

$$[\,2\sin(x)\cos(x),\ 4\sin(x)\cos(x)^3 - \sin(x)\cos(x)\,]$$

Podemos indicar a **expand** que no opere algún término, incluyendo este al final de la instrucción (lo mismo puede hacerse para varios términos, escribiéndolos en forma de secuencia):

```
> expand((1 - x)*(1 - 2*x)/(x*(1 + y)*(1 + 2*x)), 1 + y);
```

$$\frac{1}{x\,(1+y)\,(1+2\,x)} - \frac{3}{(1+y)\,(1+2\,x)} + \frac{2\,x}{(1+y)\,(1+2\,x)}$$

10.4 Agrupación de términos

Por otro lado, la función **combine** nos permite agrupar términos dentro de una expresión. Dependiendo del tipo de ésta y de los términos que se desea combinar, existen diferentes opciones aplicables.

Por ejemplo, para agrupar términos en una expresión trigonométrica usamos la opción **trig**:

```
> combine(2*sin(x)*cos(x), trig);
```

$$\sin(2\,x)$$

Otras opciones aplicables son: **log**, **exp** y **power**, las cuales pueden ser combinadas, como puede apreciarse en los siguientes ejemplos:

```
> combine(2*ln(x^2 - 1) - ln(x^2 + 1), ln);
```

$$2\ln(x^2-1) - \ln(x^2+1)$$

```
> combine((x^b)^2 - sqrt(3)^(1/5), power);
```

$$x^{(2\,b)} - 3^{(1/10)}$$

```
> combine([2*sin(x)*cos(x), 2*cos(x)^2 - 1], trig);
```

$$[\sin(2x), \cos(2x)]$$

```
> combine(exp(sin(x)*cos(b))*exp(cos(x)*sin(b)), [trig, exp]);
```

$$e^{\sin(x+b)}$$

```
> Integral := Int(exp(sin(x)*cos(b))*exp(cos(x)*sin(b)), x);
```

$$Integral := \int e^{(\sin(x)\cos(b))}\, e^{(\cos(x)\sin(b))}\, dx$$

```
> combine(Integral, [trig, exp]);
```

$$\int e^{\sin(x+b)}\, dx$$

10.5 Normalización

Podemos simplificar expresiones compuestas por fracciones algebraicas por medio de la función ***normal***. Al normalizar una fracción algebraica, ésta se expresa en forma de una fracción irreducible ***p/q***. Por ejemplo:

```
> normal((x^2 - y^2)/(x - y)^3);
```

$$\frac{x+y}{(x-y)^2}$$

```
> expr := sin((x*(x + 1) - x)/(x + 2))^2 + cos(x^2*(x + 2)/(x^2 - 4))^2;
```

$$expr := \sin\left(\frac{x\,(x+1)-x}{x+2}\right)^2 + \cos\left(\frac{x^2\,(x+2)}{x^2-4}\right)^2$$

```
> normal(expr);
```

$$\sin\left(\frac{x^2}{x+2}\right)^2 + \cos\left(\frac{x^2}{x-2}\right)^2$$

Podemos incluir la opción **expanded** para que el resultado de la normalización sea desplegado en forma expandida:

```
> normal((x^4 - y^4)/(x - y)^3, expanded);
```

$$\frac{x^3 + y\,x^2 + y^2\,x + y^3}{x^2 - 2\,x\,y + y^2}$$

No es recomendable usar esta función si el numerador o denominador se complican al ser factorizados, por ejemplo:

```
> normal((x^10 - 1)/(x - 1));
```

$$x^9 + x^8 + x^7 + x^6 + x^5 + x^4 + x^3 + x^2 + x + 1$$

10.6 Agrupación de términos respecto a una variable

No siempre es posible factorizar o simplificar una expresión, en algunos casos solo se pueden agrupar términos comunes. Para ejemplificar esto consideremos la siguiente expresión:

```
> expr := x^3*y^2 + x^3*y*b + x^3*y*c + x^3*b - x^3*c + x^2*y*a -
> x^2*y*b + x^2*c - x*y^2*a - x*y^2*b + x*y*c - x*y*d - x*y - x*a -
> x*b + y*a + y*b - 1;
```

$$expr := x^3 y^2 + x^3 y b + x^3 y c + x^3 b - x^3 c + x^2 y a - x^2 y b + x^2 c - x y^2 a - x y^2 b + x y c - x y d - x y - x a - x b + y a + y b - 1$$

Intentaremos simplificarla y factorizarla:

```
> simplify(expr);
```

$$x^3 y^2 + x^3 y b + x^3 y c + x^3 b - x^3 c + x^2 y a - x^2 y b + x^2 c - x y^2 a - x y^2 b + x y c - x y d - x y - x a - x b + y a + y b - 1$$

```
> factor(expr);
```

$$x^3 y^2 + x^3 y b + x^3 y c + x^3 b - x^3 c + x^2 y a - x^2 y b + x^2 c - x y^2 a - x y^2 b + x y c - x y d - x y - x a - x b + y a + y b - 1$$

Como puede apreciarse, no obtenemos una expresión más simple. En este caso podemos usar la función ***collect*** para agrupar términos e intentar obtener una forma más sencilla. Primero intentaremos transformar esta expresión en un polinomio en términos de **x**:

```
> collect(expr, x);
```

$$(y^2 + y b + y c + b - c) x^3 + (y a - y b + c) x^2 + (-b - a - y^2 b + y c - y d - y^2 a - y) x - 1 + y a + y b$$

También podemos agrupar en términos de **x**, de tal forma que cada coeficiente sea un polinomio en términos de **y**:

```
> collect(expr, [x, y]);
```

$$(y^2 + (b + c) y - c + b) x^3 + ((a - b) y + c) x^2 + ((-b - a) y^2 + (-1 + c - d) y - b - a) x - 1 + (a + b) y$$

Otra posible opción es expresar el polinomio en términos de **y**:

```
> collect(expr, y);
```

$$(-x b - x a + x^3) y^2 + (x^3 c - x^2 b - x + x^2 a + x c - x d + b + x^3 b + a) y - x a - x b + x^3 b - x^3 c - 1 + x^2 c$$

Por medio de **collect** podemos aplicar varias opciones a la expresión que estamos manipulando, por ejemplo podemos factorizar cada coeficiente:

```
> collect(expr, x, factor);
```

$$(y^2 + y b + y c + b - c) x^3 + (y a - y b + c) x^2 + (-b - a - y^2 b + y c - y d - y^2 a - y) x - 1 + y a + y b$$

También podemos aplicar funciones a los coeficientes; por ejemplo, aplicaremos la función **sin** a cada uno de los coeficientes de **expr**:

```
> collect(expr, x, sin);
```

$$\sin(y^2 + y b + y c + b - c) x^3 + \sin(y a - y b + c) x^2 - \sin(b + a + y^2 b - y c + y d + y^2 a + y) x + \sin(-1 + y a + y b)$$

Otra posible opción es agrupar términos con respecto a dos o más variables:

```
> collect(expr, [y, x]);
```

$$(x^3+(-b-a)\,x)\,y^2+((b+c)\,x^3+(a-b)\,x^2+(-1+c-d)\,x+a+b)\,y-1+(-c+b)\,x^3 \\ +x^2\,c+(-b-a)\,x$$

Veamos otro ejemplo, con la siguiente expresión:

```
> expr2 := ((y^3 + (b + c)*y^2 + (b - c)*y)/(y^2 - (a + b)*y))*x^3 +
> ((a - b)*y^2 + c*y)*x^2/(y^2 + a*y) - ((a + b)*y^3 - (c - d - 1)*y^2 +
> (a + b)*y)*x/(y^2 + (a + b)*y) - 1;
```

$$expr2 := \frac{(y^3+(b+c)\,y^2+(-c+b)\,y)\,x^3}{y^2-(a+b)\,y}+\frac{((a-b)\,y^2+y\,c)\,x^2}{y^2+y\,a} \\ -\frac{((a+b)\,y^3-(-1+c-d)\,y^2+(a+b)\,y)\,x}{y^2+(a+b)\,y}-1$$

Si aplicamos **simplify**, lejos de simplificarse, se complica la expresión:

```
> simplify(expr2);
```

$$-(x^2\,y^3\,a-x^2\,y\,a^3-x^2\,y^3\,b+x^2\,y\,b^3+x^2\,y^2\,c+x^3\,y\,c\,a\,b+x^3\,y\,b^2+x^3\,b\,a^2+x^3\,b^2\,a \\ -x^3\,c\,a^2+x^3\,y^2\,b^2+x^3\,y^3\,c-x^3\,y^2\,c+2\,x^3\,y^3\,a+2\,x^3\,y^3\,b+x^3\,y^2\,a^2+x^3\,y^2\,b \\ +y\,b^2+2\,b\,a^2+b^2\,a+x^3\,y^4+y\,a^2+2\,x^3\,y^2\,c\,a+x^3\,y^2\,c\,b-x^3\,y\,c\,b-x^3\,c\,a\,b \\ +2\,x^3\,y\,b\,a-2\,x^3\,y\,c\,a+x^3\,y\,b\,a^2+x^3\,y\,b^2\,a+x^3\,y\,c\,a^2+3\,x^3\,y^2\,a\,b-x^2\,y\,b\,a^2 \\ +x^2\,y\,b^2\,a-2\,x^2\,c\,a\,b+2\,y\,b\,a+x\,y\,d\,b\,a-x\,y\,c\,a\,b+x\,a^3-x\,y^3+x\,y\,b^2 \\ +2\,x\,b\,a^2+x\,b^2\,a+x\,y\,a^2+x\,y^3\,c-x\,y^4\,b+x\,y^3\,b^2-x\,y^3\,d-x\,y^4\,a+x\,y^2\,a^3 \\ -x\,y^2\,c\,b+2\,x\,y\,b\,a-x\,y\,c\,a^2+x\,y\,d\,a^2+x\,y^2\,d\,b+x\,y^3\,b\,a+x\,y^2\,b^2\,a \\ +2\,x\,y^2\,a^2\,b-x^2\,c\,a^2-x^2\,c\,b^2+a^3-x\,y^2\,a-y^2\,a-y^3)/((-y+a+b)\,(y+a) \\ (y+a+b))$$

Intentaremos simplificarla usando diferentes variantes de la función *collect*:

```
> collect(expr2, x);
```

$$\frac{(y^3+(b+c)\,y^2+(-c+b)\,y)\,x^3}{y^2-(a+b)\,y}+\frac{((a-b)\,y^2+y\,c)\,x^2}{y^2+y\,a} \\ -\frac{((a+b)\,y^3-(-1+c-d)\,y^2+(a+b)\,y)\,x}{y^2+(a+b)\,y}-1$$

```
> collect(expr2, [x, y]);
```

$$-1+\frac{(y^3+(b+c)\,y^2+(-c+b)\,y)\,x^3}{y^2+(-b-a)\,y}+\frac{((a-b)\,y^2+y\,c)\,x^2}{y^2+y\,a} \\ -\frac{((a+b)\,y^3+(1-c+d)\,y^2+(a+b)\,y)\,x}{y^2+(a+b)\,y}$$

```
> collect(expr2, [x, y], normal);
```

$$-1+\frac{(y^3+(b+c)\,y^2+(-c+b)\,y)\,x^3}{y^2+(-b-a)\,y}+\frac{((a-b)\,y^2+y\,c)\,x^2}{y^2+y\,a} \\ -\frac{((a+b)\,y^3+(1-c+d)\,y^2+(a+b)\,y)\,x}{y^2+(a+b)\,y}$$

```
> collect(expr2, [x, y], factor);
```

$$-1+\frac{(y^3+(b+c)\,y^2+(-c+b)\,y)\,x^3}{y^2+(-b-a)\,y}+\frac{((a-b)\,y^2+y\,c)\,x^2}{y^2+y\,a}$$
$$-\frac{((a+b)\,y^3+(1-c+d)\,y^2+(a+b)\,y)\,x}{y^2+(a+b)\,y}$$

```
> collect(expr2, [y, x], factor);
```

$$-1-\frac{(y^2+y\,b+y\,c+b-c)\,x^3}{-y+a+b}+\frac{(y\,a-y\,b+c)\,x^2}{y+a}-\frac{(b+a+y^2\,b-y\,c+y\,d+y^2\,a+y)\,x}{y+a+b}$$

10.7 Manipulación del numerador y denominador de una expresión racional

Las funciones **numer** y **denom** nos permiten extraer el numerador y denominador, respectivamente, de una expresión racional. Por ejemplo:

```
> expr_rac := (2*x - sqrt(x + 1) - (1/2)*x^3 - x*sin(2*x))/(1 + 2*x +
> sqrt(x + 1) - (1/2)*x^3 + x^2*sin(2*x));
```

$$expr\_rac := \frac{2\,x-\sqrt{x+1}-\frac{x^3}{2}-x\sin(2\,x)}{1+2\,x+\sqrt{x+1}-\frac{x^3}{2}+x^2\sin(2\,x)}$$

```
> numer(expr_rac);
```

$$4\,x-2\,\sqrt{x+1}-x^3-2\,x\sin(2\,x)$$

```
> denom(expr_rac);
```

$$2+4\,x+2\,\sqrt{x+1}-x^3+2\,x^2\sin(2\,x)$$

Nótese que la expresión es normalizada antes de extraer el numerador y denominador. Comparese **expr_rac** con la expresión normalizada:

```
> normal(expr_rac);
```

$$-\frac{-4\,x+2\,\sqrt{x+1}+x^3+2\,x\sin(2\,x)}{2+4\,x+2\,\sqrt{x+1}-x^3+2\,x^2\sin(2\,x)}$$

Una vez extraídos el numerador y denominador, podemos aplicar diferentes simplificadores a cada parte:

```
> factor(numer(expr_rac)) / factor(denom(expr_rac));
```

$$\frac{4\,x-2\,\sqrt{x+1}-x^3-2\,x\sin(2\,x)}{2+4\,x+2\,\sqrt{x+1}-x^3+2\,x^2\sin(2\,x)}$$

```
> collect(numer(expr_rac), x) / factor(denom(expr_rac));
```

$$\frac{-x^3+(4-2\sin(2\,x))\,x-2\,\sqrt{x+1}}{2+4\,x+2\,\sqrt{x+1}-x^3+2\,x^2\sin(2\,x)}$$

10.8 Extracción de los coeficientes de un polinomio

Maple nos permite extraer los coeficientes de un polinomio por medio de las funciones **coeff** y **coeffs**. Consideremos la siguiente expresión:

```
> pol := x^3*y^2 + x^3*y*b + x^3*y*c + x^3*b - x^3*c + x^2*y*a - x^2*y*b +
> x^2*c - x*y^2*a - x*y^2*b + x*y*c - x*y*d - x*y - x*a - x*b + y*a +
> y*b - 1;
```

$$pol := x^3 y^2 + x^3 y b + x^3 y c + x^3 b - x^3 c + x^2 y a - x^2 y b + x^2 c - x y^2 a - x y^2 b + x y c - x y d - x y - x a - x b + y a + y b - 1$$

Podemos expresar este polinomio en términos de x por medio de **collect**:

```
> collect(pol, x);
```

$$(y^2 + y b + y c + b - c) x^3 + (y a - y b + c) x^2 + (-b - a - y^2 b + y c - y d - y^2 a - y) x - 1 + y a + y b$$

También podemos extraer cada uno de los coeficientes del polinomio, especificando la variable principal y la potencia de la cual deseamos el coeficiente:

```
> coeff(pol, x, 3);
```

$$y^2 + y b + y c + b - c$$

```
> coeff(pol, x, 2);
```

$$y a - y b + c$$

```
> coeff(pol, x, 1);
```

$$-b - a - y^2 b + y c - y d - y^2 a - y$$

```
> coeff(pol, x, 0);
```

$$-1 + y a + y b$$

Podemos extraer todos los coeficientes, en forma de secuencia, usando la función **coeffs** de la siguiente forma:

```
> coeffs(pol, x);
```

$$-1 + y a + y b,\ -b - a - y^2 b + y c - y d - y^2 a - y,\ y a - y b + c,\ y^2 + y b + y c + b - c$$

En los siguientes ejemplos, extraeremos los coeficientes de **pol** respecto a **y**. Nótese que automáticamente se agrupan los términos con respecto a la variable especificada, antes de extraer los coeficientes. Compárese con la salida de **collect**:

```
> coeff(pol, y, 2);
```

$$-x b - x a + x^3$$

```
> coeffs(pol, y);
```

$$-x a - x b + x^3 b - x^3 c - 1 + x^2 c,\ -x b - x a + x^3,\ x^3 c - x^2 b - x + x^2 a + x c - x d + b + x^3 b + a$$

```
> collect(pol, y);
```

$$(-x b - x a + x^3) y^2 + (x^3 c - x^2 b - x + x^2 a + x c - x d + b + x^3 b + a) y - x a - x b + x^3 b - x^3 c - 1 + x^2 c$$

10.9 Ordenamiento de términos

La función **sort** no permite ordenar los elementos de una expresión. Esta función también puede ser aplicada a listas de expresiones. Por ejemplo:

```
> sort(1 + x^2 - x);
```

$$x^2 - x + 1$$

Por omisión las expresiones son ordenadas en forma descendente, aunque se puede especificar de que manera se debe realizar el ordenamiento. Por ejemplo, ordenaremos los elementos de la siguiente expresión, en el primer caso solo con respecto a **x**, en el segundo caso con respecto a **x** y con respecto a **y**; y finalmente con respecto a **y** y con respecto a **x**:

```
> poli := x^4*y + 12*x^5*y^2 + 40*x^6*y^4 - 32*x^4*y^3 - 368*x^4*y -
> 320*x^3*y + 768*x + 1024;
```

$$poli := -367\,x^4 y + 12\,x^5 y^2 + 40\,x^6 y^4 - 32\,x^4 y^3 - 320\,x^3 y + 768\,x + 1024$$

```
> sort(poli, x);
```

$$40\,y^4 x^6 + 12\,y^2 x^5 - 367\,y\,x^4 - 32\,y^3 x^4 - 320\,y\,x^3 + 768\,x + 1024$$

```
> sort(poli, [x, y]);
```

$$40\,x^6 y^4 + 12\,x^5 y^2 - 32\,x^4 y^3 - 367\,x^4 y - 320\,x^3 y + 768\,x + 1024$$

```
> sort(poli, [y, x]);
```

$$40\,y^4 x^6 - 32\,y^3 x^4 + 12\,y^2 x^5 - 367\,y\,x^4 - 320\,y\,x^3 + 768\,x + 1024$$

Podemos especificar el orden en que se llevará a cabo el ordenamiento, para ello se pueden incluir como opciones '<' o '>', para un orden ascendente o descendente, respectivamente. Por ejemplo:

```
> sort([-2, 4, 8, 5, -4, 10, 25], `>`);
```

$$[25, 10, 8, 5, 4, -2, -4]$$

```
> sort([-2, 4, 8, 5, -4, 10, 25], `<`);
```

$$[-4, -2, 4, 5, 8, 10, 25]$$

También podemos expecificar la variable y el orden al mismo tiempo:

```
> sort(poli, [x, `>`]);
```

$$40\,y^4 x^6 + 12\,y^2 x^5 - 32\,y^3 x^4 - 367\,y\,x^4 - 320\,y\,x^3 + 768\,x + 1024$$

10.10 Conversión de expresiones

Otra de las facilidades proporcionadas por Maple es la conversión de expresiones, la cual puede llevarse a cabo por medio de la función **convert**. Ésta nos permite convertir entre diferentes tipos, tales como:

- Convertir series en polinomios:

```
> s1 := taylor(sin(x), x=0);
```

$$s1 := x - \frac{1}{6}x^3 + \frac{1}{120}x^5 + O(x^6)$$

```
> convert(s1, polynom);
```

$$x - \frac{1}{6}x^3 + \frac{1}{120}x^5$$

- Convertir series en expresiones racionales:

```
> s2 := taylor(exp(x), x=0);
```

$$s2 := 1 + x + \frac{1}{2}x^2 + \frac{1}{6}x^3 + \frac{1}{24}x^4 + \frac{1}{120}x^5 + O(x^6)$$

```
> convert(s2, ratpoly);
```

$$\frac{1 + \frac{3}{5}x + \frac{3}{20}x^2 + \frac{1}{60}x^3}{1 - \frac{2}{5}x + \frac{1}{20}x^2}$$

- Convertir una expresión racional en fracciones parciales, con respecto a **x**:

```
> g := (x^3 - 3*x + 1)*(x^4 - 5*x^3) / (2*x^4 + 4*x^2 - 5*x);
```

$$g := \frac{(x^3 - 3x + 1)(x^4 - 5x^3)}{2x^4 + 4x^2 - 5x}$$

```
> convert(g, parfrac, x);
```

$$\frac{x^3}{2} - \frac{5x^2}{2} - \frac{5x}{2} + \frac{57}{4} + \frac{-30x^2 - 278x + 285}{4(2x^3 + 4x - 5)}$$

- Conversión de senos y tangentes a logaritmos o exponenciales:

```
> convert(sinh(x) + sin(x), exp);
```

$$\frac{1}{2}e^x - \frac{1}{2}\frac{1}{e^x} - \frac{1}{2}I\left(e^{(x\,I)} - \frac{1}{e^{(x\,I)}}\right)$$

```
> convert(arctanh(x), ln);
```

$$\frac{1}{2}\ln(x + 1) - \frac{1}{2}\ln(1 - x)$$

- Conversión de listas y conjuntos en sumas y productos....

```
> S := seq(x[i]^2, i=1..10);
```

$$S := x_1^2, x_2^2, x_3^2, x_4^2, x_5^2, x_6^2, x_7^2, x_8^2, x_9^2, x_{10}^2$$

```
> convert([S], `+`);
```

$$x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 + x_6^2 + x_7^2 + x_8^2 + x_9^2 + x_{10}^2$$

```
> convert({S}, `*`);
```

$$x_1^2 x_2^2 x_3^2 x_4^2 x_5^2 x_6^2 x_7^2 x_8^2 x_9^2 x_{10}^2$$

Con esta misma función se pueden llevar a cabo otras conversiones más simples, por ejemplo:

- Conversiones de radianes a grados y viceversa:

```
> convert(Pi/2, degrees);
```

$$90\ degrees$$

```
> convert(120*degrees, radians);
```

$$\frac{2\pi}{3}$$

- Conversión de números entre diferentes bases. Por ejemplo:

  - Convertir de decimal a binario, a hexadecimal y a octal:

```
> convert(34, binary);
                              100010

> convert(1234, hex);
                               4D2

> convert(235, octal);
                               353
```

  - Convertir a decimal un número en cualquier base entre 2 y 36. La sintaxis en este caso es:

    convert(número, decimal, base)

    Donde **base** es un número entre 2 y 36, o bien una de las palabras: **binary**, **octal** o **hex**, para binario, octal o hexadecimal, respectivamente. En bases mayores que 10 se pueden usar las letras a (o A) a la z (o Z), para representar los dígitos 10 a 35, respectivamente. Por ejemplo:

```
> convert("1A.C", decimal, hex);
                            26.75000000

> convert(110101, decimal, 2);
                               53

> convert(GH23A, decimal, 20);
                            2696870
```

  - Convertir a octal o hexadecimal un número en cualquier base entre 2 y 36. Este caso es análogo al anterior. Por ejemplo:

```
> convert(3212, octal, decimal);
                              6214

> convert(6214, hex, octal);
                              1846

> convert(1846, decimal, hex);
                              6214

> convert(23483, hex, 9);
                              5BBB
```

Los siguientes ejemplos muestran otro tipo de conversiones válidas:

- Convertir números de punto flotante en fracciones:

```
> convert(.3456, fraction);
                              216
                              ---
                              625
```

- Convertir listas en matrices:

```
> l := [[1, 2, 3], [4, 5, 6], [7, 8, 9]];
```

$$l := [[1, 2, 3], [4, 5, 6], [7, 8, 9]]$$

```
> convert(l, Matrix, 3, 3);
```

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

- Convertir enteros en bytes y cadenas en enteros:

```
> convert([65, 66, 67], 'bytes');
```

"ABC"

```
> convert(`Hola`, bytes);
```

[72, 111, 108, 97]

Esta función permite realizar una gran cantidad de conversiones de tipos; por razones de espacio solo se han presentado algunos ejemplos. Véase la página de ayuda **?convert** para obtener una lista completa.

Capítulo 11

Gráficas en dos dimensiones

Las funciones soportadas por Maple pueden ser divididas en varios grupos principales. Uno de estos grupos lo constituyen las funciones para despliegue de gráficas y animaciones. Este sistema proporciona varias instrucciones con capacidad para desplegar gráficas de diferentes tipos de funciones; por ejemplo, pueden generar gráficas de funciones explícitas, implícitas, paramétricas, gráficas de puntos, tanto en dos como en tres dimensiones. En este capítulo se presentan las principales instrucciones proporcionadas para despliegue de gráficas en dos dimensiones, algunas de las diferentes opciones que permiten modificar el aspecto de éstas, así como la manera de manipular las regiones correspondientes. También se describe la nueva función **interactive** del paquete **plots**, la cual nos facilita la creación de una gráfica a través de la nueva herramienta gráfica **Interactive Plot Builder** incluida en Maple 8, la cual describiremos más adelante.

11.1 Gráficas de funciones explícitas

Maple nos permite desplegar gráficas de funciones explícitas de una variable por medio de la función **plot**. Su sintaxis es la siguiente:

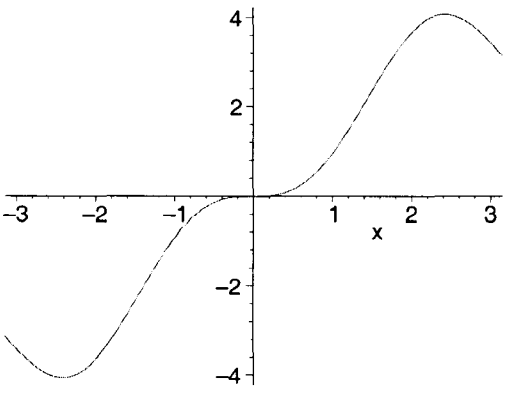
plot(f, r, ops)

Donde **f** es la función o expresión que se desea graficar (de una variable), **r** es el rango en el cual se desea desplegar dicha función, y **ops** son las opciones que se aplicarán a la gráfica. Veamos un ejemplo:

```
> f := x -> x + sin(x) - sin(2*x);
```

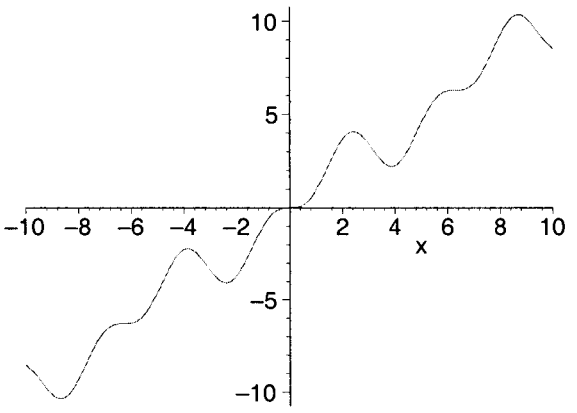
$$f := x \rightarrow x + \sin(x) - \sin(2\,x)$$

```
> plot(f(x), x=-Pi..Pi);
```



Nótese que el rango es expresado en la forma *"x=a..b"*, donde *"a"* y *"b"* son los extremos del rango en el cual se desea el despliegue y además $a < b$. Este rango puede omitirse, colocando simplemente la variable de la cual depende la función o expresión.

```
> plot(f(x), x);
```



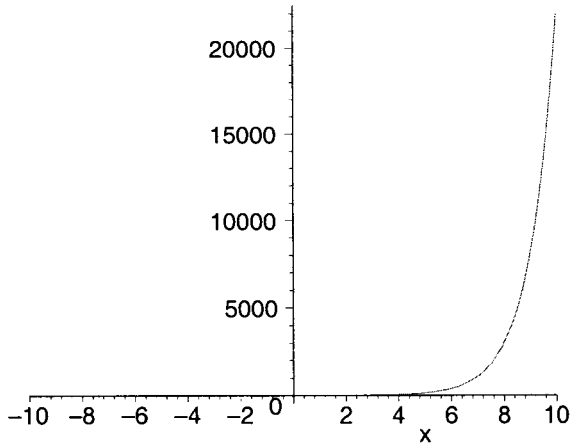
En este caso, el rango tomado por omisión es **"x = -10..10"**.

Cuando se expresa un rango para el eje x, el rango del eje y queda determinado por los valores que toma la función al evaluarla en los diferentes puntos de la variable. Esto puede provocar que las gráficas aparezcan deformadas o que su aspecto sea muy diferente del que realmente tiene la función dada, lo cual puede impedir apreciar el verdadero comportamiento de ésta. Por ejemplo:

```
> g := x -> sin(x^3)*exp(exp(cos(x))) + exp(x);
```

$$g := x \to \sin(x^3)\, e^{(e^{\cos(x)})} + e^x$$

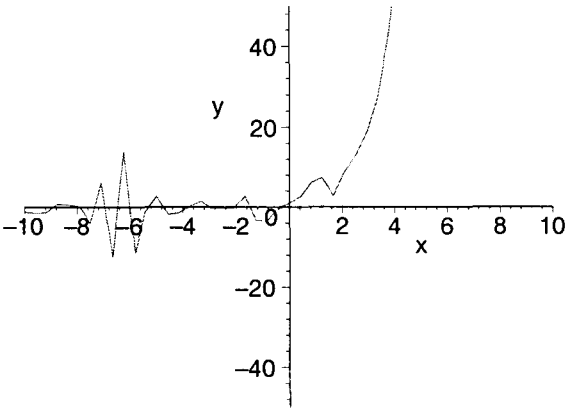
```
> plot(g(x), x=-10..10);
```



La función **plot** nos permite indicar explicitamente un rango para el eje y, con lo cual es posible restringir el despliegue a lo largo de este eje. Este rango puede expresarse como *"y=c..d"*, o bien simplemente como *"c..d"*, sin colocar la variable y.

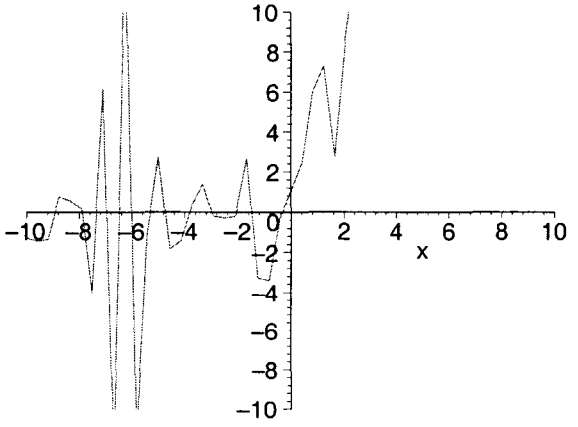
Apliquemos esto último a la gráfica anterior:

```
> plot(g(x), x=-10..10, y=-50..50);
```



Esta es una forma en la cual puede apreciarse mejor el verdadero aspecto de la gráfica; incluso, si restringimos aun más el rango para el eje y, obtenemos la siguiente gráfica más detallada:

```
> plot(g(x), x=-10..10, -10..10);
```



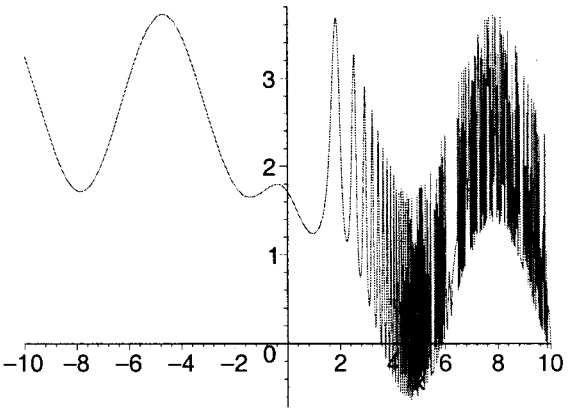
Otra forma de apreciar el aspecto real de la gráfica es utilizando el botón 11, para deplegarla a escala normal (este botón aparece en el menú contextual al seleccionar una región gráfica). Es importante señalar que, de forma predeterminada, las gráficas aparecen con una cierta distorsión, esto se debe a que son desplegadas dentro de una región cuadrada y por lo tanto al generarlas Maple las ajusta para que se adapten a esta región en la cual se mostraran al usuario.

También debe tomarse en cuenta que en algunas gráficas es necesario restringir no el eje y, sino el eje x para obtener un despliegue más fiel de la función. Por ejemplo:

```
> k := x -> sin(x) + exp(cos(exp(x)));
```

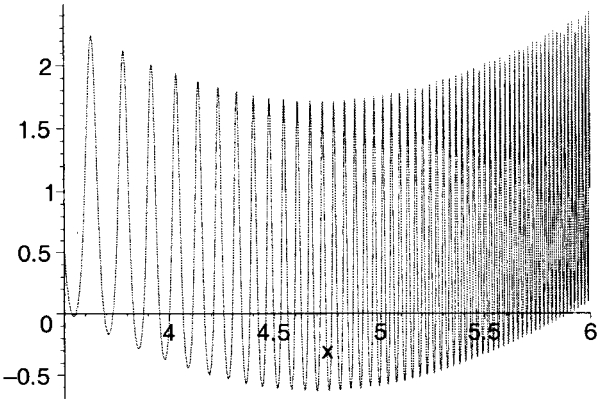
$$k := x \to \sin(x) + e^{\cos(e^x)}$$

```
> plot(k(x), x=-10..10);
```



Si restringimos el rango en x a **"3.5 .. 6"** podemos apreciar mejor el comportamiento de esta función en este intervalo.

```
> plot(k(x), x=3.5..6);
```



En general, para desplegar una gráfica, utilizando por ejemplo **plot(h(x), x=a..b)**, Maple procede a grandes rasgos de la siguiente forma:

- Primero, toma el intervalo **"x = a..b"** y lo particiona en un conjunto de puntos (el valor predeterminado es 50). A esta partición se le conoce con el nombre de ***"malla"***.
- A continuación, se evalua la función **h(x)** en cada uno de los puntos de la malla.
- Con estos datos se forman un conjunto de parejas ordenadas de la forma:

  $[x_i,\ \mathrm{f}(x_i)]$, donde x_i es un punto de la malla.

- A continuación se crea una estructura que contiene los puntos obtenidos, el color con el que se desplegará la gráfica, el título de ésta (en caso de que se desee colocar alguno), el tipo de los ejes, el ancho y estilo de la línea utilizada, etc.

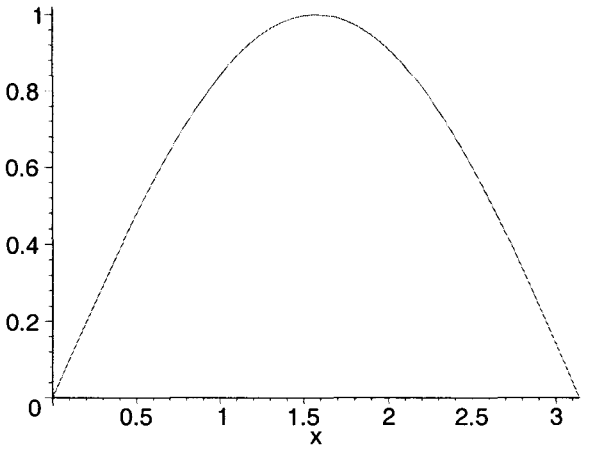
- Finalmente, Maple toma esta estructura y la despliega en la pantalla, uniendo los puntos contenidos en dicha estructura con el tipo de línea adecuada y colocando el resto de los elementos especificados, como los ejes, títulos y demás.

Esta estructura, generada por Maple para crear una gráfica, puede ser asignada a una variable para su despliegue o manipulación posterior. Veamos el siguiente ejemplo:

```
> g1 := plot(sin(x), x=0..Pi):
```

Posteriomente podemos desplegar esta gráfica con la instrucción **display** del paquete **plots**, o simplemente invocandola en la línea de comandos.

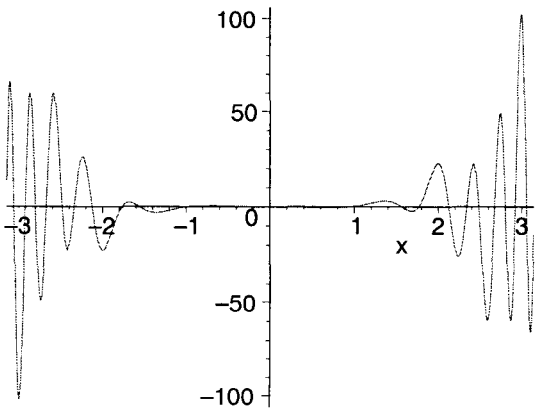
```
> plots[display](g1);
```



Nota: cuando se asigna una gráfica a una variable es recomendable usar como terminador ":", de lo contrario se desplegará toda la estructura generada para el despliegue.

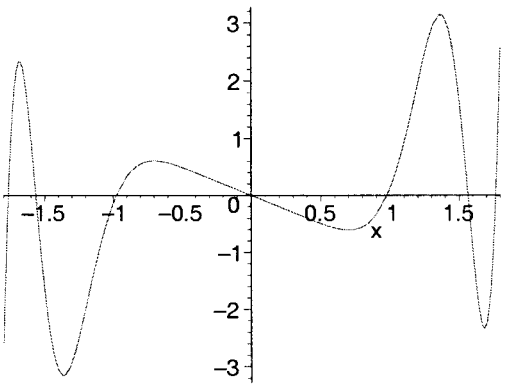
Veamos el siguiente ejemplo:

```
> plot(x^4*sin(x^3) - x^3*cos(x^2) + x^2*sin(x) - x, x=-Pi..Pi);
```



Nótese que en esta gráfica tampoco se aprecia el comportamiento de la función alrededor de $x = 0$. Podemos restringir nuevamente el rango en este eje para poder apreciar mejor la parte que no es clara:

```
> plot(x^4*sin(x^3) - x^3*cos(x^2) + x^2*sin(x) - x, x=-1.8..1.8);
```



Generalmente las gráficas generadas por Maple son desplegadas en la misma hoja de trabajo, esto puede modificarse de tal forma que sean mostradas en una ventana independiente; para ello solicitamos la opción **Preferences** del menú **File**; la cual desplegará la ventana que se muestra a continuación. en ella solicitamos la ficha **Plotting** y en la región **Plot Display** seleccionamos **Window** (esta opción se encuentra colocada en **Inline** de manera predeterminada), como se muestra en la Figura 11.1.

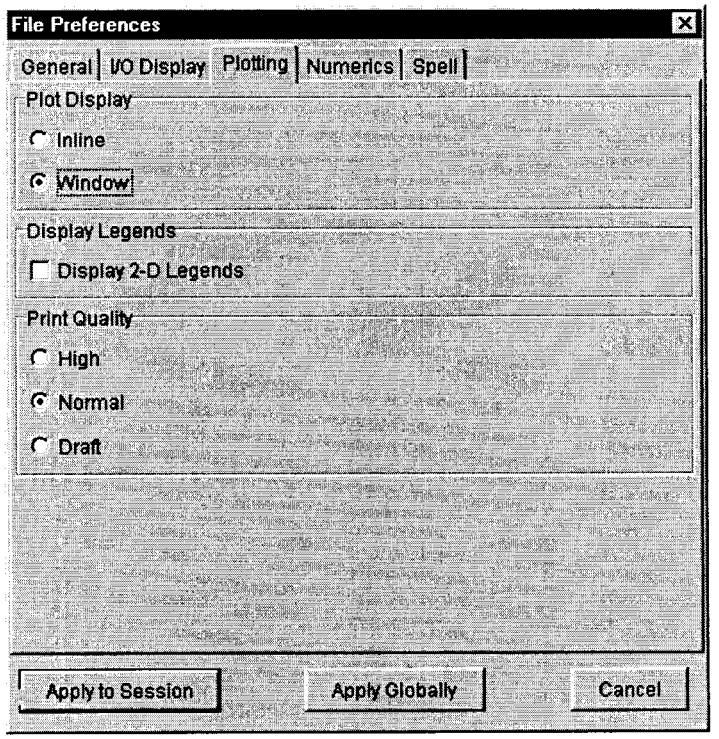


Figura 11.1: Ventana de preferencias

A una gráfica en dos dimensiones se le puede asignar una leyenda, la cual se despliega en la parte inferior de la misma. Esta leyenda puede asignarse seleccionando la gráfica y solicitando la opción **Edit Leyend** en el menú **Leyend**. En la ventana mostrada en la Figura 11.1, en la región **Display Legends**, se puede indicar a Maple si estas leyendas deben o no mostrarse, el mismo resultado puede obtenerse si seleccionamos la gráfica y solicitamos la opción **Show Leyend** en el menú **Leyend**. Otra manera en la que podemos hacer estas operaciones es oprimiendo el botón derecho del ratón sobre la gráfica, esto desplegará un menú contextual, en él seleccionamos el submenú **Leyend**.

En esta misma ventana, en la región **Print Quality**, podemos establecer la calidad con la que deben imprimirse las gráficas cuando imprimimos el documento.

11.2 Principales opciones aplicables a gráficas en dos dimensiones

Existe un conjunto de opciones que pueden aplicarse tanto a **plot** como a la mayoria de las instrucciones que generan gráficas en dos dimensiones, con el fin de obtener mejores despliegues. A continuación describiremos algunas de ellas.

11.2.1 Restricción del rango vertical

Esta opción fue tratada en parte en la sección anterior, es bastante útil al desplegar funciones cuya gráfica se dispara verticalmente, ya que nos permite ***"recortar"*** dicha gráfica de tal forma que podamos apreciar mejor su comportamiento en un área menor a la mostrada de manera predeterminada.

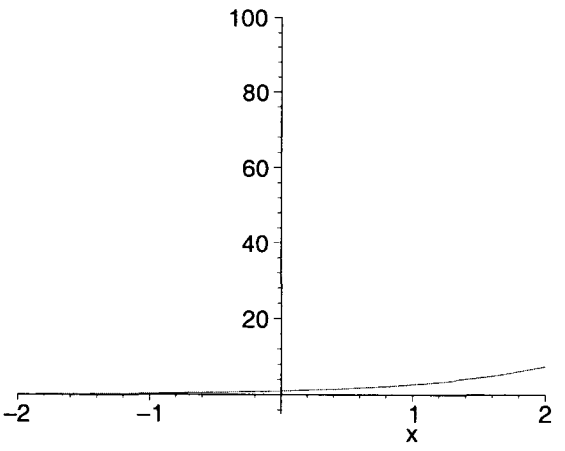
Además de lo visto anteriormente, **plot** proporciona la opción **view** que nos permite especificar los rangos horizontal y vertical para el despliegue. Su sintaxis es:

plot(f, x, view=[x1..x2, y1..y2])

Los rangos proporcinados a través de esta opción determinan el área del plano que se deberá mostrar en la gráfica.

Veamos el siguiente ejemplo:

```
> plot(exp(x), x, view=[-2..2, -5..100]);
```



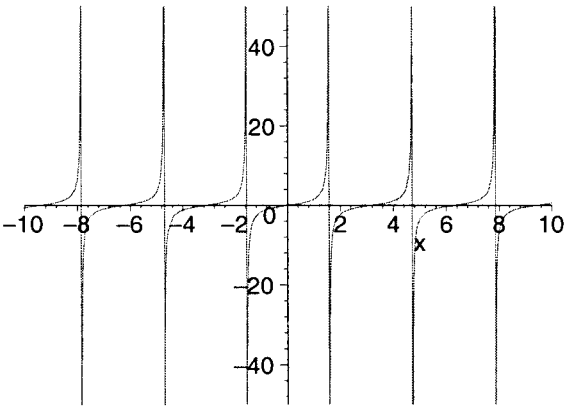
Los valores predeterminados que toma esta opción son **[-10 .. 10, fmin .. fmax]**. Donde **fmin** y **fmax** son los valores mínimo y máximo, respectivamente, obtenidos al evaluar la función en los puntos del primer rango.

11.2.2 Desplegar solo las secciones en las cuales la función es continua

Esto puede hacerse utilizando la opción **discont=true**, con la cual Maple despliega la gráfica solo en los intervalos del rango en los cuales la función es continua. Si no se incluye esta opción, las discontinuidades pueden aparecer en forma de líneas verticales.

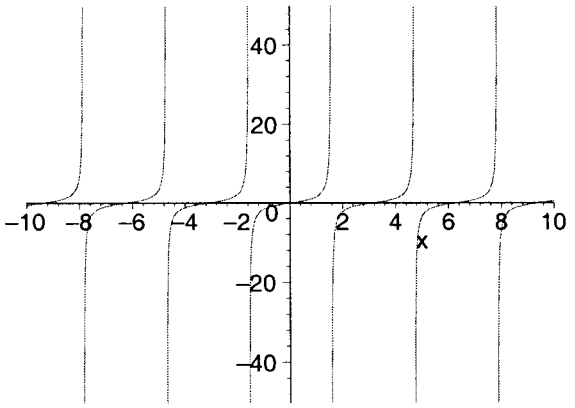
Para ejemplificar esto veamos el siguiente caso:

```
> plot(tan(x), x=-10..10, -50..50);
```



En esta gráfica se muestran algunas líneas verticales en los puntos donde existen discontinuidades. Agregaremos la opción **discont=true**:

```
> plot(tan(x), x=-10..10, -50..50, discont=true);
```



Nótese como en esta última gráfica no se despliegan las lineas verticales, las cuales corresponden precisamente a los puntos en los cuales la función es discontinua. Para desplegar esta gráfica, al incluir la opción antes mencionada, Maple utiliza la instrucción **discont**, la cual determina las discontinuidades de una función dada, y a continuación divide el intervalo a graficar en un conjunto de subintervalos en los cuales la función es continua.

La sintaxis de la instrucción **discont** es:

discont(f,x)

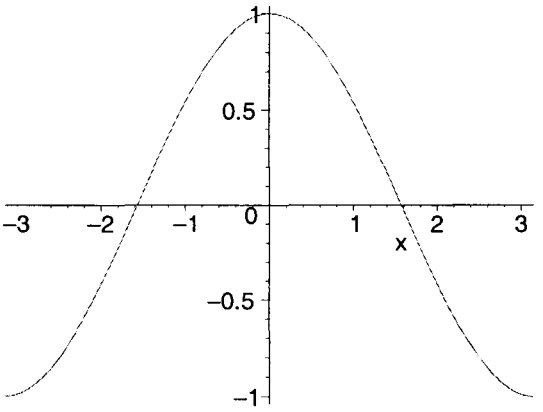
donde **f** es una función (o expresión) y x es la variable de la cual depende. Consúltese su página de ayuda.

11.2.3 Colocar diferentes tipos de ejes

La opción **axes** nos permite colocar diferentes tipos de ejes en el despliegue de una gráfica en dos dimensiones. Los valores que puede tomar esta opción son:

- **normal** Este es el valor predeterminado. Muestra los ejes de la siguiente forma:

```
> plot(cos(x), x=-Pi..Pi, axes=normal);
```



Esta opción también puede aplicarse directamente del menú contextual para regiones gráficas. Al seleccionar una de estas regiones, en este menú aparecen una serie de botones que permiten modificar el aspecto de las gráficas, uno de ellos es el botón  , el cual (en el caso de las gráficas en dos dimensiones) despliega los ejes de la misma forma que al aplicar la opción **axes=normal**. Otra forma en la que puede aplicarse es colocando el ratón sobre la gráfica y oprimiendo el botón derecho, en el menú contextual que aparece seleccionamos el submenú **Axes**, en el cual podemos seleccionar los mismos tipos de ejes que se describen en esta sección.

A continuación se muestran otras opciones aplicables:

- **frame**. Despliega el eje x en el borde inferior de la gráfica y el eje y en el borde izquierdo. Esta opción también puede ser aplicada directamente desde el menú contextual para regiones gráficas, a través del botón .

- **boxed**. Muestra los ejes en forma de un marco que contiene a la gráfica. Se puede aplicar también mediante el botón  , de la barra contextual.

- **none**. Suprime el despliegue de los ejes en la gráfica. También puede ser aplicada por medio del botón  , de la barra contextual.

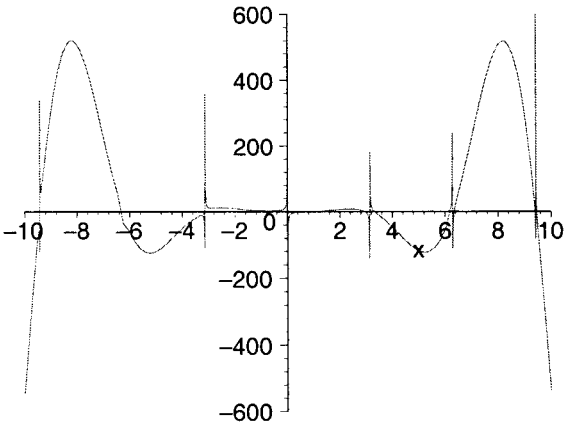
11.2.4 Modificación de la malla

Las funciones para despliegue de gráficas en dos dimensiones soportan dos opciones que nos permiten modificar la malla utilizada, estas son **numpoints** y **resolution**. La primera nos permite especificar el número mínimo de puntos que formarán la malla (el valor predeterminado es 50) y la segunda nos permite especificar el número máximo (por omisión 200).

Al generar la estructura de una gráfica, Maple evalua la función en los puntos de la malla y después los une con una linea para presentar el despliegue. Pero, ¿qué sucede si la función tiene una discontinuidad en un punto que no está considerado en la malla?

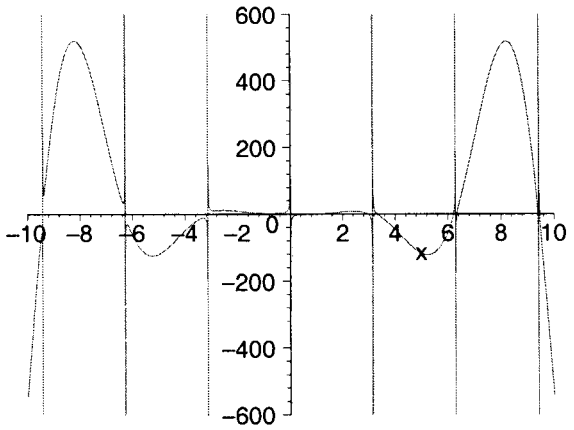
La función **plot** (entre otras) tiene la capacidad de utilizar más puntos en la malla en las regiones en las cuales se encuentran las discontinuidades. Sin embargo, en funciones que oscilan demasiado esto no es suficiente, en estos casos es conveniente incrementar el número de puntos en la malla. Por ejemplo:

```
> plot(sin(x)*x^3 - 1/sin(Pi - x) + 1, x=-10..10, -600..600);
```



Ahora incrementaremos el número de puntos de la malla a 400:

```
> plot(sin(x)*x^3 - 1/sin(Pi - x) + 1, x=-10..10, -600..600,
> numpoints=400);
```



Al utilizar una malla más fina la gráfica aparece más detallada, incluso pueden apreciarse discontinuidades que anteriormente no se mostraban.

11.2.5 Asignación de colores a la gráfica

Por omisión Maple despliega las gráficas en color rojo (excepto cuando se despliegan varias funciones), pero puede utilizarse la opción ***"color = c"*** para especificar uno diferente. Esta constante puede asumir los valores predefinidos que se muestran en la tabla 11.1.

aquamarine	cyan	khaky	pink	turquoise
black	gold	magenta	plum	violet
blue	green	maroon	red	wheat
brown	gray	navy	sienna	white
coral	grey	orange	tan	yellow

Table 11.1: Colores predefinidos para gráficas

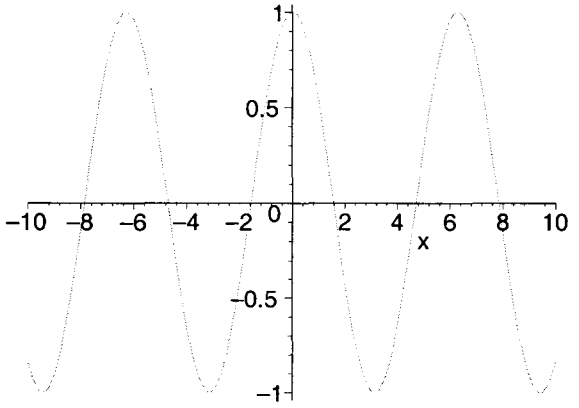
También es posible expresar el color en formato ***RGB***, utilizando la función:

COLOR(RGB, r, v, a)

donde ***"r"***, ***"v"*** y ***"a"*** especifican las componentes rojo, verde y azul, respectivamente, del color deseado (estos componentes solo pueden tomar valores entre cero y uno).

Por ejemplo veamos la siguiente gráfica:

```
> plot(cos(x), x, color=COLOR(RGB, .737, .737, 0));
```



Una forma más conveniente de definir colores utilizando el modo **RGB** es a través de la definición de ***"macros"***. Por ejemplo:

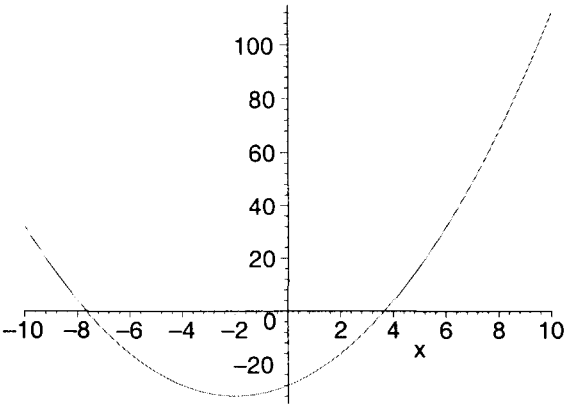
macro(c1=COLOR(RGB, C-R, C-G, C-B))

Esta instrucción define una macro de nombre **c1**, la cual representa el color dado por las componentes RGB recibidas como segundo, tercero y cuarto argumentos. Este nuevo color creado puede ser usado de la siguiente forma, en el despliegue de una gráfica:

```
> macro(c1 = COLOR(RGB, .437, .437, 0));
```

c1

```
> plot(x^2 + 4*x - 28, x, color=c1);
```



Otras opciones útiles, aplicables a este tipo de gráficas incluyen:

- **Colocar etiquetas en los ejes.** Esto puede hacerse a través de la opción **labels=[etqx, etqy]**. Donde **"etqx"** y **"etqy"** son las etiquetas para el eje x y el eje **y** respectivamente. Por ejemplo, la siguiente opción:

  labels=[velocidad, tiempo]

  desplegará la etiqueta **"velocidad"** en el eje x y **"tiempo"** en el eje y. Estas etiquetas deben estar dadas en forma de cadenas en caso de que contengan espacios en blanco. Puede utilizarse también la opción **labelfont=[f, e, t]**, para especificar un tipo de fuente, estilo y tamaño de las etiquetas de los ejes (véase la siguiente opción para una descripción de los argumentos **f**, **e** y t).

- **Utilizar diferentes tipos de fuentes.** Maple nos permite utilizar varios fuentes, estilos y tamaños en los textos desplegados en una gráfica. Esto puede aplicarse con la opción **font=[f, e, t]**. Donde **"f"** es el tipo de fuente, el cual puede ser **TIMES**, **COURIER**, **HELVETICA** o **SYMBOL**; **"e"** determina el estilo, el cual varía dependiendo de la fuente. Para **TIMES** los estilos disponibles son **ROMAN**, **BOLD**, **ITALIC** y **BOLDITALIC**; para **HELVETICA** y **COURIER** el estilo puede ser **BOLD**, **OBLIQUE** o **BOLDOBLIQUE**, o bien puede ser omitido. La fuente **SYMBOL** no acepta ningún tipo de estilo. Finalmente el argumento **"t"** indica el tamaño de la fuente en puntos. Por ejemplo, si se aplican las opciones:

  labels=[velocidad, 'eje y'], font=[TIMES, ITALIC, 14]

  a una gráfica en dos dimensiones, se desplegarán las etiquetas **"velocidad"** y **"eje y"**, utilizando la fuente **TIMES**, en estilo **ITALIC** y con un tamaño de 14 puntos.

- **Modificar la fuente para los ejes.** Podemos especificar el tipo de fuente a ser utilizado para los números que se despliegan a lo largo de los ejes, a través de la opción **axesfont=[f, e, t]**. Donde **"f"**, **"e"** y **"t"** son como en la opción **font**.

- **Colocar un título a la gráfica.** Esto puede hacerse a través de la opción **title=ct**, donde **"ct"** es la cadena de caracteres que se colocará como título. Además, podemos incluir también la opción **titlefont=[f, e, t]**, para determinar el tipo de fuente, estilo y tamaño con el que se desplegará este título. Los argumentos de **titlefont** son como en la opción **font**.

- **Utilizar líneas de diferente grosor y estilo.** El grosor de las líneas puede ser dado a través de la opción **thickness=g**, donde **"g"** indica el grosor y puede tomar valores enteros entre cero y tres. El valor predeterminado es cero, el cual corresponde al tipo de linea más delgada utilizada en el despliegue de este tipo de gráficas, el grosor de la línea aumenta conforme al valor de **"g"**.

  Otra opción aplicable es **linestyle=n**, donde **"n"** indica el estilo de la linea. Los valores cero y uno generan lineas continuas, mientras que los valores más grandes generan lineas punteadas de diferentes formas.

Todas las opciones descritas anteriormente pueden ser aplicadas a cualquier tipo de gráfica en dos dimensiones, tanto las generadas por **plot** como las generadas por otras funciones.

Para obtener una lista completa de las opciones aplicabes a funciones en dos dimensiones consúltese la hoja de ayuda **?plot[options]**, o bien **?plot,options**.

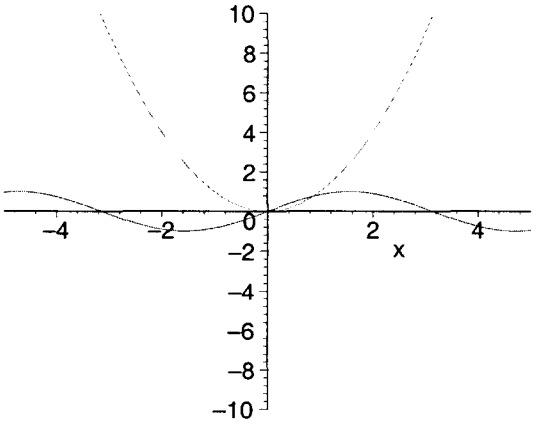
11.3 Despliegue de multiples funciones explícitas

Maple nos permite desplegar las gráficas de varias funciones en un mismo despliegue, colocando éstas como argumento en forma de conjunto, es decir:

plot({f1, f2, f3,...}, x=a..b, opciones)

Veamos el siguiente ejemplo:

```
> plot({sin(x), x^2, exp(sqrt(abs(x)))}, x=-5..5, -10..10);
```



Nótese que en este caso Maple asigna automáticamente un color diferente a cada gráfica. Además, no es necesario que dichas funciones dependan de la misma variable al definirlas. Por ejemplo:

```
> f1 := x -> sin(x);
```

$$f1 := \sin$$

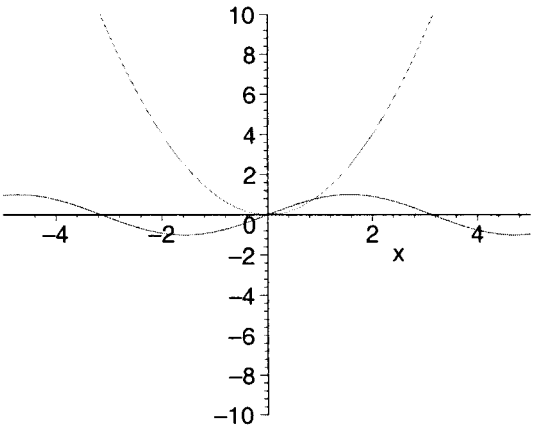
```
> f2 := r -> r^2;
```

$$f2 := r \to r^2$$

```
> f3 := s -> exp(sqrt(abs(s)));
```

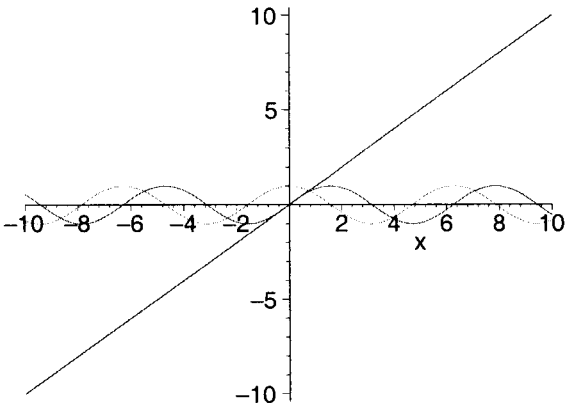
$$f3 := s \to e^{\sqrt{|s|}}$$

```
> plot({f1(x), f2(x), f3(x)}, x=-5..5, -10..10);
```



Cuando se despliegan varias funciones en una misma gráfica, se puede asignar un color a cada función mediante la opción **color**, de la siguiente forma:

```
> plot({sin(x), cos(x), x}, x, color=[blue, brown, green]);
```



Otra manera de graficar varias funciones en un mismo despliegue es por medio de la función **display**, esto se abordará más adelante.

11.4 Gráficas de puntos

La instrucción **plot**, además de funciones, también nos permite generar gráficas de puntos. Éstos deben estar dados en forma de una lista de parejas, de la siguiente manera:

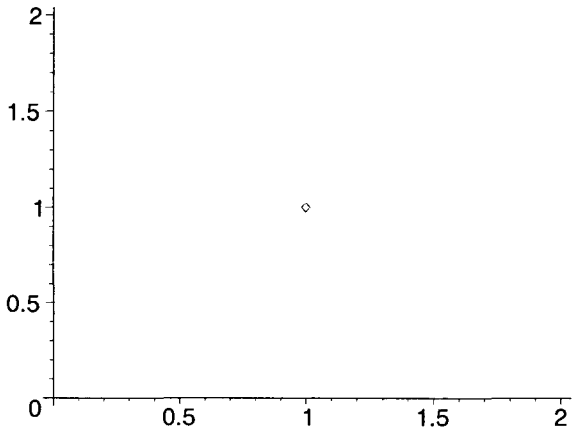
plot([[x1, y1], [x2, y2], [x3, y3], ...])

Esta función toma cada uno de los puntos en la lista y genera una gráfica uniendolos con una línea. Para que estos puntos puedan apreciarse es necesario incluir la opciones **style=point**, lo cual indica a Maple que se trata de una gráfica de puntos. También se puede incluir **symbol=ts**, para especificar el tipo de símbolo que se usará para mostrar cada punto, y la opción ***symbolsize=n*** que permite especificar el tamaño del símbolo (por omisión 10). Como tipo de símbolo puede usarse:

BOX, CROSS, CIRCLE, POINT, DIAMOND

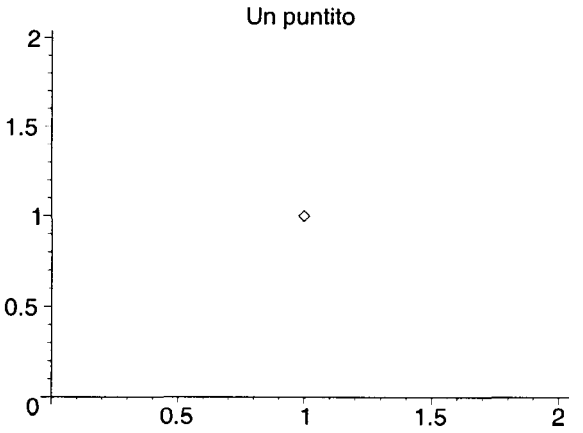
Por ejemplo:

```
> plot([[1, 1]], style=point, symbol=DIAMOND, symbolsize=15);
```



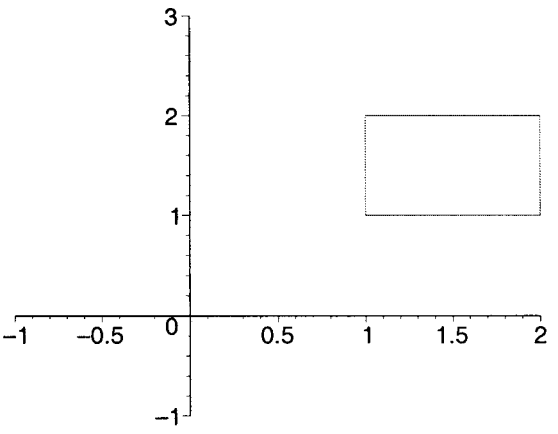
A estas gráficas podemos aplicarles cualquiera de las opciones descritas anteriormente. Por ejemplo :

```
> plot([[1, 1]], style=point, symbol=DIAMOND, symbolsize=18,
> color=blue, title='Un puntito');
```



Veamos ahora la siguiente gráfica.

```
> plot([[1, 1], [1, 2], [2, 2], [2, 1], [1, 1]], view=[-1..2,
> -1..3]);
```



En este caso, al colocar los puntos en forma de una lista estos son unidos entre sí por medio de una línea (nótese que no estamos incluyendo la opción **style=point**).

11.5 Funciones paramétricas

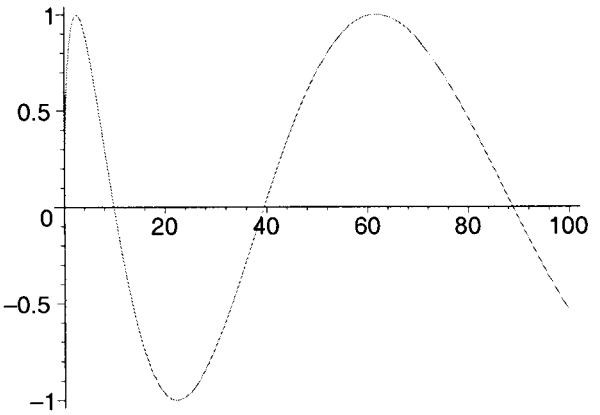
Otra de las capacidades de la función **plot** es que nos permite desplegar gráficas de funciones dadas en forma paramétrica. La sintaxis es:

plot([r(x), t(x), rango], opciones)

Donde **"r(x)"** y **"t(x)"** son las componentes paramétricas de la función a graficar. A este tipo de gráficas se les pueden aplicar todas las opciones vistas previamente.

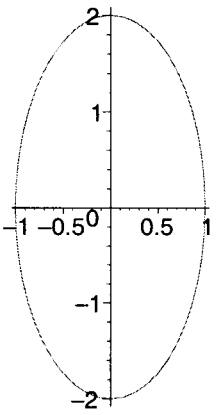
Veamos un ejemplo:

```
> plot([t^2, sin(t), t=0..10], color=magenta);
```



De la misma manera podemos desplegar otras gráficas:

```
> plot([sin(t), 2*cos(t), t=0..2*Pi], scaling=constrained);
```



Una forma en la que podemos graficar una función dada en la forma **"y=f(x)"**, es por medio de la parametrización **"[x, f(x)]"**.

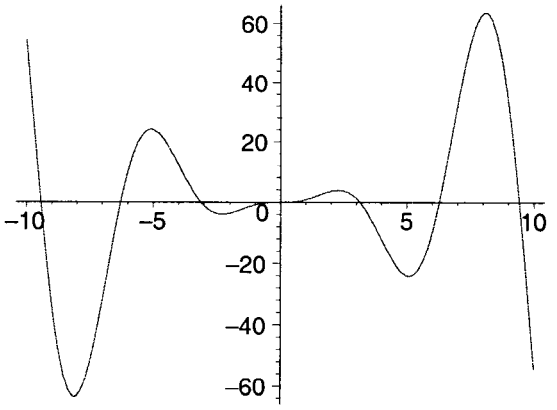
En el siguiente ejemplo deplegaremos la gráfica de " $\sin(x)\,x^2$", parametrizada como:

"$[x, \sin(x)\,x^2]$"

```
> f := x -> sin(x)*x^2;
```

$$f := x \to \sin(x)\,x^2$$

```
> plot([x, f(x), x=-10..10], color=blue);
```



11.6 Coordenadas polares

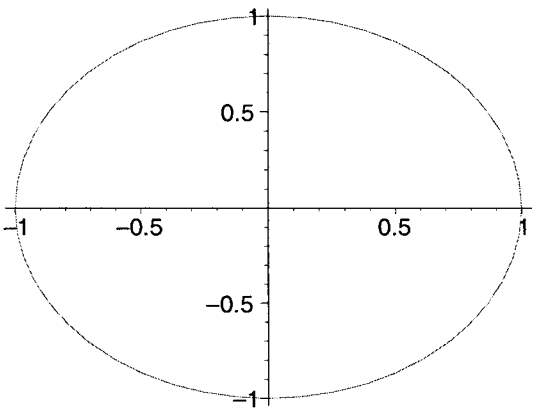
En Maple, las gráficas en coordenadas polares son manejadas como gráficas de funciones paramétricas, a las cuales se les agrega la opción **coords=polar**.

Su sintaxis es la misma que en el caso de las funciones paramétricas, es decir:

plot([r(t), theta(t), rango], coords=polar, otras_opciones)

En este caso "**r(t)**" representa el radio y "**theta(t)**" el ángulo de la función. Por ejemplo:

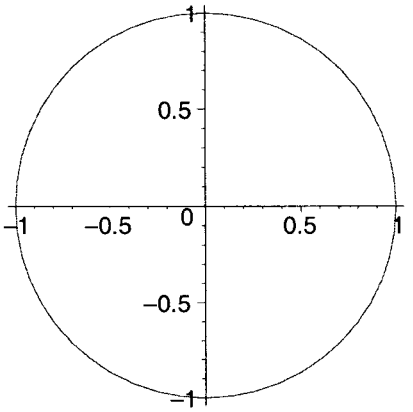
```
> plot([1, t, t=0..2*Pi], coords=polar);
```



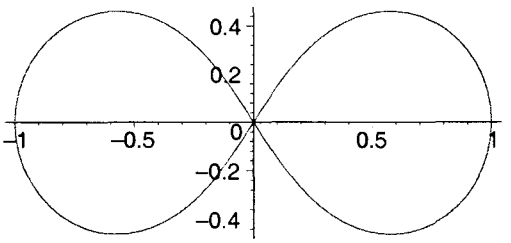
Esta gráfica corresponde a un círculo de radio *"r(t)=1"*.

Para poder apreciar el efecto de esta opción, comparense las siguientes gráficas:

```
> plot([sin(t), cos(t), t=0..2*Pi], scaling=constrained,
> color=blue);
```



```
> plot([sin(t), cos(t), t=0..2*Pi], coords=polar, scaling=constrained,
> color=blue);
```



11.7 Gráficas en dos dimensiones con el paquete Plots

Este es uno de los paquetes más robustos de Maple, proporciona un conjunto de funciones especializadas para la generación de diversos tipos de gráficas, como veremos a continuación.

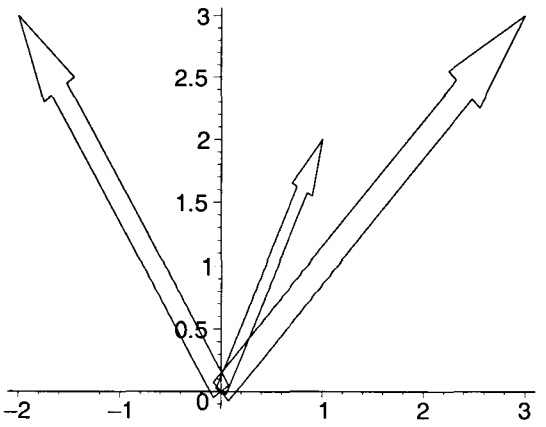
Antes de hacer uso de estas funciones debemos cargar el paquete con la instrucción: **with(plots)**.

11.7.1 Gráficas de vectores

Este tipo de gráficas pueden generarse por medio de la instrucción **arrow(v)**. Donde "**v**" es un vector, un conjunto de vectores o una lista de vectores, cada uno de los cuales puede ser dado en la forma: **[x, y]** o **<x, y>**. También puede ser generado por medio de las funciones: **array([x, y])**, o bien, **Vector([x, y])**.

Veamos un ejemplo:

```
> arrow({<-2, 3>, [1, 2], <3, 3>});
```



Algunas de las opciones que se pueden aplicar a este tipo de gráficas son:

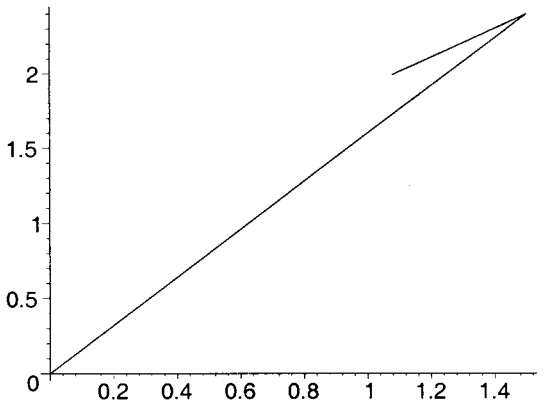
- **shape = harpoon, arrow, double_arrow.**

  Esta opción determina el tipo de flecha que será utilizada, el valor predeterminado es **double_arrow** (existe otra opción conocida como **cylindrical_arrow**, la cual solo puede ser usada en el despligue de vectores en tres dimensiones). Por ejemplo:

```
> v1 := array(1..2, [1.5, 2.4]);
```

$$v1 := [1.5, 2.4]$$

```
> arrow(v1, shape=harpoon);
```



- **width = n.** Esta opción determina el ancho del vector. Por ejemplo:

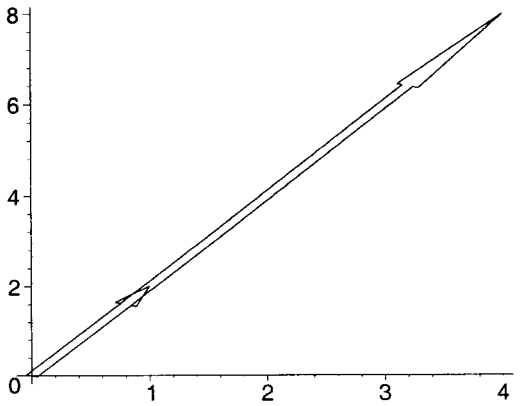
```
> v2 := Vector([1, 2]);
```

$$v2 := \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

```
> v3 := Vector([4, 8]);
```

$$v3 := \begin{bmatrix} 4 \\ 8 \end{bmatrix}$$

```
> arrow({v2, v3}, shape=double_arrow, width=0.1);
```



11.7.2 Gráficas de complejos

Otra de las funciones incluidas en el paquete **plots** es **complexplot**. Esta función permite obtener gráficas en dos dimensiones de una o más funciones, expresiones, procedimientos, funciones paramétricas o listas de numeros complejos. Su sintaxis es:

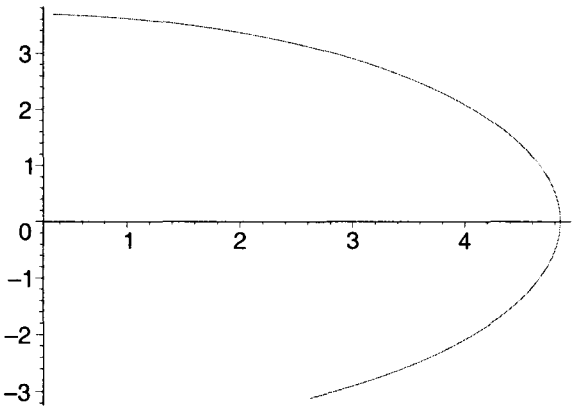
complexplot(f, x=a..b)

Donde **f** es una función de una variable, la cual representa un mapeo de los reales en los complejos; ***"x=a..b"*** especifica el rango en el cual se graficará. Por ejemplo:

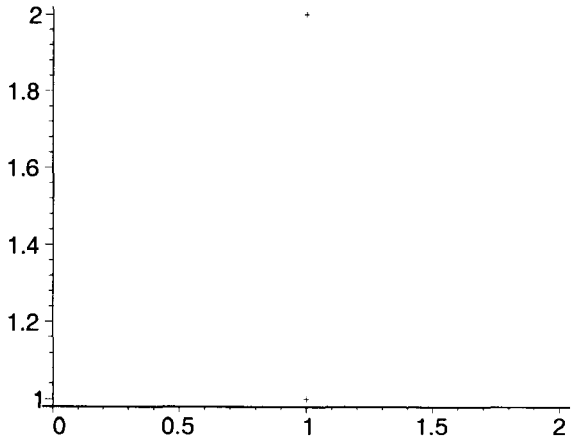
```
> fc := x -> Pi*(cos(x + I));
```

$$fc := x \to \pi \cos(x + I)$$

```
> complexplot(fc(x), x=-1.5..1);
```




```
> complexplot([1 + 2*I, 3 + 4*Pi*i, sin(Pi/2) + I], x=-10..10,
> style=point, symbolsize=18);
```



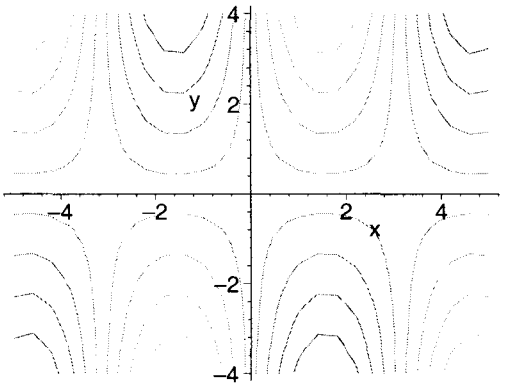
11.7.3 Gráficas de contornos

Este tipo de gráficas pueden ser generadas por medio de la instrucción:

contourplot(f, x=a..b, y=c..d)

Donde **f** es la función o expresión a gráficar (de dos variables), mientras que los intervalos incluidos determinan un área rectangular en la que se desplegará la gráfica. Por ejemplo:

```
> contourplot(sin(x)*y, x=-5..5, y=-4..4);
```



11.7.4 Mapas de densidad

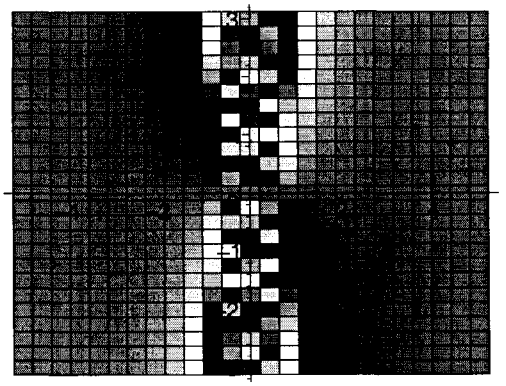
Éstos pueden obtenerse por medio de la instrucción **densityplot**. Su sintaxis es:

densityplot(f, x=a..b, y=c..d)

densityplot(f, a..b, c..d)

Donde **f** es una función o expresión (de dos variables) y los intervalos determinan un área rectangular en la que se desplegará la gráfica. Por ejemplo:

```
> densityplot(sin(y/x^3), x=-5..5, y=-3..3);
```



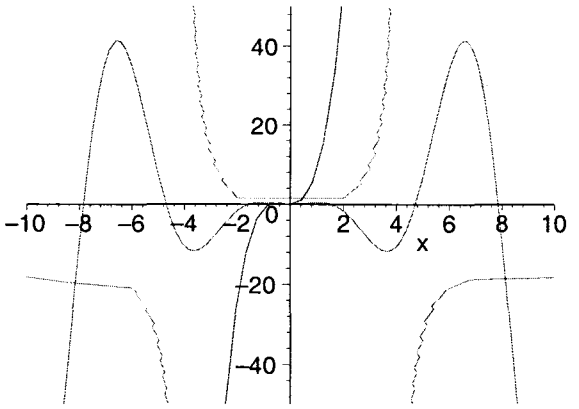
11.7.5 Despliegue de gráficas con estructuras diferentes

Este paquete nos proporciona la función **display**, la cual nos permite mostrar varias funciones en un mismo despliegue, sin importar que tengan estructuras diferentes. Su sintaxis es:

display([g1, g2, g3, ...])

Donde *"g1, g2, g3, ..."* son estructuras gráficas previamente creadas. Por ejemplo:

```
> g1 := plot(x^2*cos(x), x, -50..50, color=brown):

> g2 := plot(4*x^2 + 5*x^3, x, -50..50, color=blue):

> g3 := implicitplot(x - 1/x = y*sin(x), x=-50..50, y=-50..50,
> color=magenta):

> plots[display]([g1, g2, g3]);
```



11.7.6 Gráficas de funciones implícitas

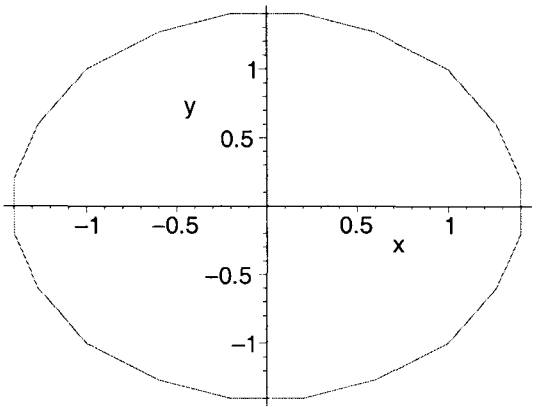
Otra función contenida en **plots** es **implicitplot**, la cual nos permite obtener gráficas de funciones cuya regla de correspondencia está dada de manera implícita. Su sintaxis es:

implicitplot(f, x=a..b, y=c..d)

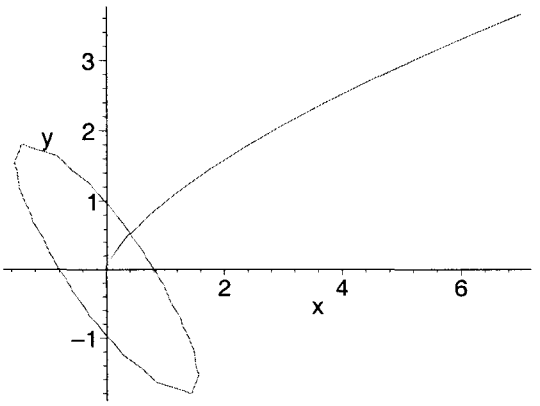
donde **f** es la función o expresión en términos de dos variables (pueden ser varias expresiones o funciones dadas en forma de conjunto), mientras que **x** y **y** determinan los intervalos para los que se desplegará la gráfica.

Veamos los siguientes ejemplos:

```
> implicitplot(x^2 + y^2 = 2, x=-5..5, y=-5..5);
```




```
> implicitplot({x = y*sqrt(y), (x + y)^2 = cos(x)}, x=-7..7,
> y=-4..4);
```



11.7.7 Gráficas de desigualdades

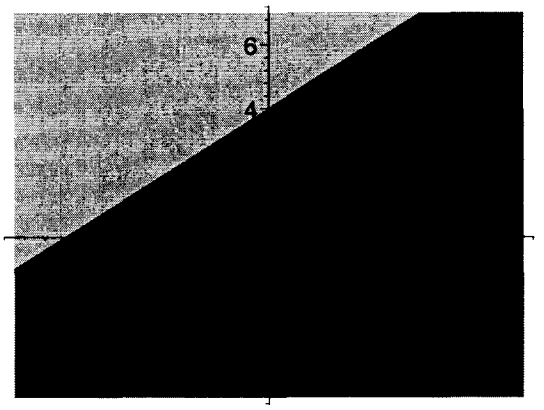
Este tipo de gráficas pueden generarse con la función **inequal**. Su sintaxis es:

inequal(exp, x=a..b, y=c..d)

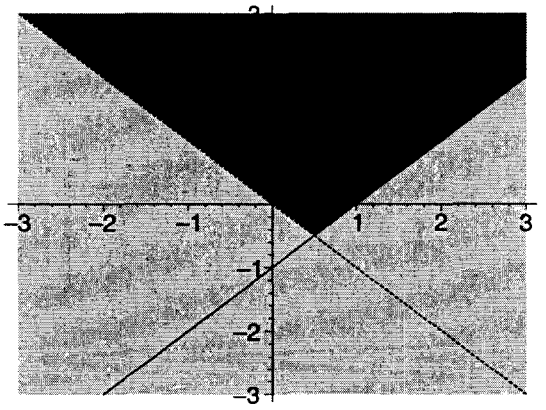
Donde **exp** es una ecuación, inecuación o un conjunto de éstas. Los intervalos incluidos determinan un área rectangular en la que se desplegará la gráfica.

Veamos los siguientes ejemplos:

```
> inequal((x + 1) < (y - 3), x=-5..5, y=-5..7);
```




```
> inequal({x + y > 0, x - y <= 1}, x=-3..3, y=-3..3,
> optionsfeasible=(color=red), optionsexcluded=(color=grey) );
```



En este último ejemplo, la opción: **"optionsfeasible"**, determina el color con el cual se desplegará el área donde las desigualdades se cumplen; mientras que la opción **"optionsexcluded"** determina el color para el área donde las desigualdades no se cumplen. Por omisión, esta función despliega en color claro el área en que las desigualdades se cumple y en color obscuro el área en la que no se cumplen.

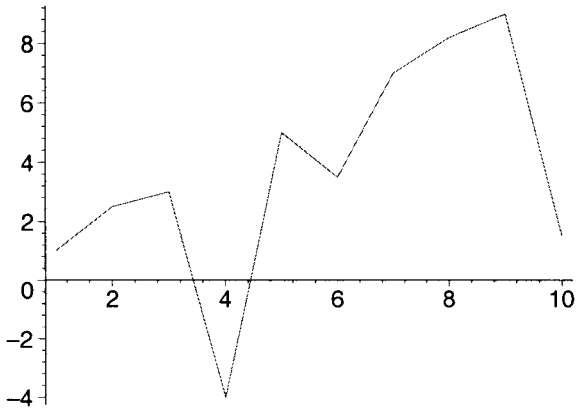
11.7.8 Gráficas de listas

La función **listplot**, nos permite graficar listas de números. Su sintaxis es:

listplot(L)

Donde **L** es una lista de datos numéricos o puntos. Por ejemplo:

```
> listplot([1, 2.5, 3, -4, 5, 3.5, 7, 8.2, 9, 1.5], color=red);
```



En este ejemplo puede notarse que la función **listplot** toma una lista de la forma:

[a, b, c, d, e, f, g, h, ...]

y a partir de ella crea un conjunto de puntos:

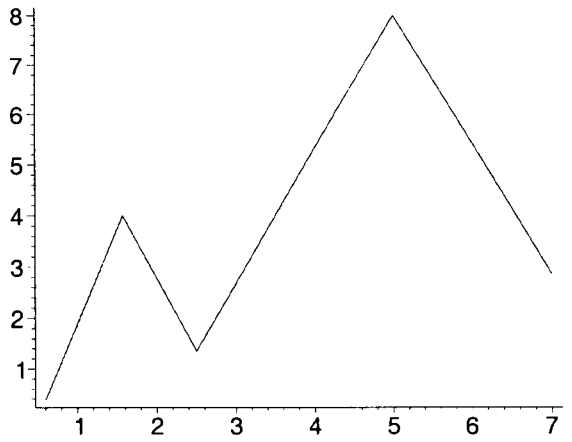
(1, a), (2, b), (3, c), (4, d), (5, e) ...

los cuales une con una línea en el orden que fueron colocados en la lista. Esta última también puede darse en la forma:

[[x1, y1], [x2, y2], [x3, y3], [x4, y4], ...]

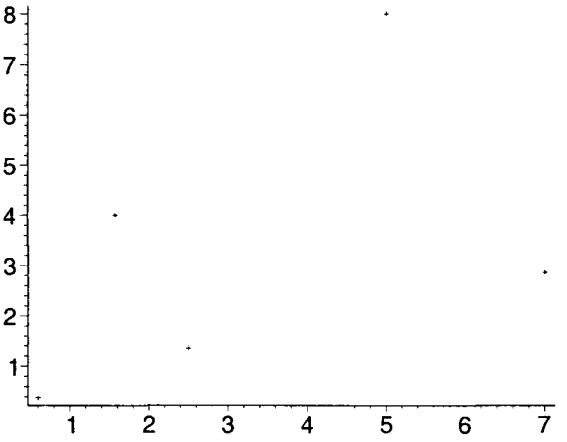
Veamos un ejemplo:

```
> listplot([[.6, sin(Pi/8)], [Pi/2, 4], [2.5, exp(0.3)], [5, 8],
> [7, 9/Pi]], color=blue);
```



Si deseamos que los puntos sean graficados como tales, debemos incluir la opción **style=point**.

```
> listplot([[.6, sin(Pi/8)], [Pi/2, 4], [2.5, exp(0.3)], [5, 8],
> [7, 9/Pi]], style=point, symbolsize=14, color=blue);
```



11.7.9 Gráficas de soluciones de ecuaciones diferenciales

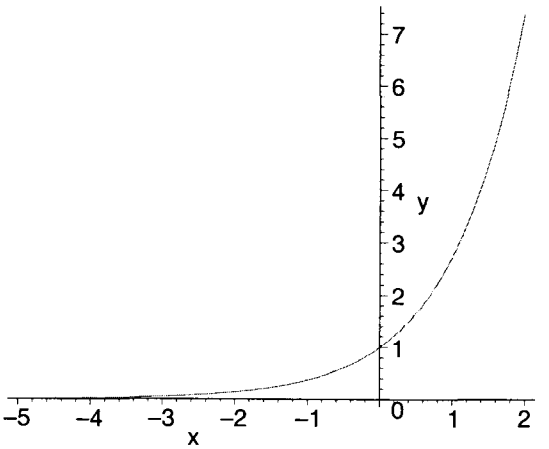
La función **odeplot** del paquete **plots**, nos permite graficar una solución particular para una ecuación diferencial o un sistema de ecuaciones diferenciales, obtenida mediante **dsolve**; siempre que esta solución se encuentre en forma numérica. Su sintaxis es:

odeplot(sol)

Donde **sol** es una solución particular para una ecuación diferencial (o un sistema de ecuaciones diferenciales), obtenida de manera numérica; es decir, mediante la instrucción: **dsolve(..., numeric)**.

Por ejemplo, graficaremos la solución de la ecuación: $\frac{d}{dx}\,y(x) = y(x)$, para $y(0) = 1$.

```
> s := dsolve({D(y)(x) = y(x), y(0)=1}, type=numeric, range=-5..2);

> odeplot(s);
```



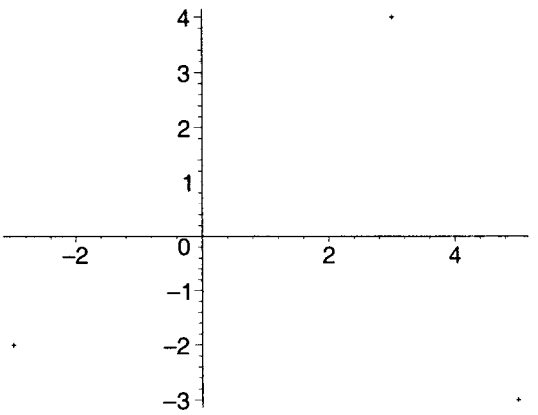
11.7.10 Gráficas de puntos

Otra de las funciones contenidas dentro de este paquete es **pointplot**, la cual nos permite generar gráficas de puntos. Su sintaxis es:

pointplot(pts, opciones)

Donde **pts** es una lista o conjunto de puntos. Por ejemplo:

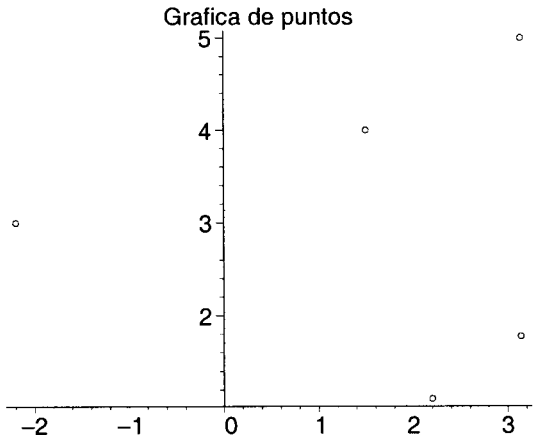
```
> pointplot({[3, 4], [5, -3], [-3, -2]}, symbolsize=14);
```



pointplot es similar a **listplot**, pero a diferencia de ésta última, **pointplot** no une los puntos recibidos con una línea de manera predeterminada, para que lo haga debe incluirse la opción **style=line**.

Esta función también es similar a **plot**, invocada con la opción **style=point**. Veamos otro ejemplo:

```
> pointplot({[-2.2, 3], [1.5, 4], [3.14, 5], [Pi, sqrt(Pi)],
> [2.2, exp(0.1)]}, symbol=circle, symbolsize=15, color=blue,
> title='Grafica de puntos');
```



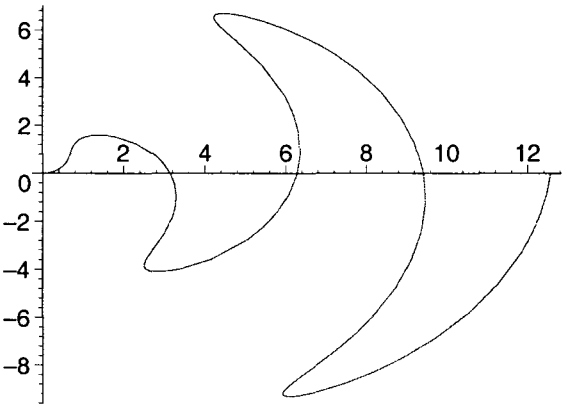
11.7.11 Gráficas en coordenadas polares

Las funciones dadas en coordenadas polares pueden ser graficadas por medio de la instrucción **polarplot**. La sintaxis es:

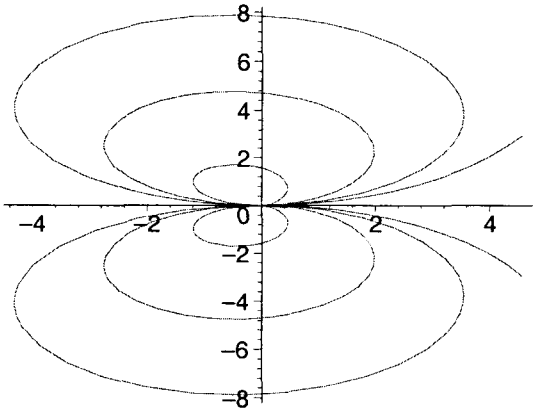
polarplot(expr, opciones)

Donde **expr** es una curva o un conjunto de curvas dadas en coordenadas polares. Las opciones aplicables a esta función son las mismas de **plot**. Por ejemplo:

```
> polarplot([t, sin(t), t=0..4*Pi], color=blue);
```




```
> polarplot(x*sin(x), x=-10..10);
```

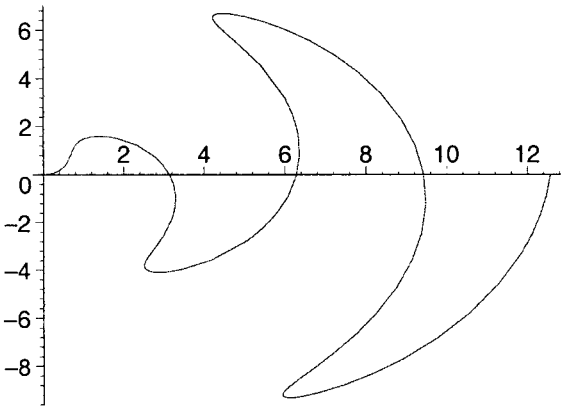


En estos ejemplos, cuando la función a graficar está dada en la forma "**[f(x), g(x)]**", la primera componente es considerada el radio y la segunda el ángulo. En cambio, si la función está dada en la forma "**h(x)**" (como en la segunda gráfica), la variable es considerada el ángulo y la función el radio.

Este tipo de gráficas también pueden desplegarse por medio de **plot**, incluyendo la opción **coords=polar**.

Veamos el siguiente ejemplo:

```
> plot([t, sin(t), t=0..4*Pi], coords=polar, color=blue);
```



11.7.12 Gráficas de polígonos

La función **polygonplot** del paquete **plots**, nos permite desplegar gráficas de polígonos. La sintaxis es:

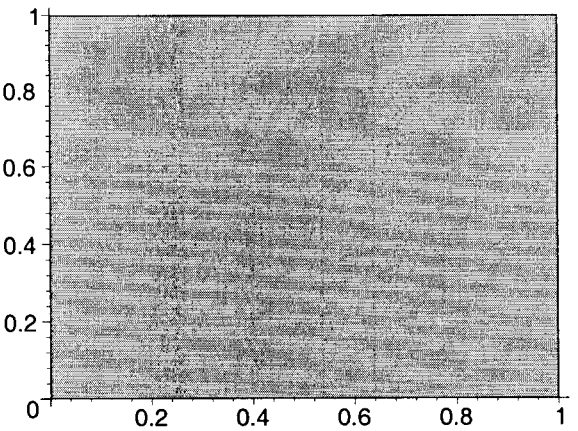
polygonplot(pts, opciones)

Donde **pts** son los puntos de un polígono o de un conjunto de polígonos (cada uno debe ser dado como una lista de puntos). Las opciones aplicables son las mismas de **plot**. Veamos algunos ejemplos:

```
> pol1 := [[0, 0], [1, 0], [1, 1], [0, 1]];
```

$$pol1 := [[0, 0], [1, 0], [1, 1], [0, 1]]$$

```
> polygonplot(pol1, color=cyan);
```



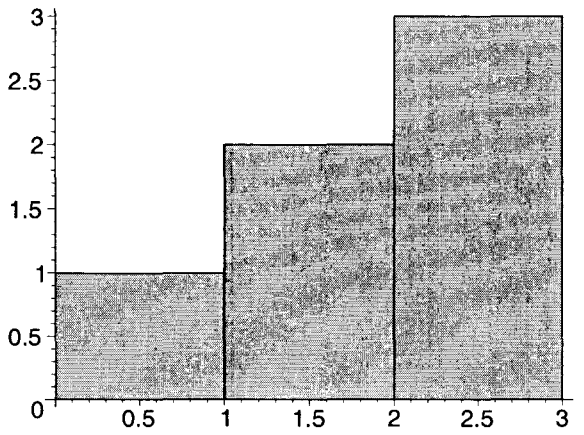

```
> pol2 := [[1, 0], [2, 0], [2, 2], [1, 2]];
```

$$pol2 := [[1, 0], [2, 0], [2, 2], [1, 2]]$$

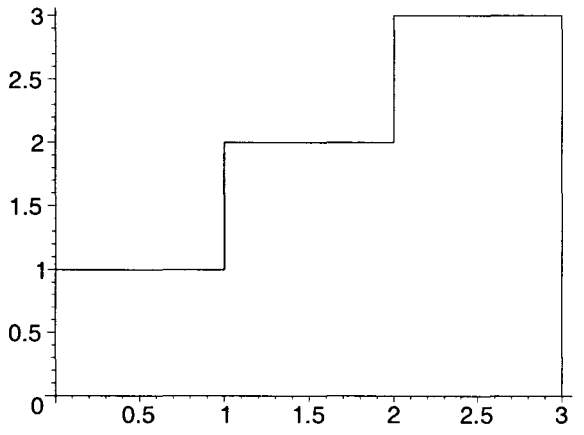
```
> pol3 := [[2, 0], [3, 0], [3, 3], [2, 3]];
```

$$pol3 := [[2, 0], [3, 0], [3, 3], [2, 3]]$$

```
> polygonplot([pol1, pol2, pol3], color=gray);
```




```
> polygonplot([[0, 0], [0, 1], [1, 1], [1, 2], [2, 2], [2, 3],
> [3, 3], [3, 0]]);
```



11.7.13 Gráficas de texto

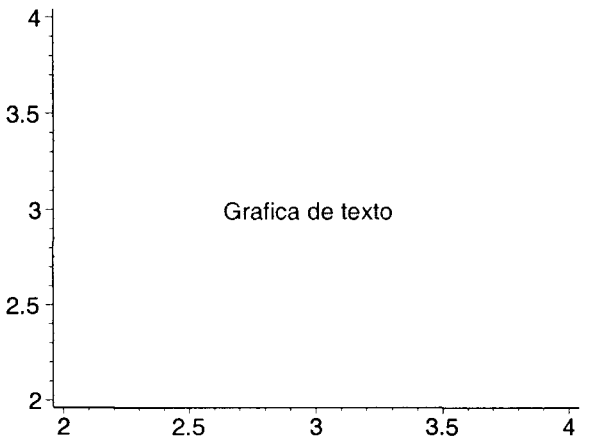
Otra función útil es **textplot**, por medio de la cual se pueden crear gráficas de cadenas de texto en dos dimensiones. Su sintaxis es:

textplot(expr, opciones)

Donde **expr** es una lista que contiene las coordenadas de un punto en dos dimensiones, así como la cadena de texto que será colocada en este punto.

Las opciones aplicables a esta función son básicamente las mismas de **plot**. Por ejemplo:

```
> textplot([3, 3, 'Grafica de texto'], color=blue);
```



En una misma gráfica se pueden colocar varias cadenas de texto, expresándolas como una lista de listas o un conjunto de listas. Por ejemplo:

```
> txt1 := [-2, 2, 'Texto 1'];
```

$$txt1 := [-2, 2, Texto\ 1]$$

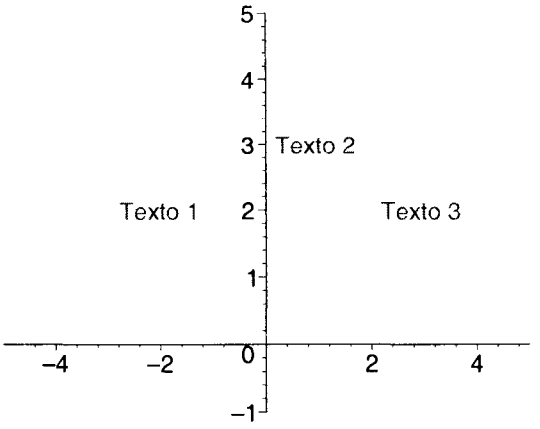
```
> txt2 := [1, 3, 'Texto 2'];
```

$$txt2 := [1, 3, Texto\ 2]$$

```
> txt3 := [3, 2, 'Texto 3'];
```

$$txt3 := [3, 2, Texto\ 3]$$

```
> textplot({txt1, txt2, txt3}, view=[-5..5,-1..5], color=red);
```

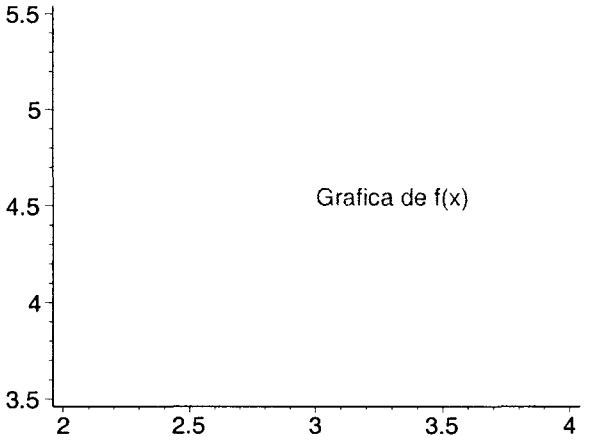


De manera predeterminada, **textplot** coloca la cadena de texto centrada con respecto al punto recibido como argumento (tanto horizontal como verticalmente). Esto puede modificarse por medio de la opción:

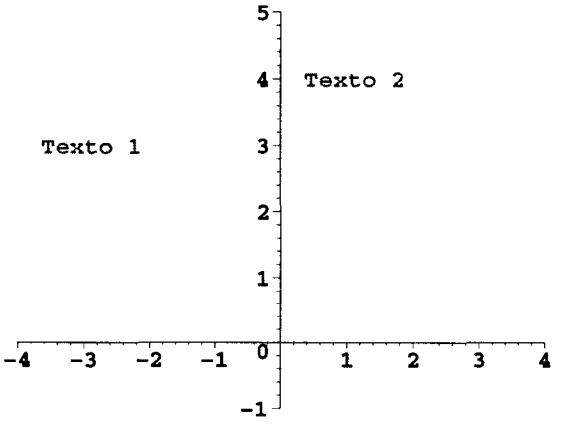
align=opciones de alineación

Las opciones de alineación pueden ser **ABOVE**, **BELOW**, **RIGHT** o **LEFT**, las cuales pueden usarse individualmente o en combinación. Por ejemplo:

```
> textplot([3, 4.5, 'Grafica de f(x)'], align={ABOVE, RIGHT}, color=brown);
```




```
> textplot([[-2, 3, 'Texto 1'], [2, 4, 'Texto 2']], align=LEFT,
> color=maroon, font=[COURIER, BOLD, 10], view=[-4..4, -1..5]);
```



Este tipo de despliegues pueden ser combinados con gráficas de funciones por medio de la instrucción **display**. Veamos un ejemplo.

Primero definimos las siguientes funciones:

```
> f1 := x -> sin(x);
```

$$f1 := \sin$$

```
> f2 := x -> x^3/(20*Pi^2);
```

$$f2 := x \rightarrow \frac{1}{20}\frac{x^3}{\pi^2}$$

```
> f3 := x -> -x^3*sin(x)/(20*Pi^2);
```

$$f3 := x \rightarrow -\frac{1}{20}\,\frac{x^3 \sin(x)}{\pi^2}$$

A continuación generamos sus gráficas:

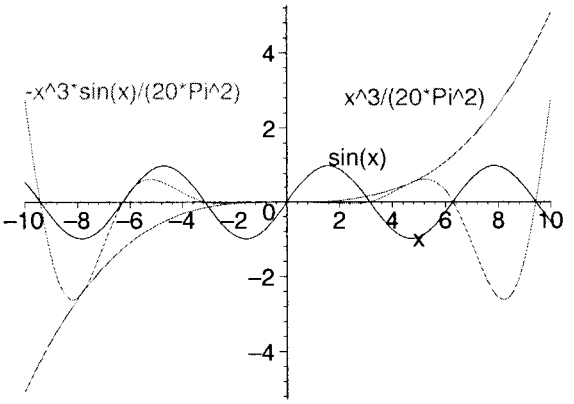
```
> g1 := plot(f1(x), x=-10..10, color=blue):
> g2 := plot(f2(x), x=-10..10, color=brown):
> g3 := plot(f3(x), x=-10..10, color=magenta):
```

Después generamos una gráfica de texto para cada función:

```
> tx1 := textplot([Pi/2, f1(Pi/2), 'sin(x)'], color=blue,
> align={ABOVE, RIGHT}):
> tx2 := textplot([8, f2(8), 'x^3/(20*Pi^2)'], color=brown,
> align={ABOVE, LEFT}):
> tx3 := textplot([-10, f3(-10), '-x^3*sin(x)/(20*Pi^2)'],
> color=magenta,
> align={ABOVE,RIGHT}):
```

Finalmente desplegamos todos estos elementos con **display**:

```
> display([g1, g2, g3, tx1, tx2, tx3]);
```



Existen otras funciones para despliegue de gráficas en dos dimensiones, incluidas en el paquete **plots**. Consulte la ***página de ayuda*** mediante la instrucción: **?plots** para obtener información sobre ellas.

11.7.14 Manipulación de la región gráfica

En las secciones anteriores se describió la utilidad de algunos de los botones que aparecen en la barra contextual cuando se selecciona una región gráfica. Otros de los elementos que aparecen en esta barra son:

- 2.08, 2.24. En esta región se despliegan un par de coordenadas, cada vez que se seleccciona un punto en la gráfica por medio del ratón.

-  . Estos dos botones permiten modificar el despliegue de la gráfica seleccionada, utilizando una linea continua o un conjunto de puntos, respectivamente. Estos botones tienen el mismo efecto sobre la gráfica que las opciones **style=LINE** y **style=POINT**.

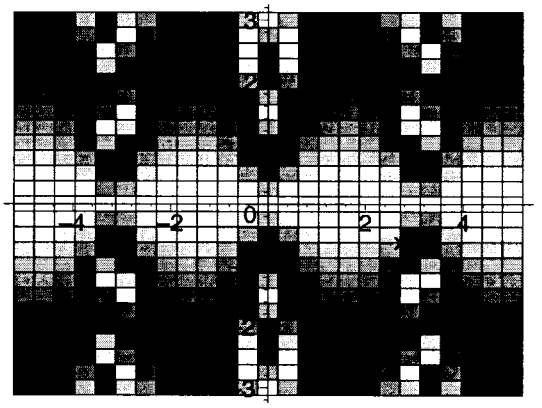
-  . Estos dos botones solo son aplicables a gráficas en las que se despliegan superficies. El primero de estos permite desplegar la gráfica con una malla, mientras que el segundo elimina esta malla.

  La acción de estos botones es equivalente a utilizar las opciones:

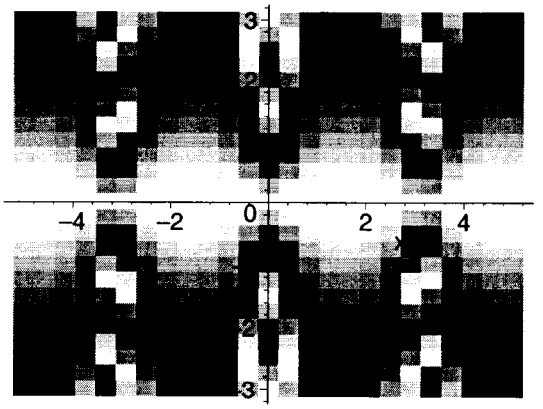
  style=PATCH y **style=PATCHNOGRID**

  Un caso en el que se pueden aplicar estos botones en gráficas de dos dimensiones es en los mapas de densidades. Para ejemplificar estas opciones comparense las siguientes gráficas:

```
> plots[densityplot](cos(y/sin(x)), x=-5..5, y=-3..3, style=PATCH);
```




```
> plots[densityplot](cos(y/sin(x)), x=-5..5, y=-3..3, style=PATCHNOGRID);
```



Muchas de las opciones descritas en este capítulo pueden ser aplicadas directamente a las gráficas por medio del menú contextual, para ello colocamos el apuntador del ratón sobre una gráfica y oprimimos el botón derecho, esto desplegrará el menú contextual que se muestra en la Figura 11.2.

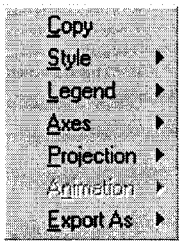


Figura 11.2: Menú contextual para gráficas

Por medio de éste podemos modificar en la gráfica, por ejemplo, el estilo de las líneas, el tipo de simbólos usados (por ejemplo para gráficas de puntos) y el tipo de ejes, entre otras cosas. Además, la opción **Export As**, nos permite exportar la gráfica como imagen a un archivo, en formatos tales como: GIF, JPEG, EPS (Postscript encapsulado), entre otros.

11.7.15 Creación de gráficas de forma interactiva

Esta versión de Maple nos ofrece una opción para la creación de gráficas en dos dimensiones de manera interactiva, haciendo uso de una nueva herramienta conocida como **Interactive Plot Builder**. Ésta puede ser invocada por medio de la instrucción **interactive** del paquete **plots**.

Para ejemplificar esto generaremos la gráfica de la función **sin(sqrt(abs(x)))**; para ello invocamos la función **interactive** como se muestra a continuación:

```
> plots[interactive](sin(sqrt(abs(x))));
```

esta instrucción desplegará la ventana del **Interactive Plot Builder** que se muestra en la Figura 11.3.

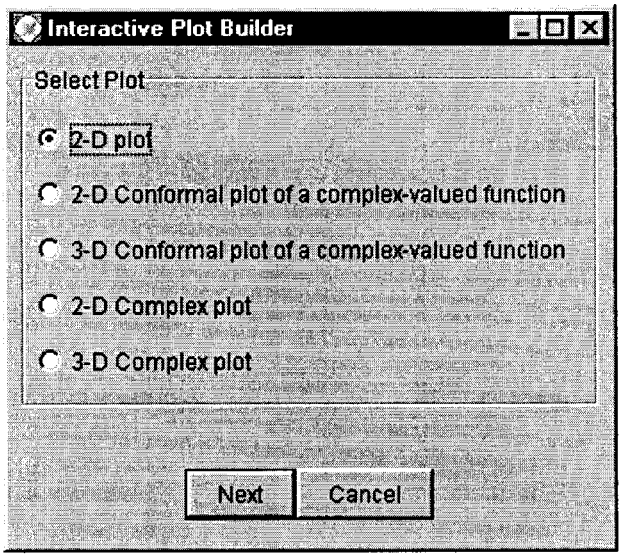


Figura 11.3: Ventana inicial del Interactive Plot Builder

Seleccionamos **2-D plot** en esta ventana y oprimimos el botón **Next**. A continuación aparecerá la ventana de la Figura 11.4.

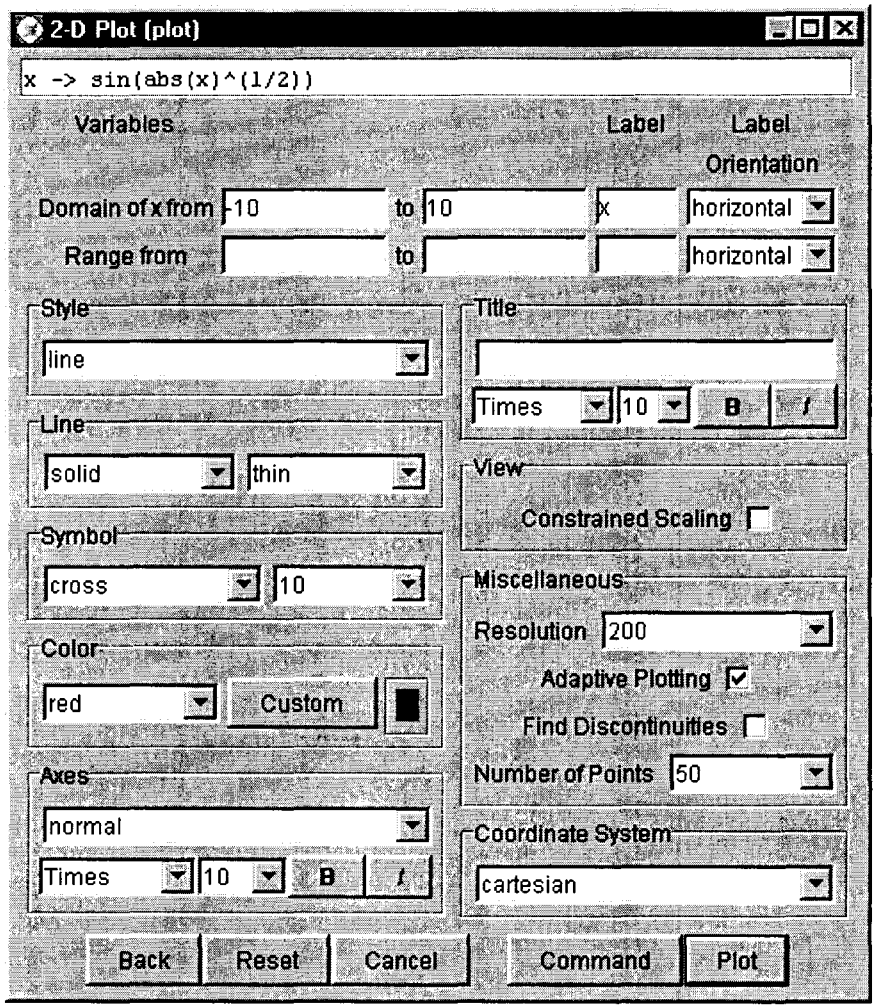


Figura 11.4: Interactive Plot Builder

En ésta última podemos especificar todas las opciones que deberán aplicarse a nuestra gráfica, por ejemplo: el rango para el eje y (en la sección **Range from .. to**), el color de la gráfica, el sistema de coordenadas a usar, el tipo de línea, el tipo de ejes, etcétera. De hecho, usando esta herramienta podemos aplicar todas las opciones que se describieron en este capítulo. Al final, oprimimos el botón **Plot**, con lo cual se desplegará la gráfica.

Acerca de la ventana del **Interactive Plot Builder**, ésta nos permite generar otro tipo de gráficas, por ejemplo gráficas de complejos en dos y tres dimensiones, como puede verse en la Figura 11.3.

11.7.16 Creación de gráficas a partir de una expresión de salida

Otra opción más que podemos usar para generar una gráfica es seleccionando, en una región de salida, una expresión o parte de ésta y oprimiendo el botón derecho del ratón; en el menú contextual que aparece seleccionamos el submenú **Plots** y en éste podemos seleccionar las opciones **Plot Builder** o **2D-Plot**

(también nos proporciona la opción **3D-Plot** para despliegue de gráficas en tres dimensiones), como puede verse en la Figura 11.5.

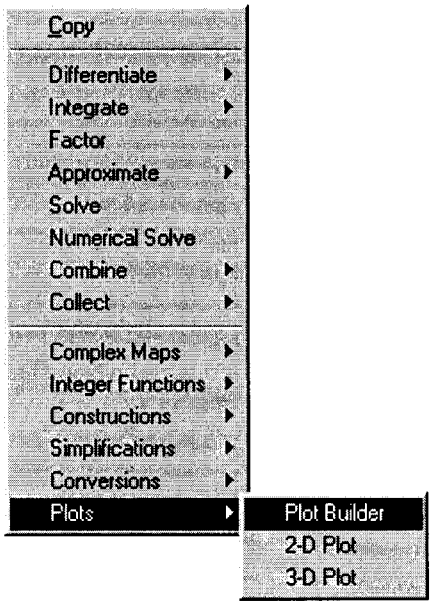


Figura 11.5: Menú contextual para generación de gráficas

La primera opción desplegará la ventana descrita en la sección anterior, mientras que la segunda opción generará automáticamente una gráfica de la expresión seleccionada (este despliegue se genera invocando a la función **smartplot**).

Además de la forma anterior, esta versión de Maple nos proporciona una manera más para el despliegue de gráficas en dos dimensiones a partir de una expresión de salida (este mismo método puede ser usado para despliegue de gráficas en tres dimensiones). Seleccionamos, en el submenú **Plot** del menú **Insert** la opción **2-D**; esto insertará una región gráfica vacía en nuestra hoja de trabajo. Para poder desplegar la gráfica de una función en esta región insertada, seleccionamos, de una región de salida, una expresión o una parte de ésta y, por medio de las opciones del menú **Edit**, copiamos y pegamos esta expresión dentro de la región gráfica vacia; Maple automáticamente generará una gráfica de la expresión seleccionada. De esta misma forma, en el mismo despliegue podemos copiar y pegar varias expresiones, con lo cual obtendremos una gráfica que incluya todas éstas.

Capítulo 12

Gráficas en tres dimensiones

Maple también proporciona un conjunto de funciones útiles para el despliegue y manejo de gráficas en tres dimensiones. Estas instrucciones permiten generar gráficas de diversos tipos, entre las cuales se pueden incluir gráficas de funciones explícitas, implícitas, de funciones paramétricas, gráficas de puntos, curvas de nivel, gráficas de funciones expresadas en coordenadas cilindricas, esféricas y polares, entre otras. A continuación revisaremos las principales funciones que nos permiten generar este tipo de gráficas.

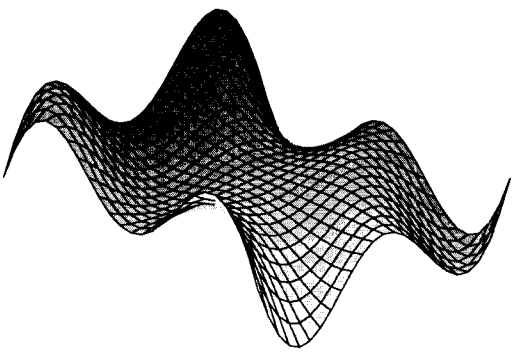
12.1 Funciones explícitas

Una de las instrucciones para despliegue de gráficas en tres dimensiones más importantes de Maple es **plot3d**. Ésta nos permite obtener gráficas de funciones explicitas, entre otras cosas. Su sintaxis es:

plot3d(expr, x=a..b, y=c..d, opciones);

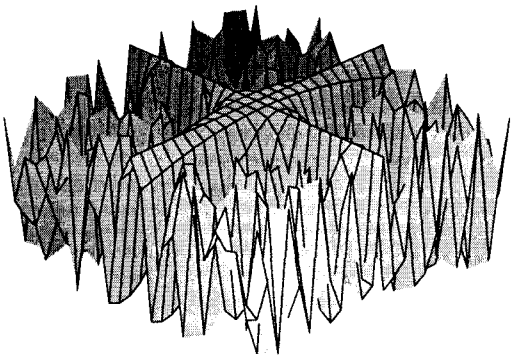
Donde **expr** es una expresión o función en términos de dos variables, mientras que *"x=a..b"* y *"y=c..d"* representan los intervalos en los cuales se evaluará la función para gráficarla. Veamos un ejemplo:

```
> plot3d(x*y*sin(x - y), x=-Pi..Pi, y=-Pi..Pi);
```



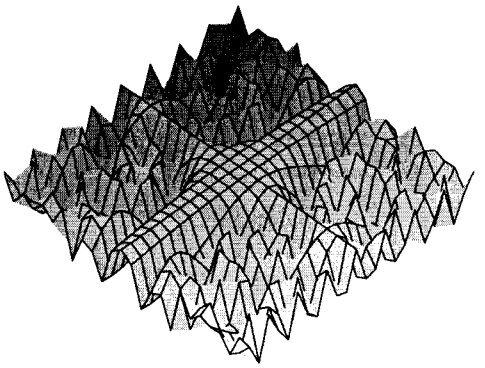
A diferencia de la función **plot**, en el caso de **plot3d**, los rangos para x e y no pueden omitirse, pues esto generará un mensaje de error. Por otro lado, al igual que en el caso de dos dimensiones, las gráficas en tres dimensiones no son desplegadas a escala. Consideremos la siguiente gráfica:

```
> plot3d(cos(x*y^2), x=-5..5, y=-5..5);
```



Una forma de hacer que ésta aparezca a escala real, es mediante el uso del botón 1:1, que aparece en la barra contextual para regiones gráficas. Otra manera de hacer esto es incluyendo, como parte de **plot3d**, la opción **scaling=CONSTRAINED**. Compárese la gráfica anterior con la siguiente:

```
> plot3d(cos(x*y^2), x=-5..5, y=-5..5, scaling=CONSTRAINED);
```



Esta última muestra el aspecto verdadero de la función $\cos(x\,y^2)$.

12.1.1 Movimiento de la gráfica

Una caractéristica muy útil de este tipo de despliegues generados por Maple, es que podemos manipularlos directamente en la región gráfica. Por ejemplo, para poder visualizar la gráfica de la instrucción anterior desde puntos de vista diferentes, es suficiente con seleccionar con el ratón la región en la cual se encuentra y, manteniendo el cursor sobre esta región, oprimir el botón izquierdo y mover el ratón; automáticamente Maple comienza a mover la gráfica, permitiendonos observar el aspecto de ésta desde cualquier punto.

A grandes rasgos, al invocar la función:

plot3d(f(x, y), x=a..b, y=c..d);

Maple procede de la siguiente forma para desplegar la gráfica:

- Toma los intervalos **"a..b"** y **"c..d"**, formando con ellos una malla rectangular; para ello, cada uno de los intervalos es dividido en 25 puntos (es decir, se forma una malla de 625 puntos).
- A continuación se evalua la función en cada uno de los puntos de esta malla.
- Con estos datos se forma un conjunto de puntos de la forma **[xi, yi, f(xi, yi)]**, donde **[xi, yi]** es un punto de la malla.
- Cada cuadrícula de la malla, después de ser evaluada por medio de la función, debe ser cubierta con una superficie. Para calcular el aspecto de cada una de estas superficies, Maple considera el color asignado a la gráfica, la posición en la que se desplegará, una luz ambiental, así como una luz que incide sobre la gráfica desde una dirección especifica del espacio y con un color particular.
- Tomando en cuenta los datos anteriores, Maple genera una estructura que contiene todos los datos acerca de la gráfica, tales como los puntos de la cuadrícula formada por la malla, el tipo de ejes a desplegar, la posición en que se desplegará la gráfica, los colores asignados, etc.
- Una vez generada esta estructura, Maple la despliega en la pantalla, manteniendo en memoria toda esta información. Es precisamente esta característica la que nos permite seleccionar y mover la gráfica directamente sobre la región que la contiene.

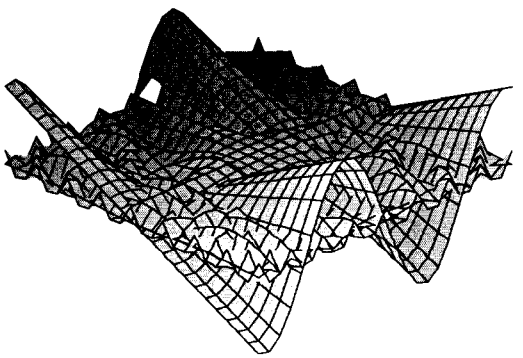
12.2 Despliegue de varias funciones explícitas

La función **plot3d** nos permite colocar, en la misma instrucción, varias funciones o expresiones para desplegarlas en la misma gráfica; para ello éstas deben ser introducidas en forma de conjunto, es decir:

plot3d({f1(x, y), f2(x, y), f3(x, y), ...}, x=a..b, y=c..d);

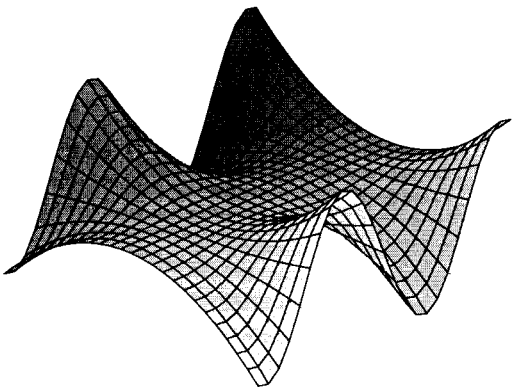
Por ejemplo:

```
> plot3d({sin(x)*y, sin(x^2)*cos(y), sin(cos(x*y))}, x=-5..5, y=-5..5);
```



Cuando se introducen expresiones todas deben depender de las mismas variables, de lo contrario no obtendremos los resultados esperados. Por ejemplo:

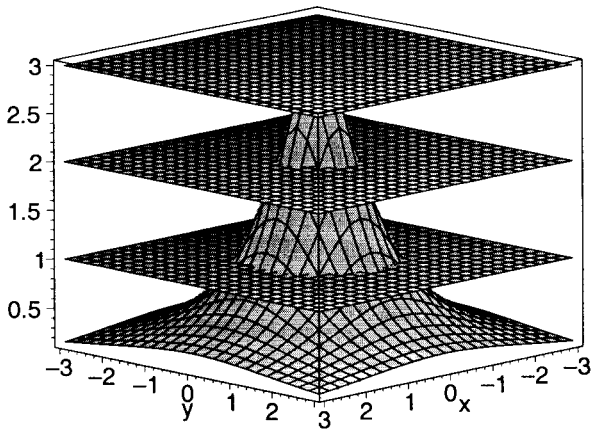
```
> plot3d({sin(x)*y^2, a^2*b^2, sin(cos(g - f))}, x=-5..5, y=-5..5);
```



Puede notarse que, en esta última gráfica, solo se despliega la función $\sin(x)\,y^2$, pues las otras dos dependen de variables para las cuales no se proporcionan los intervalos a graficar.

En el siguiente ejemplo, desplegaremos la gráfica de **3/(1 + x^2 + y^2)**, además de los planos **z=1**, **z=2** y **z=3**.

```
> plot3d({3/(1 + x^2 + y^2), 1, 2, 3}, x=-3..3, y=-3..3, axes=boxed,
> orientation=[45,75]);
```



12.3 Principales opciones aplicables a gráficas en tres dimensiones

Existen diversas opciones que se pueden aplicar a las gráficas generadas por la función **plot3d**. Entre ellas tenemos: restringir el despliegue verticalmente, determinar la región del espacio que deseamos visualizar,

asignar distintos tipos de ejes, utilizar colores fijos o esquemas de colores, desplegar la superficie usando puntos, mallas de alambres o contornos, modificar la posición y el color de la luz incidente, especificar el color de la luz ambiental, aplicar diferentes tipos de proyecciones, entre muchas otras. Cabe mencionar que estas opciones, aplicables a gráficas generadas con **plot3d**, también pueden ser usadas en otras funciones de Maple que generan gráficas en tres dimensiones. A continuación revisaremos algunas de estas opciones más comunmente usadas.

12.3.1 Diferentes tipos de ejes

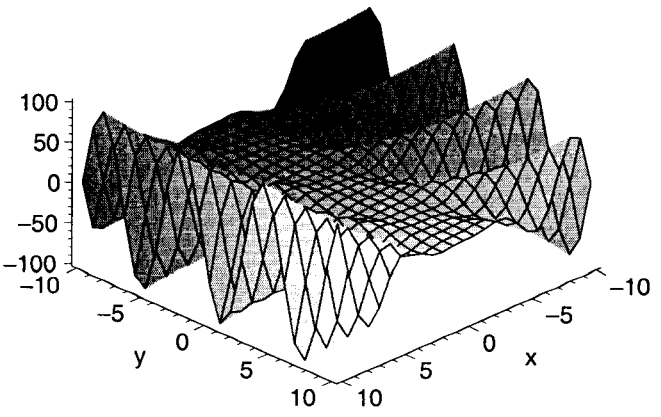
En las gráficas en tres dimensiones es posible modificar el tipo de ejes desplegados por medio de la opción **axes**. Los valores válidos para esta opción son los siguientes (pueden ser escritos en mayúsculas o minúsculas):

- **BOXED**. Despliega los ejes en forma de una caja. También puede ser aplicada directamente sobre la gráfica, por medio del botón  , de la barra contextual.
- **NORMAL**. Permite desplegar los ejes en forma de rectas que se intersectan en el origen. Esta opción también se puede aplicar directamente sobre la gráfica, a través del botón  , de la barra contextual.
- **FRAME**. Despliega los ejes en forma de un marco exterior a la gráfica. También puede aplicarse usando el botón  , de la barra contextual.
- **NONE**. Elimina el despliegue de los ejes. Esta es la opción que se usa como predeterminada, por lo cual no es necesario especificarla. Sin embargo, también se puede aplicar directamente sobre la gráfica, por medio del botón  , de la barra contextual.

Esta opción también puede aplicarse directamente en la región gráfica, sin necesidad de especificarla en la linea de comandos. Cuando se selecciona una región con una gráfica en tres dimensiones, en la barra de menús aparece **Axes**, dentro de éste se encuentran las opciones **Boxed**, **Framed**, **Normal** y **None**; para aplicarlas es suficiente con seleccionarlas en este menú y la gráfica se desplegará nuevamente en su región con el tipo de ejes indicado. Estas opciones también pueden aplicarse desde el menú contextual que se despliega si colocamos el apuntador del ratón sobre una gráfica en tres dimensiones y oprimimos el botón derecho.

Veamos un ejemplo:

```
> plot3d(x^2*sin(x + y), x=-10..10, y=-10..10, axes=FRAME);
```



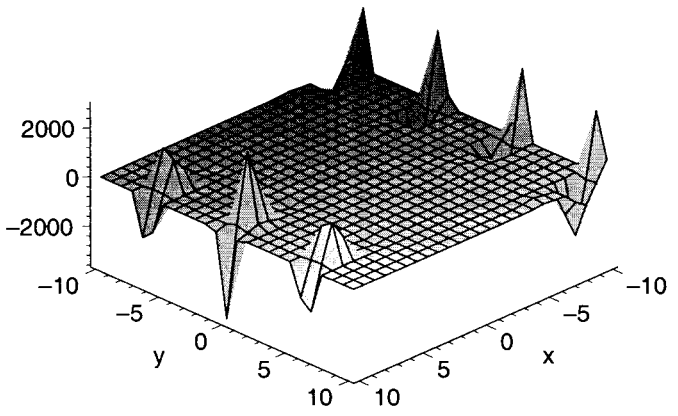
12.3.2 Restricción del rango vertical

Cuando se despliegan gráficas de funciones que crecen demasiado verticalmente, la opción **view** nos permite restringir el rango vertical e incluso especificar la región del espacio que deseamos visualizar. Por ejemplo, consideremos la siguiente función:

```
> f := (x, y) -> sin(x)^3*exp(x*cos(y));
```

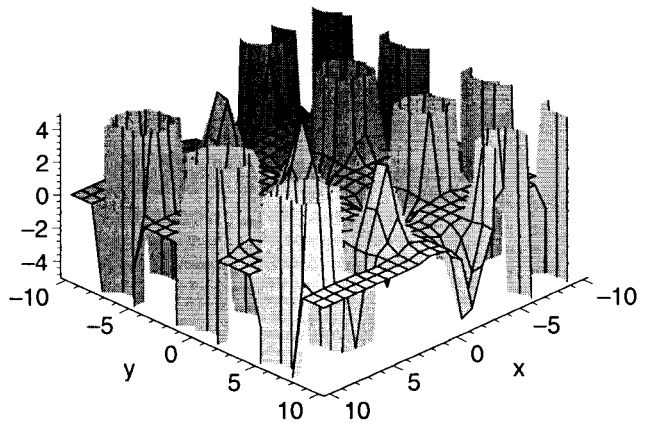
$$f := (x, y) \rightarrow \sin(x)^3 \, e^{(x \cos(y))}$$

```
> plot3d(f(x, y), x=-10..10, y=-10..10, axes=FRAME);
```



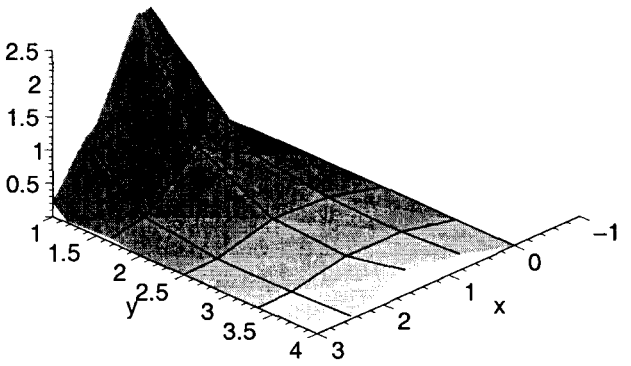
Puede notarse que la gráfica de la función crece demasiado para los valores de **x**, **y** especificados. A continuación restringiremos el despliegue en el eje **z**, para tener otra apreciación de esta función.

```
> plot3d(f(x, y), x=-10..10, y=-10..10, axes=FRAME, view=-5..5);
```



En algunos casos es útil hacer este tipo de restricciones para tener una mejor apreciación del comportamiento de la función graficada. Otra forma en la que podemos utilizar la opción descrita anteriormente es para indicar la región especifica del espacio en la que deseamos visualizar la gráfica. Por ejemplo, para la gráfica anterior, desplegaremos solo la región delimitada por **x=-1..3**, **y=1..4** y **z=0..2.5**.

```
> plot3d(f(x, y), x=-10..10, y=-10..10, axes=FRAME,
> view=[-1..3, 1..4, 0..2.5]);
```

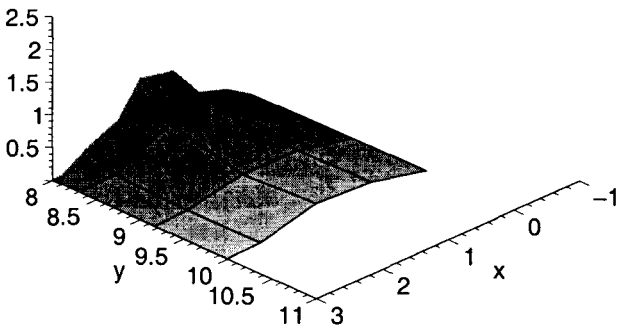


De esta manera podemos obtener secciones de la gráfica, especificando una región del espacio a través de la opción:

view=[xmin..xmax, ymin..ymax, zmin..zmax]

Nótese que, cuando es usada de esta forma, la opción **view** es independiente de los intervalos especificados para **x** e **y**; estos intervalos determinan la región del espacio en que la gráfica es visible. Si se especifica una región fuera de estos intervalos (por medio de **view**), solo se despliega la parte visible de la gráfica. Veamos un ejemplo:

```
> plot3d(f(x, y), x=-10..10, y=-10..10, axes=FRAME,
> view=[-1..3, 8..11, 0..2.5]);
```

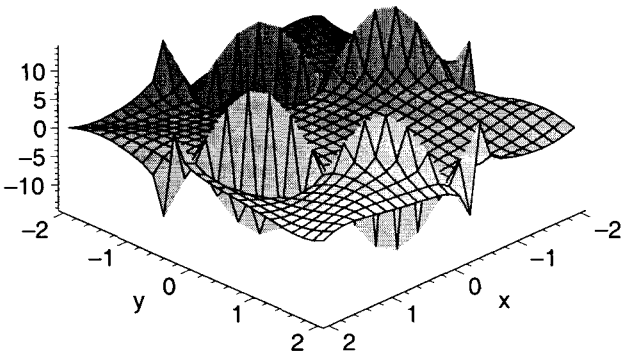


En este caso la gráfica no es desplegada para valores de **y** mayores que 10, esto por que los intervalos de graficación solo comprenden valores de -10 a 10 para esta variable.

12.3.3 Modificación de la malla

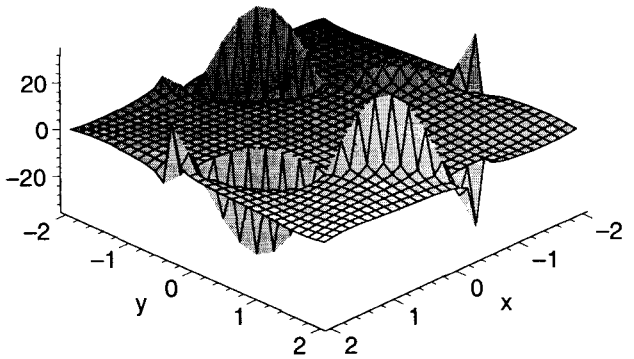
Existen varias opciones que permiten modificar la malla utilizada por la función **plot3d**. Una de estas opciones nos permite especificar el número de puntos que deben considerarse al crear la malla. Como se mencionó anteriormente, los intervalos de graficación proporcionados a **plot3d** son divididos por omisión en 25 puntos cada uno, generando con ellos una malla de 625 puntos. La opción **numpoints=n**, nos permite determinar la cantidad total de puntos a usar. Al incluir esta opción, Maple divide cada uno de los intervalos en una cantidad de puntos igual a **sqrt(n)**, generando así una malla de **n** puntos. Veamos un ejemplo:

```
> plot3d(sin(x^2 - y^2)/cos(x + y), x=-2..2, y=-2..2, axes=framed);
```



Desplegaremos nuevamente esta gráfica, con **900** puntos en la malla.

```
> plot3d(sin(x^2 - y^2)/cos(x + y), x=-2..2, y=-2..2, axes=framed,
> numpoints=900);
```

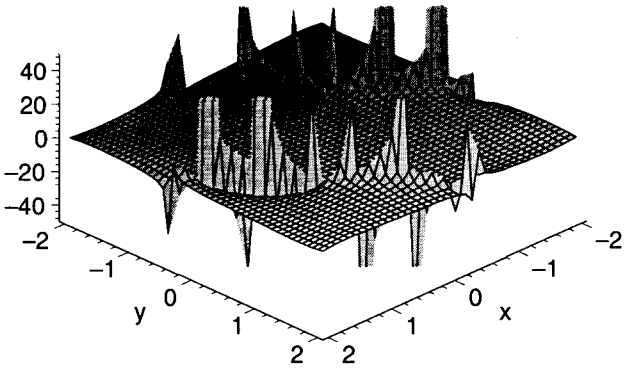


Claramente al incrementar el número de puntos se aprecia mejor el aspecto de la gráfica.

Otra opción útil para modificar la malla es **grid**, la cual nos permite especificar exactamente la cantidad de puntos a usar para cada uno de los intervalos de graficación, en forma de una lista. Para ejemplificar esto

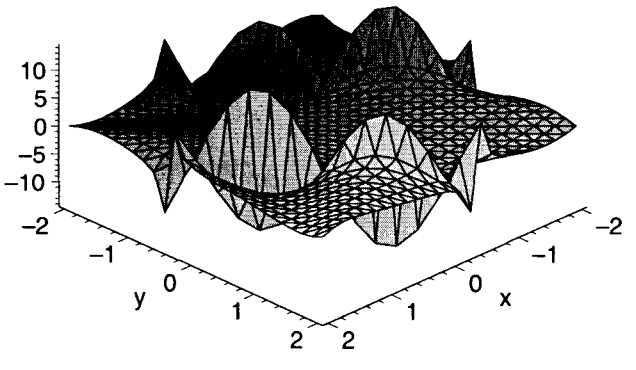
desplegaremos nuevamente la gráfica anterior con una malla dada por **grid=[40, 50]**; esto es, 40 puntos para el eje **x**, y 50 puntos para el eje **y**.

```
> plot3d(sin(x^2 - y^2)/cos(x + y), x=-2..2, y=-2..2, axes=framed,
> grid=[40, 50], view=-50..50);
```



En esta última gráfica tenemos un mejor aspecto de la función. Otra opción que tenemos disponible es **gridstyle**, la cual nos permite especificar el tipo de malla a utilizar, los valores que puede tomar son **rectangular** y **triangular**. El tipo utilizado como predeterminado es **rectangular**. Al utilizar la opción **triangular**, cada rectángulo formado por los puntos de la malla es dividido en dos triángulos, esto en ciertos casos nos puede generar una mejor vista de la gráfica de una función.

```
> plot3d(sin(x^2 - y^2)/cos(x + y), x=-2..2, y=-2..2, axes=framed,
> gridstyle=triangular);
```



Esta última opción puede ser aplicada directamente a la gráfica desde el submenú **Grid Style** del menú **Style**, el cual aparece en la barra de menús para gráficas en tres dimensiones. Esto mismo puede hacerse a través del menú contextual que aparece al oprimir el botón derecho sobre la gráfica.

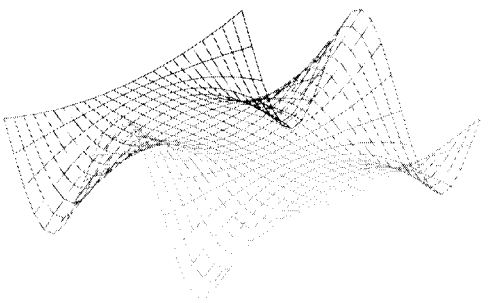
12.3.4 Modificación del estilo de la superficie

La opción **style**, nos permite especificar diferentes estilos para desplegar la superficie de la gráfica. Estos estilos, además de poder incluirlos en forma de opciones, pueden ser aplicados directamente sobre la gráfica desde el menú **Style** de la barra de menús para gráficas en tres dimensiones, así como también por medio de los botónes de la barra contextual y por medio del menú contextual desplegado al oprimir el botón derecho del ratón sobre una gráfica. A continuación describiremos la forma de aplicar cada uno de ellos desde los menús y desde la barra contextual.

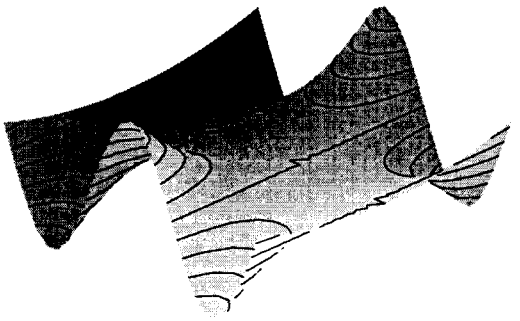
- **PATCH**. Este es el estilo predeterminado, despliega la superficie junto con la malla utilizada; es equivalente a la opción **LINE**. Puede ser también aplicado directamente sobre la gráfica mediante la opción **Patch** del menú **Style**, en la barra de menús para gráficas en tres dimensiones, o por medio del botón  , de la barra contextual correspondiente.
- **PATCHNOGRID**. Despliega la superficie sin la malla. Puede ser aplicada por medio de la opción **Patch w/o grid**, del menú **Style** o por medio del botón  , de la barra contextual.
- **WIREFRAME**. Despliega unicamente la malla. Esta opción puede aplicarse por medio de la opción **Wireframe** del menú **Style**, o por medio del botón  , de la barra contextual.
- **HIDDEN**. Despliega unicamente la malla, pero sin mostrar las líneas ocultas. Puede aplicarse directamente por medio de la opción **Hidden line** del menú **Style**, o por medio del botón  , de la barra contextual.
- **CONTOUR**. Despliega varias curvas de nivel de la función, sin desplegar la superficie. Puede aplicarse por medio de la opción **Contour**, del menú **Style**, o a través del botón  , de la barra contextual.
- **PATCHCONTOUR**. Despliega la superficie y las curvas de nivel. Puede aplicarse por medio de la opción **Patch and contour**, del menu **Style**, o a través del botón  , de la barra contextual.
- **POINT**. Despliega la superficie usando puntos. Puede aplicarse mediante la opción **Point**, del menú **Style**, o por medio del botón  , de la barra contextual.

Estos nombres de estilos pueden ser escritos tanto en minúsculas como en mayúsculas. Por ejemplo:

```
> plot3d(x^2*cos(y), x=-5..5, y=-5..5, style=WIREFRAME);
```



```
> plot3d(x^2*cos(y), x=-5..5, y=-5..5, style=PATCHCONTOUR);
```



La opción **Style** puede combinarse con las siguientes opciones:

- **contours**. Nos permite indicar las curvas de nivel que se desplegarán. Esta opción puede usarse en la forma **contours=n**, para desplegar **n** curvas de nivel distribuidas uniformemente, o bien en la forma **contours=[a, b, c, ...]**, para desplegar unicamente las curvas de nivel correspondientes a los valores **a**, **b**, **c** ..., del eje **z**.

- **linestyle**. Especifica el estilo a utilizar para las líneas desplegadas en la gráfica. Puede tomar valores numéricos entre 1 y 4, o bien las palabras correspondientes: **SOLID**, **DOT**, **DASH**, y **DASHDOT**. las cuales especifican lineas sólidas, de puntos, con guiones o con guiones y puntos, respectivamente. Esta opción puede aplicarse directamente a la gráfica por medio de las opciones del submenú **Line Style**, del menú **Style**.

- **thicknes**. Especifica el grosor de las líneas desplegadas. Los valores que puede tomar son **0**, **1**, **2** y **3**. El valor por omisión es **0**. Puede aplicarse a la gráfica por medio de las opciones del submenú **Line Width** del menú **Style**.

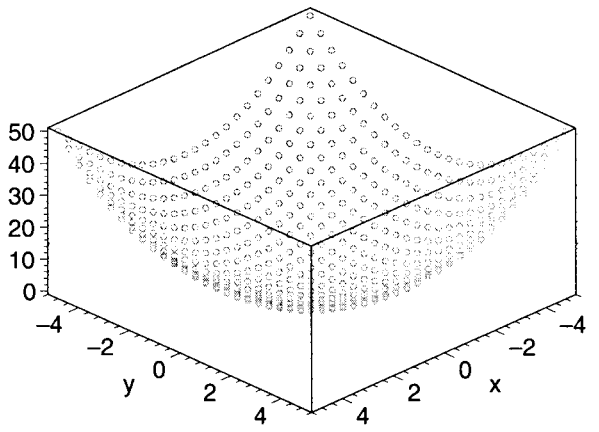
- **symbol**. Permite especificar el tipo de símbolo a usar para los puntos desplegados en la gráfica. Puede tomar los valores: **BOX**, **CROSS**, **CIRCLE**, **POINT** y **DIAMOND**. El valor predeterminado es **POINT**. Puede aplicarse a la gráfica por medio de las opciones del submenú **Symbol** del menú **Style**.

- **symbolsize**. Especifica el tamaño de cada símbolo usado en la gráfica (de los mencionados en la opción anterior). El valor predeterminado es de 10 puntos. También puede especificarse por medio de la opción **Symbol size** del menú **Style**.

Las últimas cuatro opciones listadas también pueden ser aplicadas por medio del menú contextual desplegado al oprimir el botón derecho del ratón sobre una gráfica.

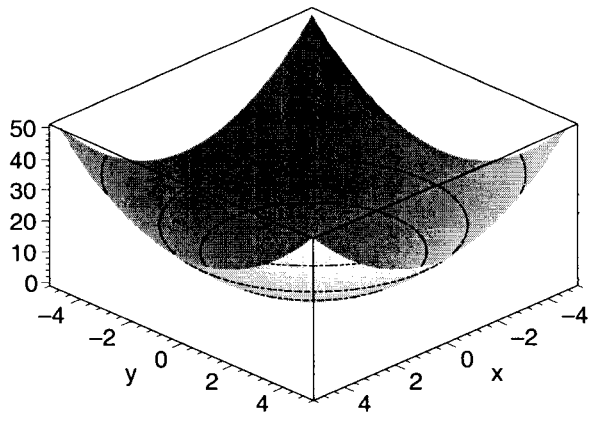
Por ejemplo, desplegaremos la siguiente gráfica con puntos y usaremos como símbolo un círculo de tamaño 15 para cada uno de ellos.

```
> plot3d(x^2 + y^2, x=-5..5, y=-5..5, style=POINT, symbol=CIRCLE,
> symbolsize=15, axes=BOXED);
```



A continuación desplegaremos la misma gráfica mostrando la superficie y las curvas de nivel para z = 2, 3.5 y 4, usando una línea punteada.

```
> plot3d(x^2 + y^2, x=-5..5, y=-5..5, style=patchcontour,
> contours=[10, 18.5, 35], linestyle=DOT, axes=BOXED);
```



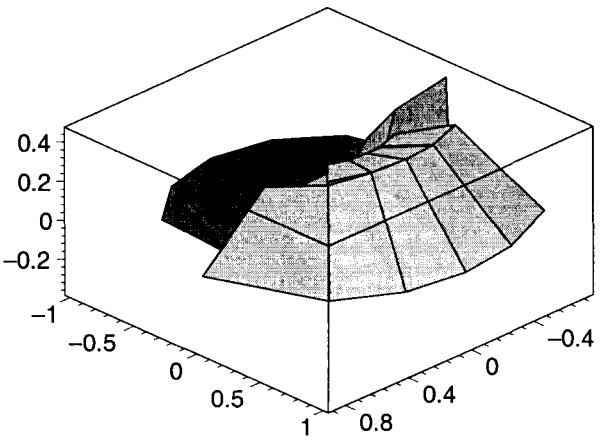
12.3.5 Especificación de un sistema de coordenadas

La opción **coords** nos permite especificar el sistema de coordenadas que se debe utilizar para desplegar la gráfica. El sistema predeterminado es el Cartesiano (Cartesian), otros sistemas que pueden usarse son:

bipolarcylindrical, bispherical, cardioidal, cardioidcylindrical, casscylindrical, confocalellip, confocalparab, conical, cylindrical, ellcylindrical, ellipsoidal, hypercylindrical, invcasscylindrical, invellcylindrical, invoblspheroidal, invprospheroidal, logcoshcylindrical, logcylindrical, maxwellcylindrical, oblatespheroidal, paraboloidal, paracylindrical, prolatespheroidal, rosecylindrical, sixsphere, spherical, tangentcylindrical, tangentsphere, y toroidal.

Por ejemplo, desplegaremos la siguiente gráfica utilizando un sistema de coordenadas cónicas.

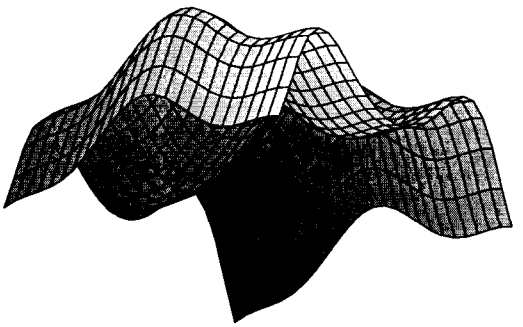
```
> plot3d(x^2 + y^3, x=-2..2, y=-2..2, coords=conical, axes=boxed);
```



12.3.6 Orientación de la gráfica

La opción **orientation=[theta, phi]**, permite especificar el punto de vista para la gráfica, es decir, determina la posición en que se mostrará la gráfica al usuario. Este punto debe ser dado en coordenadas esféricas, donde **theta** y **phi** son ángulos en grados, el valor predeterminado es 45 en ambos casos. Por ejemplo, desplegaremos la siguiente gráfica de tal forma que podamos tener una vista de ella desde la parte negativa del eje z:

```
> plot3d(sin(x^2) + cos(y)^3, x=-2..2, y=-2..2, orientation=[40,
> 130]);
```



Esta orientación es modificada automáticamente al mover la gráfica. También puede ser especificada por medio de los menús:

Los cuales aparecen en la barra contextual para regiones gráficas en tres dimensiones.

12.3.7 Otras opciones

A continuación se describen algunas otras opciones que pueden aplicarse a gráficas tridimensionales.

- Diferentes tipos de fuentes. Podemos especificar diferentes tipos de fuentes para los objetos de texto que se despliegan en la gráfica, por medio de la opción **font**. El formato de esta opción es el mismo que se describió para gráficas en dos dimensiones. Por ejemplo, la opción **font=[TIMES, BOLDITALIC, 15]**, especifica una fuente de tipo **TIMES**, en estilo negrita cursiva y tamaño 15.

- Colocar un título a la gráfica. La opción **title**, nos permite especificar un título para la gráfica. Este puede ser dado en la forma:

  "Cadena del título"

  "Cadena del título \n Cadena del subtítulo \n Cadena del sub-subtítulo..."

  Los subtítulos son desplegados cada uno debajo del anterior.

  Esta opción puede combinarse con **titlefont=f**, para especificar el tipo de fuente para el título, el cual puede ser expresado de la misma forma que en **font**.

- Colocar etiquetas en los ejes. Esto puede hacerse a través de la opción:

  labels=[*"etiqueta x", "etiqueta y", "etiquetaz"*]

  También puede especificarse el tipo de fuente a usar con la opción **labelfont=f**, donde la fuente debe indicarse de la misma forma que en **font**. Otra opción con la cual puede combinarse es con **labeldirections=[dirx, diry, dirz]**, la cual especifica la dirección en la que debe desplegarse cada etiqueta. Los valores que puede tomar son **HORIZONTAL** y **VERTICAL**, el valor por omisión es **HORIZONTAL**. Por ejemplo, las opciones:

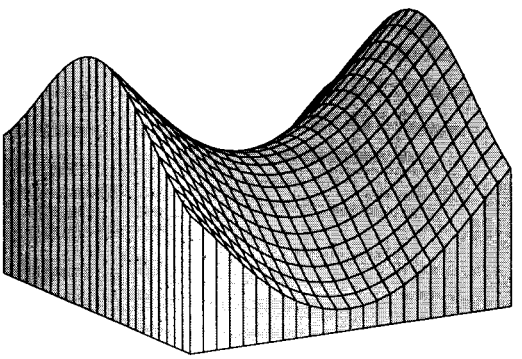
  labels=["Población", "Tiempo", "Crecimiento"], labelfont=[TIMES, ITALIC, 15],

  labeldirections=[HORIZONTAL, HORIZONTAL, VERTICAL]

  Desplegarán las etiquetas **"Población"**, **"Tiempo"** y **"Crecimiento"** en los ejes x, y y z, respectivamente; utilizando la fuente **TIMES**, en estilo cursiva y tamaño 14. Además, las etiquetas de x e y se desplegarán en forma horizontal, mientras que para el eje z aparecerá en forma vertical.

- **filled=true/false**. El valor por omisión de esta opción es **false**; si se especifica el valor **true**, la gráfica es desplegada en forma de un solido delimitado por el plano xy y la gráfica de la función. Por ejemplo:

```
> plot3d(sin(x^2) + cos(y)^3, x=-1..1, y=-1..1, filled=true,
> orientation=[60,70]);
```



- **ambientlight=[r, g, b]**. Permite especificar el color de la luz ambiental a través de sus componentes rojo, verde y azul; las cuales deben estar dadas por números entre 0 y 1.
- **axesfont=f**. Establece el tipo de fuente a usar para los números que se despliegan a lo largo de los ejes. El tipo de fuente debe especificarse como en la opción **font**.
- **light=[phi, theta, r, g, b]**. Esta opción agrega una luz del color especificado por **"r"**, **"g"** y **"b"**, en la dirección dada por **"phi, theta"** en coordenadas esfericas. Las componentes rojo, verde y azul deben ser números entre 0 y 1, mientras que los ángulos **phi** y **theta** deben ser dados en grados.
- **lightmodel=modelo**. Existen varios modelos de luces predefinidos en Maple, los cuales pueden ser aplicados a las gráficas en tres dimensiones por medio de esta opción. Los nombres de estos modelos son **'none'**, **'light1'**, **'light2'**, **'light3'** y **'light4'**. Éstos también pueden ser aplicados a la gráfica por medio de las opciones del menú **Color**, en la barra de menús para gráficas en tres dimensiones.
- **projection=n**. Permite especificar la perspectiva con la que se visualizará la superficie. **n** puede tomar valores entre 0 y 1, o bien, pueden especificarse los nombres **'FISHEYE'**, **'NORMAL'** y **'ORTHOGONAL'**. El valor predeterminado es **projection=ORTHOGONAL**, el cual corresponde al valor 1.
- **shading=s**. Esta opción determina como es coloreada la superficie. Los valores que puede tomar **s** son: **XYZ**, **XY**, **Z**, **ZGRAYSCALE**, **ZHUE** y **NONE**.

Muchas de estas opciones también pueden ser aplicadas directamente desde el menú contextual, desplegado al oprimir el botón derecho del ratón sobre una gráfica.

12.4 Funciones paramétricas

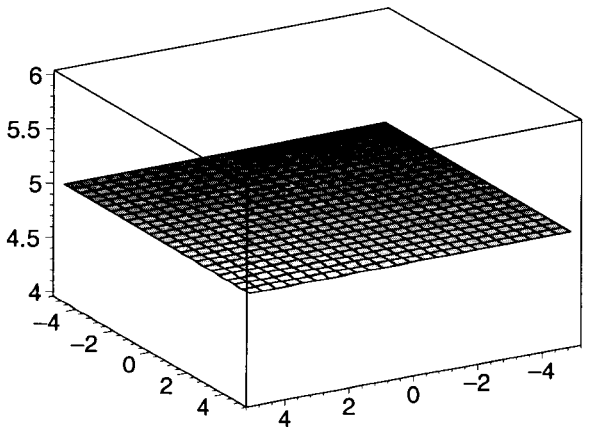
Las gráficas de funciones paramétricas pueden ser desplegadas mediante **plot3d**, introduciendo las componentes paramétricas en forma de una lista, es decir:

plot3d([fx, fy, fz], x=a..b, y=c..d, opciones);

Veamos algunos ejemplos:

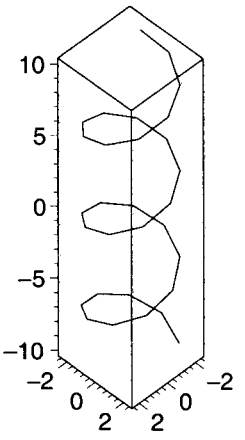
- Gráfica del plano **z=5**.

```
> plot3d([x, y, 5], x=-5..5, y=-5..5, axes=boxed,
> orientation=[60, 60]);
```



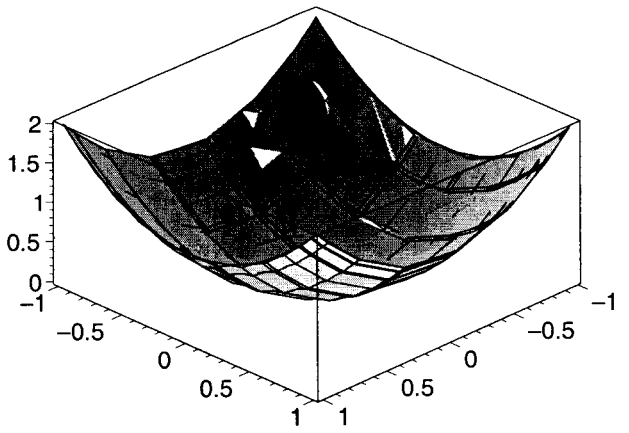
- Gráfica de una espiral de radio 2.5, alrededor del eje **z**.

```
> plot3d([2.5*cos(x), 2.5*sin(x), x], x=-10..10, y=-10..10, axes=boxed,
> scaling=CONSTRAINED, grid=[30,30]);
```



- Gráfica de un paraboloide.

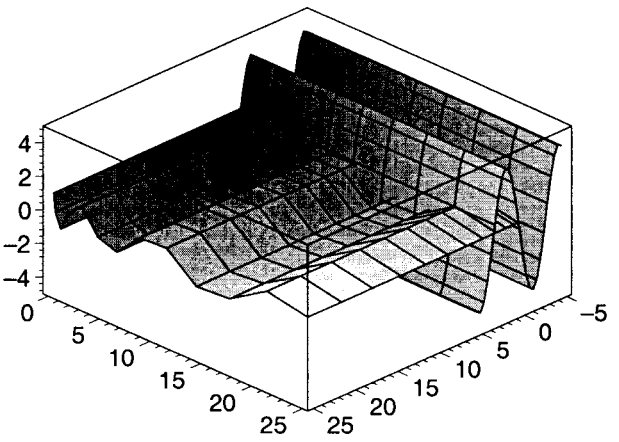
```
> plot3d([cos(x), sin(y), cos(x)^2 + sin(y)^2], x=-5..5,
> y=-5..5, axes=boxed);
```



Esta instrucción también nos da la posibilidad de desplegar varias funciones paramétricas en una misma gráfica, expresándolas en forma de conjunto.

Para ejemplificar esto veamos la siguiente gráfica:

```
> plot3d({[x^2, y^2, cos(4*y)], [x, y^2, 4*sin(x)]}, x=-5..5, y=-5..5,
> view=-5..5, axes=boxed);
```



12.5 Gráficas en tres dimensiones con el paquete plots

El paquete **plots** nos proporciona un conjunto de funciones especializadas para el despliegue de diversos tipos de gráficas, en particular contiene un conjunto de instrucciones útiles en el despliegue de gráficas en tres dimensiones. Antes de hacer uso de ellas, debemos cargar éste paquete en memoria con la instrucción **with**.

```
> with(plots):
```

A continuación, haremos una revisión de las principales funciones proporcionadas por este paquete, especiales para el despliegue de gráficas en tres dimensiones.

12.5.1 Coordenadas cilindricas

Este tipo de gráficas pueden ser desplegadas por medio de la función **cylinderplot**. Su sintaxis es:

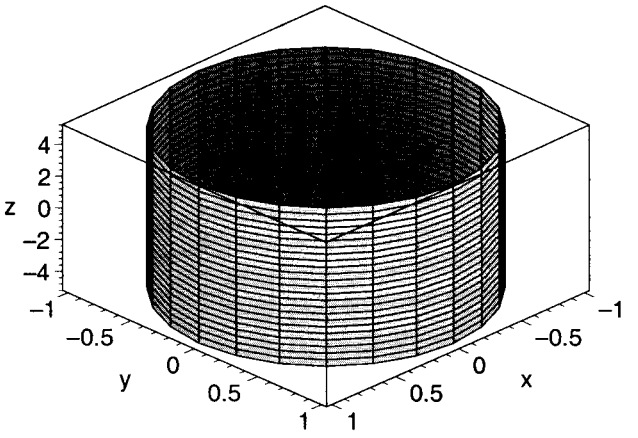
cylinderplot(expr, i1, i2,opciones);

Donde **expr** puede ser una función o expresión explícita de dos variables o bien una función paramétrica. Si la función es dada en forma paramétrica, ésta es considerada de la forma **[r, theta, z]**, donde **r** representa el radio y **theta** el ángulo; **i1** e **i2** determinan los intervalos para los dos parámetros de la superficie. Si la función es dada en forma explicita, ésta representa el radio dado en términos de **theta** y **z**. En este caso, **i1** e **i2** representan los intervalos para **theta** y **z** respectivamente.

Las opciones aplicables a esta función son las mismas de la función **plot3d**.

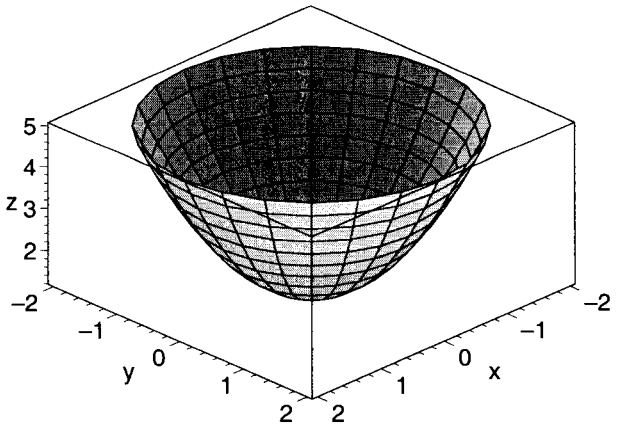
Por ejemplo, a continuación graficaremos un cilindro de **radio=1**.

```
> cylinderplot(1, theta=0..2*Pi, z=-5..5, axes=boxed);
```



En el siguiente ejemplo desplegaremos la gráfica de un paraboloide en coordenadas cilíndricas.

```
> cylinderplot(sqrt(z - 1), theta=0..2*Pi, z=-5..5, axes=boxed);
```



La función **cylinderplot** es equivalente a:

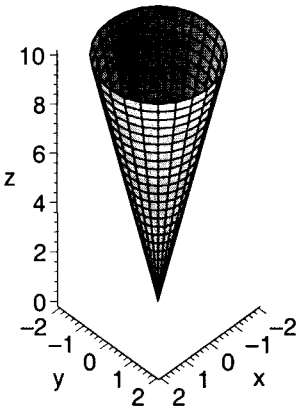
plot3d(expr, i1, i2, coords=cylindrical).

Esta función también puede ser usada para desplegar la gráfica de un sólido de revolución con ***"perfil"*** **z=z(r)**, expresándola en la forma:

[r, phi, z(r)]

Por ejemplo, mostraremos la gráfica del sólido de revolución generado por **z=5*r**.

```
> cylinderplot([r, phi, 5*r], phi=0..2*Pi, r=0..2, axes=framed,
> scaling=constrained);
```



12.5.2 Coordenadas esféricas

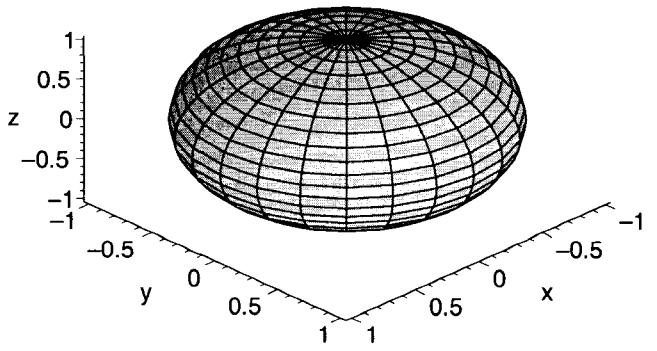
Las gráficas de funciones en coordenadas esféricas pueden ser desplegadas por medio de la función **sphereplot**. Su sintaxis es:

sphereplot(expr, i1, i2, opciones);

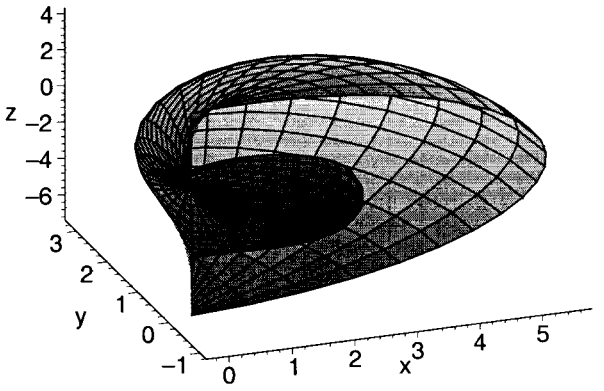
Donde **expr** es una función o expresión explícita o paramétrica en coordenadas esféricas. Si esta función está dada en forma paramétrica, sus componentes representan el radio, el ángulo **theta** y el ángulo **phi**. Si está dada en forma explícita, la función representa el radio en términos de **theta** y **phi**. Por otro lado, **i1** e **i2** representan los intervalos de graficación. Las opciones aplicables a esta función son las mismas válidas para **plot3d**.

Veamos algunos ejemplos:

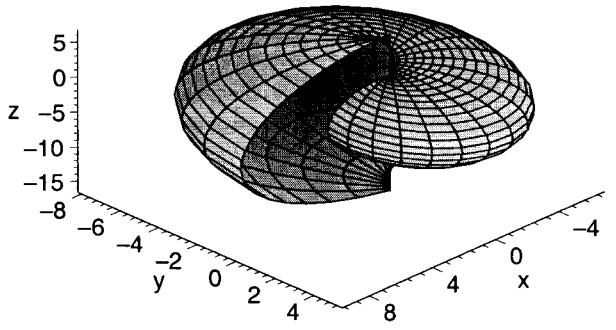
```
> sphereplot(1, theta=0..2*Pi, phi=0..Pi, axes=framed);
```



```
> sphereplot(4*cos(theta) + phi, theta=0..Pi, phi=0..Pi, axes=framed,
> orientation=[-110, 55]);
```




```
> sphereplot([theta + phi^2, theta, phi], theta=0..2*Pi, phi=0..Pi,
> axes=framed);
```



Esta función es equivalente a:

plot3d(expr, i1, i2, coords=spherical).

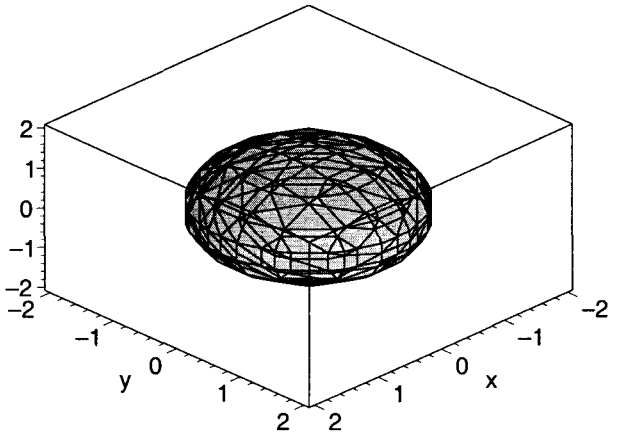
12.5.3 Funciones implícitas

Este tipo de funciones pueden ser graficadas por medio de la función **implicitplot3d**. Su sintaxis es:

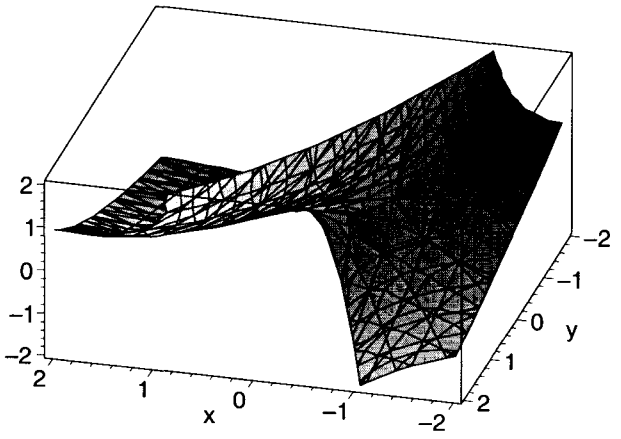
implicitplot3d(expr, x=a..b, y=c..d, z=e..f, opciones);

Donde **expr** es una ecuación en términos de tres variables. Las opciones que se aplican a esta función son las mismas de la función **plot3d**. Por ejemplo, desplegaremos las gráficas de ***x^2 + y^2 = z^2*** y ***sin(x + y)*cos(x + y) = sqrt(x*y)***.

```
> implicitplot3d(x^2 + y^2 + z^2 = 2, x=-2..2, y=-2..2, z=-2..2,
> axes=boxed);
```




```
> implicitplot3d(sin(x + z) = y - x*z, x=-2..2, y=-2..2, z=-2..2,
> axes=boxed, orientation=[105, 45]);
```



12.5.4 Gráficas de puntos

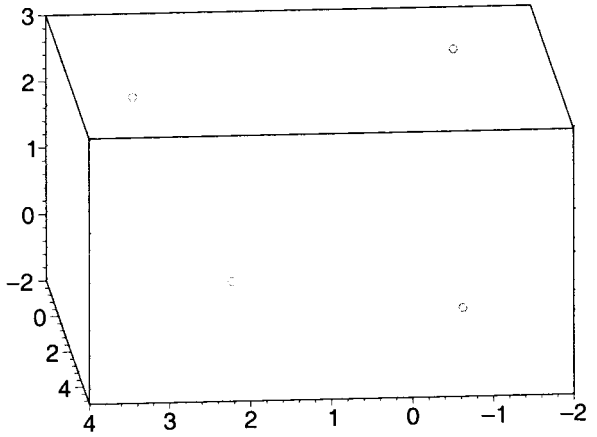
Este tipo de gráficas pueden ser generadas con la instrucción **pointplot**, su sintaxis es:

pointplot3d(L, opciones);

Donde **L** es una lista o conjunto de puntos en tres dimensiones. Las opciones aplicables son las mismas que para **plot3d**.

Veamos los ejemplos que se muestran a continuación:

```
> pointplot3d({[-1, 0, -1], [2, 2, 0], [3, -1, 2], [-1, -1.5, 2.5]},
> axes=BOXED, style=point, symbol=circle, symbolsize=20,
> view=[-2..4, -2..5, -1..3], orientation=[85, 65]);
```

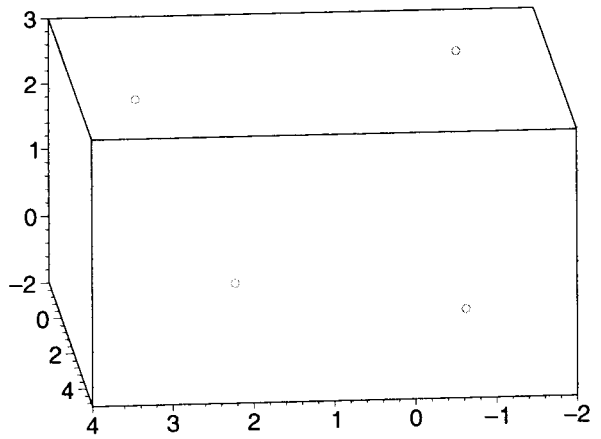


Recuérdese que los puntos de la lista también pueden ser dados en la forma:

[0, 0, 1, 1.5, -1, 2, 3, 4, 2].

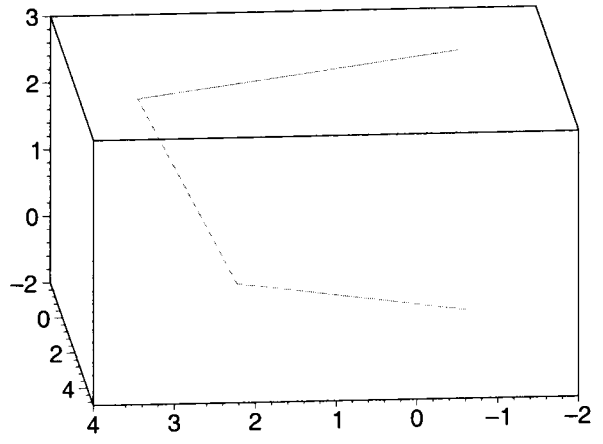
Veamos la siguiente gráfica:

```
> pointplot3d([-1, 0, -1, 2, 2, 0, 3, -1, 2, -1, -1.5, 2.5], axes=BOXED,
> symbol=circle, symbolsize=20, view=[-2..4, -2..5, -1..3],
> orientation=[85, 65]);
```



Esta función soporta también la opción **connect=true**, la cual indica que los puntos deben ser unidos con una línea en el orden en que fueron introducidos.

```
> pointplot3d([-1, 0, -1, 2, 2, 0, 3, -1, 2, -1, -1.5, 2.5], axes=BOXED,
> symbol=circle, symbolsize=20, view=[-2..4, -2..5, -1..3],
> orientation=[85, 65], connect=true);
```



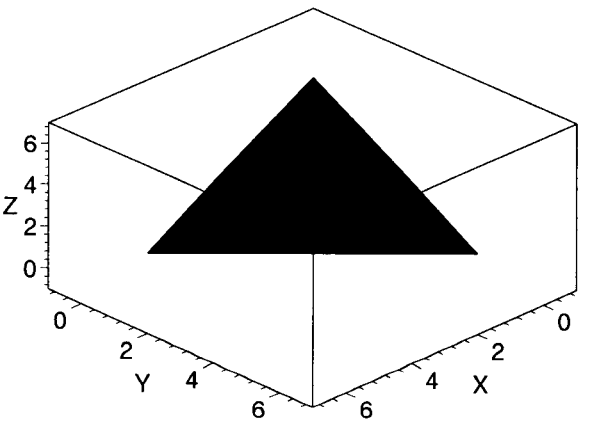
12.5.5 Gráficas de polígonos en el espacio

La función **polygonplot3d** nos permite graficar polígonos en el espacio. Su sintaxis es:

polygonplot3d(L,opciones);

Donde **L** es una lista de puntos (en tres dimensiones) que representan cada uno de los vértices. Las opciones aplicables son las mismas de **plot3d**. Por ejemplo:

```
> polygonplot3d([[0, 5, 0], [5, 0, 0], [0, 0, 5]], axes=boxed,
> view=[-1..7, -1..7, -1..7], labels=["X", "Y", "Z"], color=blue);
```



12.5.6 Gráficas de poliedros

Las gráficas de poliedros pueden ser generadas con la función **polyhedraplot**. La sintaxis de esta función es:

polyhedraplot(L,opciones);

Donde **L** es una lista de puntos en tres dimensiones de la forma:

[[x1, y1, z1], [x2, y2, z2], [x3, y3, z3], ...]

O bien, si solo es un punto:

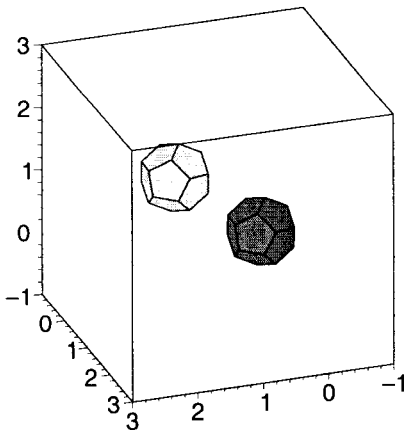
[xi, yi, zi]

Dos de las opciones soportadas son: **polyscale=c**, la cual especifica la escala a la cual se desplegará el poliedro (por omisión 1), y **polytipe**, la cual determina el tipo de poliedro a desplegar; entre los tipos de poliedros validos se encuentran: **tetrahedron**, **hexahedron**, **octahedron**, **OctagonalPrism** e **icosahedron**, el valor predeterminado es **tetrahedron**. Para obtener una lista completa de los poliedros soportados se puede ejecutar la función:

plots[polyhedra_supported]();

Una vez especificado el tipo de poliedro y los puntos en tres dimensiones, la función **polyhedraplot** despliega un poliedro en cada uno de estos puntos. Por ejemplo, desplegaremos dos dodecaedros a escala 0.5, en los puntos **[0, 0, 0]** y **[2, 2, 2]**.

```
> polyhedraplot([[0, 0, 0], [2, 2, 2]], polytype=dodecahedron,
> polyscale=0.5, style=PATCH, scaling=CONSTRAINED, orientation=[71, 66],
> view=[-1..3, -1..3, -1..3], axes=boxed);
```



12.5.7 Curvas en el espacio

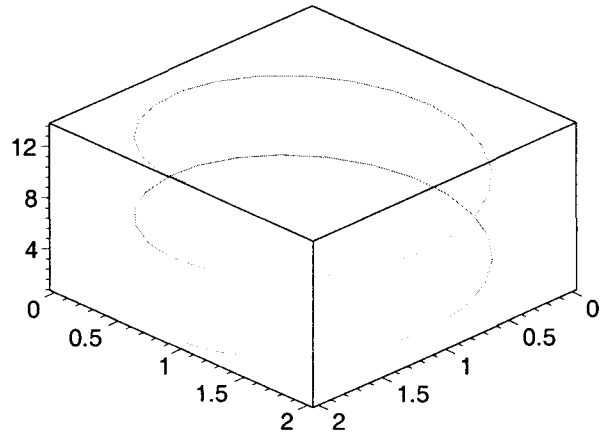
La función **spacecurve** nos permite graficar curvas en el espacio. Su sintaxis es:

spacecurve(L, options);

Donde **L** es una lista conteniendo una parametrización de la curva, una lista de puntos o una lista de curvas. Las opciones aceptadas son las mismas de **plot3d**.

Por ejemplo, graficaremos la siguiente espiral, parametrizada por **[1 + cos(t), 1 + sin(t), 1 + t]**.

```
> spacecurve([1 + cos(t), 1 + sin(t), 1 + t], t=0..4*Pi, axes=boxed);
```



El uso de esta función es similar a utilizar **plot3d** para funciones paramétricas, pero en el caso de **spacecurve** solo es necesario incluir un parámetro para generar la gráfica.

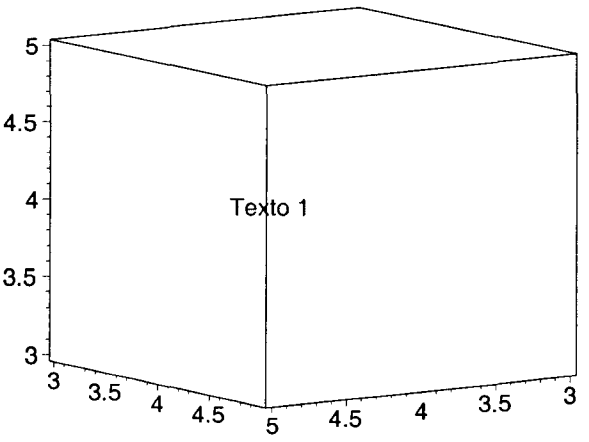
12.5.8 Gráficas de texto

Este tipo de gráficas pueden ser generadas por medio de la función **textplot3d**, su sintaxis es:

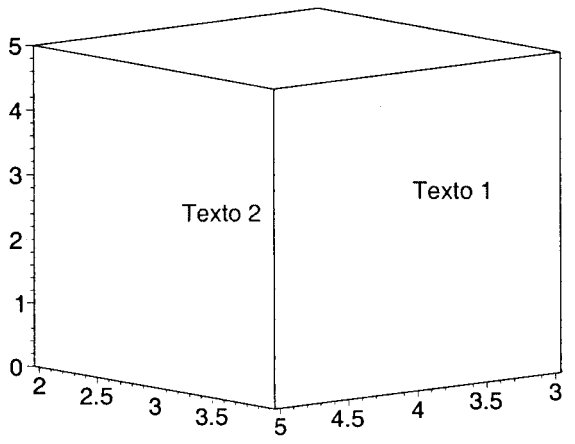
textplot3d(L, opciones);

Donde **L** es de la forma **[xi, yi, zi, "texto"]**, o bien una lista de listas de este tipo. Esta función despliega la cadena de texto recibida, en el punto dado por **xi**, **yi** y **zi**. Las opciones aceptadas son las mismas de **plot3d**; adicionalmente se puede incluir la opción **align**, la cual especifica la forma en la que debe alinearse el texto con respecto al punto. Las opciones de alineación son: **BELOW**, **RIGHT**, **ABOVE** y **LEFT**, y pueden ser usadas individualmente o combinadas. Por ejemplo:

```
> textplot3d([4, 4, 4, "Texto 1"], align=LEFT, color=blue, axes=boxed,
> orientation=[55, 80]);
```



```
> textplot3d([[4, 4, 3, "Texto 1"], [4, 2, 2, "Texto 2"]],
> align={RIGHT, ABOVE}, color=blue, axes=boxed, orientation=[50, 80],
> view=0..5);
```



12.5.9 Superposición de gráficas con estructuras diferentes

Cuando tenemos varias gráficas en tres dimensiones con características diferentes, éstas pueden ser deplegadas en una sola por medio de la instrucción **display3d**, introduciéndolas como conjunto o lista. Muchas de las opciones de **plot3d** pueden ser aplicadas a esta función.

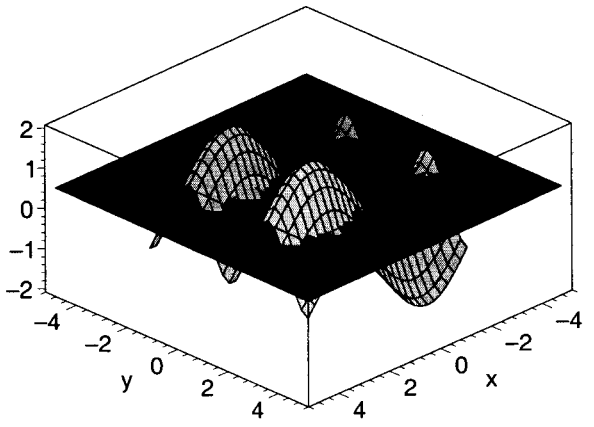
display3d es un alias para la función **display**, por lo cual puede ser usada también con este nombre.

Veamos un ejemplo:

```
> g1 := plot3d(sin(x) - cos(2*y), x=-Pi..Pi, y=-Pi..Pi):
> g2 := plot3d([x, y, 0.5], x=-5..5, y=-5..5, color=red, numpoints=100):
```

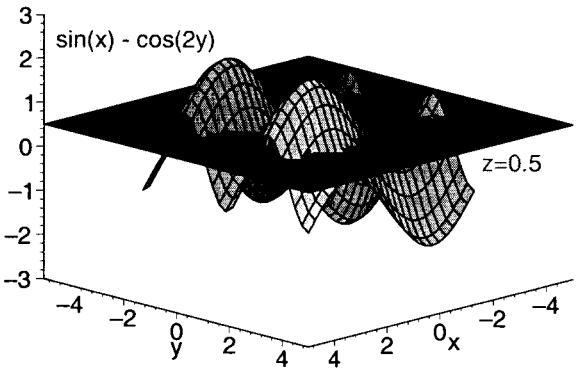
Ahora las mostraremos en un mismo despliegue, con la opción **axes=boxed**.

```
> display3d({g1, g2}, axes=boxed);
```



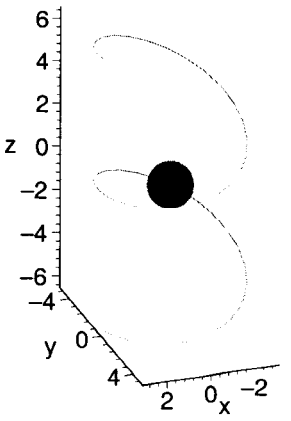
Podemos incluir también gráficas de texto.

```
> t1 := textplot3d([0, -3, 2, "sin(x) - cos(2y)"], align=LEFT,
> color=blue):

> t2 := textplot3d([-5, 4, -0.5, "z=0.5"], align=LEFT, color=red):

> display3d({g1, g2, t1, t2}, axes=framed, orientation=[45, 70],
> view=[-5..5, -5..5, -3..3]);
```



Veamos otro ejemplo:

```
> p1 := spacecurve([3*cos(t), 5*sin(t), t], t=-2*Pi..2*Pi):

> p2 := sphereplot(1, theta=0..Pi, phi=0..2*Pi, color=red,
> style=PATCHNOGRID):

> display3d({p1, p2}, scaling=constrained, axes=framed,
> orientation=[70, 65]);
```

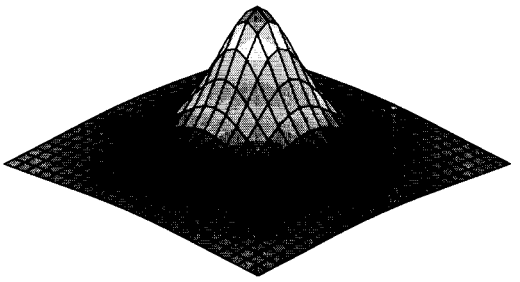


A continuación desplegaremos las gráficas de la función **3/(1 + x^2 + y^2)** y los planos **z=1**, **z=2** y **z=3**.

```
> gr := plot3d(3/(1 + x^2 + y^2), x=-3..3, y=-3..3, style=PATCH,
> shading=ZHUE):

> p1 := plot3d(1, x=-3..3, y=-3..3, shading=ZHUE, numpoints=100):

> p2 := plot3d(2, x=-3..3, y=-3..3, shading=ZHUE, numpoints=100):

> p3 := plot3d(3, x=-3..3, y=-3..3, shading=ZHUE, numpoints=100):
```

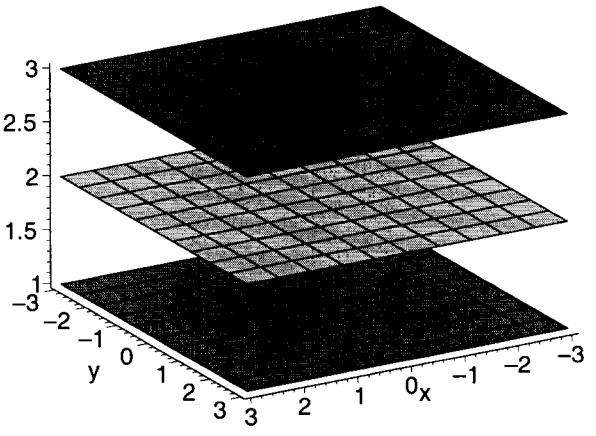
Desplegaremos primero la gráfica de la función:

```
> display3d(gr);
```



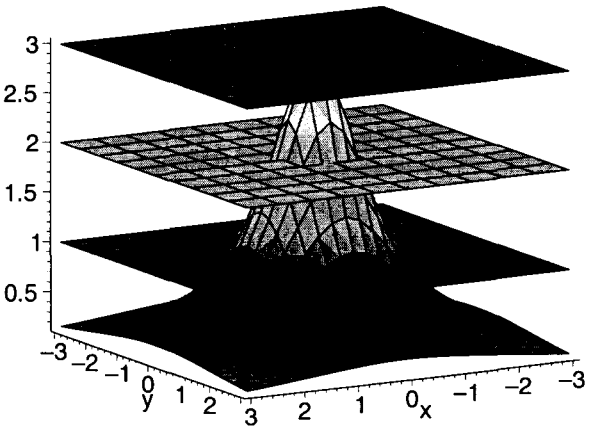
Ahora los planos:

```
> display3d({p1, p2, p3}, axes=framed, orientation=[60, 60]);
```



Y finalmente la gráfica de la función con los planos:

```
> display3d({gr, p1, p2, p3}, axes=framed, orientation=[60,75]);
```



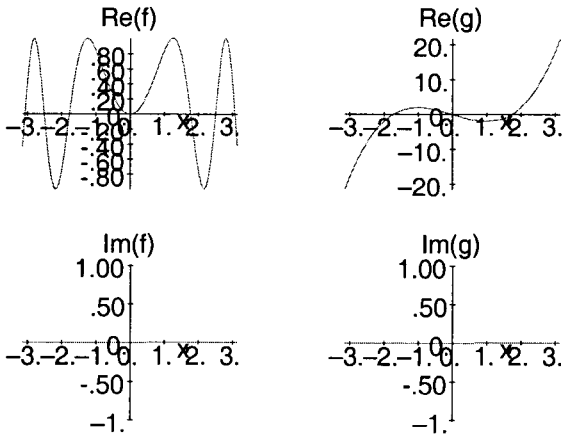
12.5.10 Comparación de gráficas

Maple 8 proporciona una nueva instrucción dentro del paquete **plots**, ésta es **plotcompare**, la cual nos permite generar un despliegue en el cual se muestran las gráficas de las partes real e imaginaria de dos funciones, con el fin de poder compararlas. Su sintaxis es:

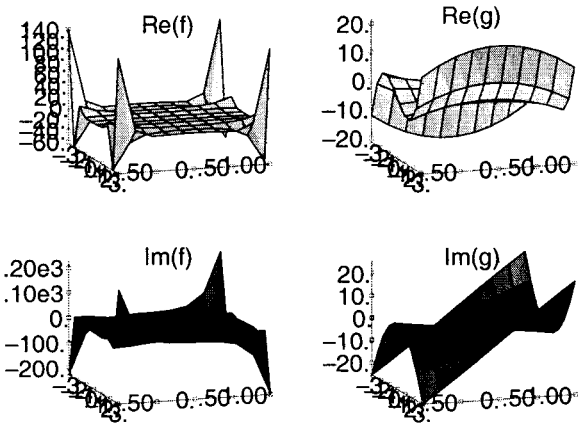
plotcompare(f(x), g(x), x=a..b, opciones);

Por ejemplo:

```
> plots[plotcompare](sin(x^2), x^3 - 3*x, x=-Pi..Pi);
```



```
> plots[plotcompare](sin(x^2), x^3 - 3*x, x=-I-Pi..I+Pi);
```



Existen algunas otras funciones dentro de este paquete. Para obtener una lista completa de ellas, así como vínculos a sus páginas de ayuda, se puede utilizar la instrucción: **?plots**

12.6 Creación de gráficas en modo interactivo

De manera análoga al caso de las funciones en dos dimensiones, la función **interactive** del paquete **plots** nos permite generar una gráfica en tres dimensiones por medio del **Interactive Plot Builder**. Para ello debemos invocar esta función en la forma: **interactive(f(x, y))**. Esto desplegará la ventana de esta herramienta, en la cual debemos seleccionar la opción **3-D plot** y oprimir el botón **Next**, a continuación se desplegará la ventana que se muestra en la Figura 12.1:

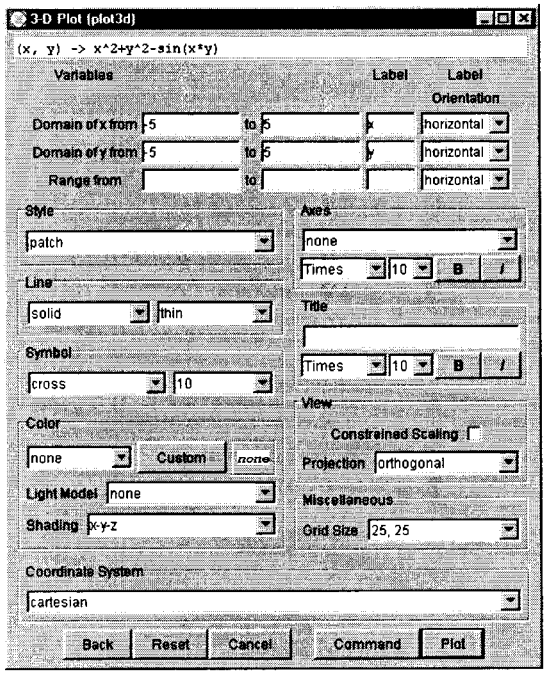


Figura 12.1: Ventana del Interactive Plot Builder

En ésta podemos seleccionar todas las opciones que deseamos aplicar a nuestra gráfica; por ejemplo, los intervalos de graficación, el tipo de superficie, los colores, el sistema de coordenadas, la proyección, etcétera. Una vez establecidas todas las características de nuestra gráfica, oprimimos el botón **Plot** para generar ésta.

Por ejemplo, para generar una gráfica de la función **x^2 + y^2 - sin(x*y)**, debemos ejecutar la instrucción:

plots[interactive](x^2 + y^2 - sin(x*y))

La ventana del **Interactive Plot Builder** también nos permite generar otro tipo de gráficas, por ejemplo: gráficas implicitas, de complejos y de contornos; tanto en dos como en tres dimensiones, y gráficas de densidades, entre otras.

12.7 Creación de gráficas a partir de una expresión de salida

De la misma forma que se describió para las gráficas en dos dimensiones, podemos seleccionar una expresión de salida (o parte de esta) y generar una gráfica de ella; para ello oprimimos el botón derecho del ratón sobre la selección y usando el submenú **Plots** que aparece, seleccionamos la opción **3-D Plot - *x,y***, las cuales se muestran en la Figura 12.2.

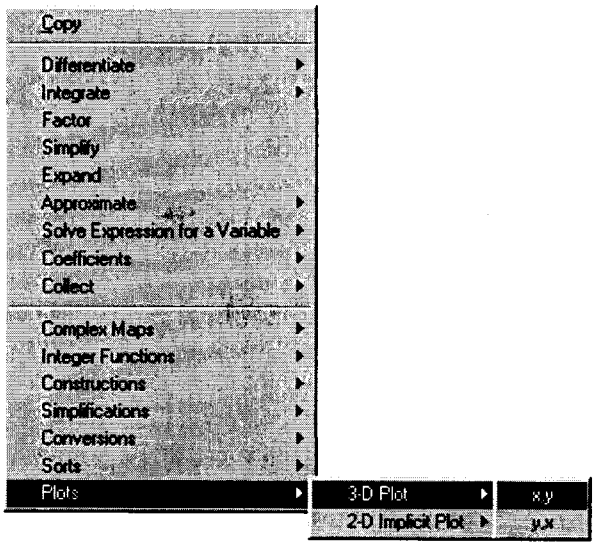


Figura 12.2: Menú para creación de imagenes

Esta opción generará automáticamente una gráfica de la expresión seleccionada (este despliegue se genera invocando a la función **smartplot3d**).

Ademas, esta versión de Maple nos proporciona una manera más para el despliegue de gráficas en tres dimensiones a partir de una expresión de salida. Seleccionamos, en el submenú **Plot** del menú **Insert** la opción **3-D**; esto insertará una región gráfica vacía en nuestra hoja de trabajo. Para poder desplegar la gráfica de una función en esta región insertada, seleccionamos, de una región de salida, una expresión o una parte de ésta y, por medio de las opciones del menú **Edit**, copiamos y pegamos esta expresión dentro de la región gráfica vacia; Maple automáticamente generará una gráfica de la expresión seleccionada. De esta misma forma, en el mismo despliegue podemos copiar y pegar varias expresiones, con lo cual obtendremos una gráfica que incluya todas éstas.

Capítulo 13

Animaciones

Una animación en Maple es una secuencia de gráficas que se muestran en sucesión dentro de una región de animación; estas regiones son semejantes a las de gráficas en dos y tres dimensiones, pero incluyen en la barra de contexto botones para controlar la animación. En Maple, las animaciones pueden crearse, tanto en dos como en tres dimensiones, de dos maneras diferentes. Una de estas formas es por medio de las funciones **animate** y **animate3d**, las cuales generan automáticamente las gráficas de la animación a partir de los argumentos recibidos. La otra forma es generando manualmente las gráficas y desplegandolas en sucesión por medio de **display** o **display3d**. A continuación revisaremos ambas formas.

13.1 Animaciones en dos dimensiones

13.1.1 Animaciones creadas con animate

La función **animate** nos permite crear animaciones de gráficos bidimensionales, producidos por funciones de una variable que dependen de un parámetro adicional, el cual hace el papel del tiempo. Esta función genera un conjunto de gráficas bidimensionales, las cuales se muestran en sucesión para simular la animación. **animate** forma parte del paquete **plots**. Su sintaxis es:

animate(expr, x=a..b, t=c..d, opciones);

animate([f(x,t), g(x,t), h(x,t), ...], x=a..b, t=c..d, opciones);

Donde **expr** puede ser una función o expresión que depende de una variable y del tiempo, dada en forma explícita o paramétrica. También se pueden incluir un conjunto de funciones como argumento. Las opciones aceptadas son las mismas de **plot**; adicionalmente, se puede incluir la opción **frames=n**, donde **n** es el número de gráficas que deben generarse en la animación, el valor por omisión es 16. Veamos un ejemplo:

Generaremos una animación que nos muestre el comportamiento de una familia de funciones de la forma **sin(x*t)**, en el intervalo **[-Pi..Pi]**, para **t** de 1 a 10.

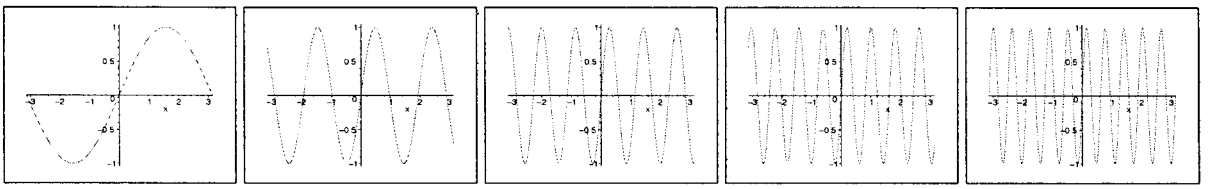
Primero cargamos la función con **with**.

```
> with(plots, animate);
```

[*animate*]

Y a continuación generamos la animación:

```
> animate(sin(x*t), x=-Pi..Pi, t=1..10);
```



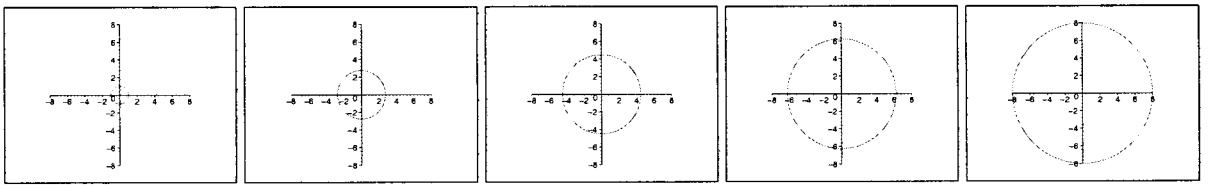
Nota: En todos los ejemplos de animaciones de este capítulo se incluyen varias gráficas, para mostrar al lector el aspecto de las animaciones.

En una región que contiene una animación, la barra contextual nos proporciona un conjunto de elementos por medio de los cuales podemos tener el control de ésta. A continuación se describe la función de cada uno de los componentes de esta barra:

- -2.06, 0.86. Cada vez que el usuario selecciona con el ratón un punto sobre la animación, en esta región se muestran sus coordenadas.
-  . Este botón nos permite detener la animación, una vez iniciada.
-  . Este botón nos permite iniciar la animación.
-  . Por medio de este botón podemos avanzar un cuadro a la vez.
-  . Con este botón podemos hacer que la animación se muestre en sentido inverso.
-  . Este botón nos permite hacer que la animación se muestre en sentido normal.
-  . Este botón nos permite disminuir la velocidad con la que se muestra la animación
-  . Por medio de este botón podemos incrementar la velocidad de la animación.
-  . Este botón nos permite ejecutar la animación un ciclo a la vez.
-  . Con este botón podemos hacer que la animación se muestre en un ciclo continuo.

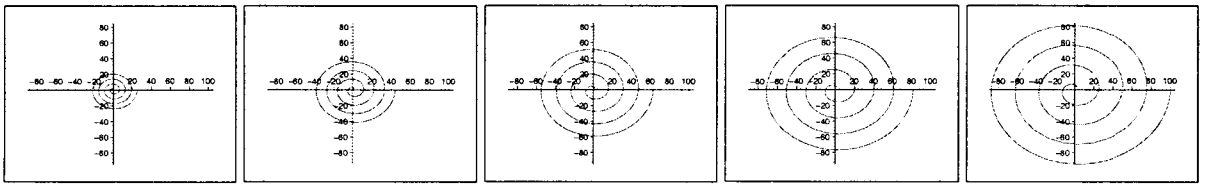
Veamos otro ejemplo, con una función paramétrica.

```
> animate([t*sin(x), t*cos(x), x=-Pi..Pi], t=1..8, view=[-8..8,-8..8]);
```



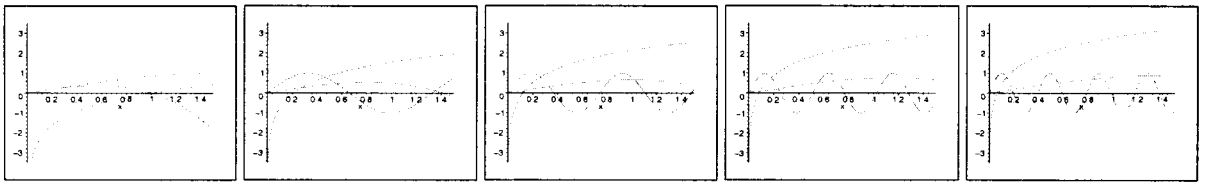
La siguiente animación muestra una función en coordenadas polares. Incluimos la opción **frames=50**, para generar 50 cuadros.

```
> animate([u*t, t, t=1..8*Pi], u=1..4, coords=polar, frames=50,
> numpoints=100, color=blue);
```



Finalmente, tenemos una animación que incluye varias funciones.

```
> animate({x-x^3/u, sin(u*x), ln(x*u)}, x=0..Pi/2, u=1..16, color=gold);
```



13.1.2 Animaciones en 2D por secuencia de gráficas

Otra forma de crear animaciones en dos dimensiones consiste en generar cada una de las gráficas por separado y después mostrarlas en secuencia por medio de la función **display**, agregando para ello la opción **insequence=true**. La función **display** muestra generalmente en un solo despliegue las gráficas que recibe como argumento, al agregar la opción mencionada, las gráficas son mostradas secuencialmente, en el orden en que fueron colocadas como argumento, creando así la animación. Una ventaja de usar este método para crear una animación es que cada uno de los cuadros puede tener una estructura diferente. Por ejemplo, crearemos las gráficas de las funciones **sin(n*x)*exp(-x^2)*n**, para **n** de 1 a 6 y **x** en [-Pi..Pi], cada una con un color diferente.

Antes que nada, cargamos la instrucción **display**, la cual pertenece al paquete **plots**.

```
> with(plots, display);
```

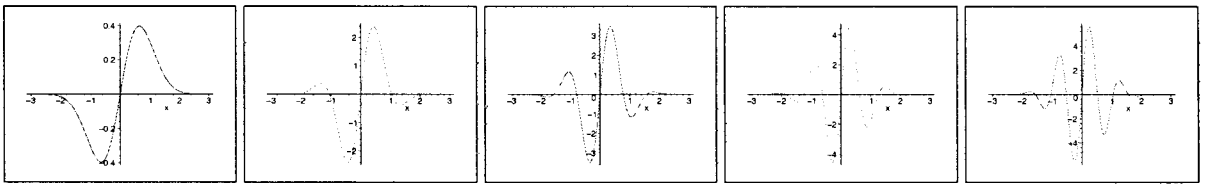
[*display*]

A continuación generamos cada una de las gráficas:

```
> g1 := plot(sin(x)*exp(-x^2), x=-Pi..Pi, color=blue):
> g2 := plot(sin(2*x)*exp(-x^2)*2, x=-Pi..Pi, color=gold):
> g3 := plot(sin(3*x)*exp(-x^2)*3, x=-Pi..Pi, color=green):
> g4 := plot(sin(4*x)*exp(-x^2)*4, x=-Pi..Pi, color=red):
> g5 := plot(sin(5*x)*exp(-x^2)*5, x=-Pi..Pi, color=cyan):
> g6 := plot(sin(6*x)*exp(-x^2)*6, x=-Pi..Pi, color=magenta):
```

Finalmente mostramos la animación comenzando en la gráfica 1 hasta la gráfica 6, y después regresando a la gráfica 1.

```
> display([g1, g2, g3, g4, g5, g6, g5, g4, g3, g2, g1], insequence=true);
```



Las gráficas utilizadas en esta animación, también pudieron haber sido creadas a partir de una secuencia, por ejemplo de la siguiente forma:

Primero generamos una lista para los colores.

```
> lcol := [blue, gold, green, red, cyan, magenta];
```

$$lcol := [blue, gold, green, red, cyan, magenta]$$

A continuación creamos la lista de gráficas usando una secuencia; recuérdese que todas ellas son de la forma **sin(n*x)*exp(-x^2)*n**. Los colores serán tomados de la lista anterior.

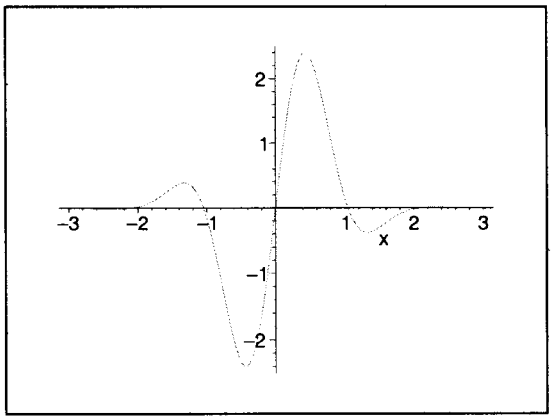
```
> graf := [seq(plot(sin(n*x)*exp(-x^2)*n, x=-Pi..Pi, color=lcol[n]),
> n=1..6)]:
```

Finalmente desplegamos las gráficas en secuencia con **display**, usando la siguiente instrucción:

```
> display(graf, insequence=true);
```

Los elementos de la lista de gráficas pueden ser invocados individualmente; por ejemplo, si queremos visualizar la segunda gráfica de **graf**.

```
> graf[2];
```



De la misma forma podemos referenciarlas para incluirlas en la animación. Por ejemplo, la siguiente instrucción genera la misma animación mostrada arriba:

```
> display([op(graf), graf[5], graf[4], graf[3], graf[2], graf[1]],
> insequence=true);
```

Las regiones de animaciones en dos dimensiones pueden ser manipuladas de la misma forma que las gráficas bidimensionales; por ejemplo, para cambiar el tipo de ejes, o el estilo y símbolo usado en las líneas, entre otras cosas.

13.2 Animaciones en tres dimensiones

13.2.1 Animaciones con animate3d

Análogamente a las animaciones en dos dimensiones, la instrucción **animate3d** nos permite crear animaciones tridimensionales, a partir de funciones que dependen de tres parámetros; los dos primeros son tomados como intervalos de graficación, mientras que el tercero puede ser considerado el tiempo. La sintaxis es:

animate3d(f(x,y,t), x=a..b, y=c..d, t=p..q, opciones);

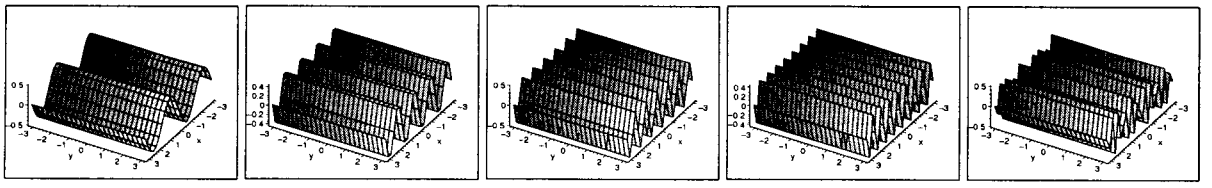
animate3d([f(x,y,t), g(x,y,t), h(x,y,t),...], x=a..b, y=c..d, t=p..q, opciones);

La función puede ser dada en forma explícita, en forma de expresión o bien en forma paramétrica. También se pueden incluir varias funciones en una misma animación. Las opciones aceptadas por **animate3d** son las mismas de **plot3d**. Adicionalmente se puede incluir la opción **frames=n**, para indicar la cantidad de cuadros que formarán la animación. Veamos un ejemplo.

Generaremos la animación de la familia de funciones **cos(t*x)*sin(t*x)**, para **x=-Pi..Pi**, **y=-Pi..Pi**, haciendo variar **t** de 1 a 5.

Antes cargamos la función **animate3d**, del paquete **plots**.

```
> with(plots, animate3d):

> animate3d(cos(t*x)*sin(t*x), x=-Pi..Pi, y=-Pi..Pi, t=1..5,
> axes=framed, orientation=[30,30]);
```

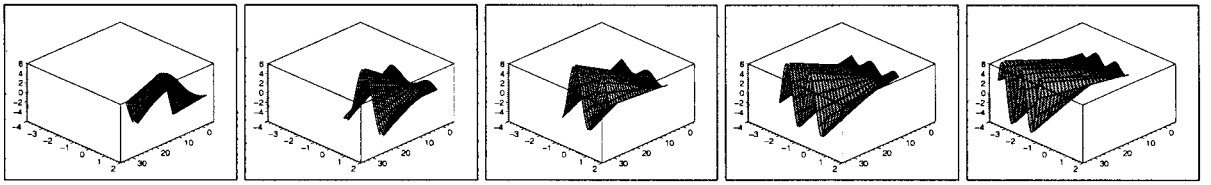


Los botones de la barra de contexto para animaciones tridimensionales tienen el mismo funcionamiento que en el caso de dos dimensiones. La única diferencia es que, en este caso, no es posible seleccionar con el ratón puntos en la gráfica para ver sus coordenadas. En lugar de las coordenadas, aparecen los menús:

Con éstos es posible modificar la orientación de las gráficas de la animación. Lo mismo puede hacerse directamente sobre la región de la animación con el ratón, en la misma forma que se manipulan las gráficas tridimensionales.

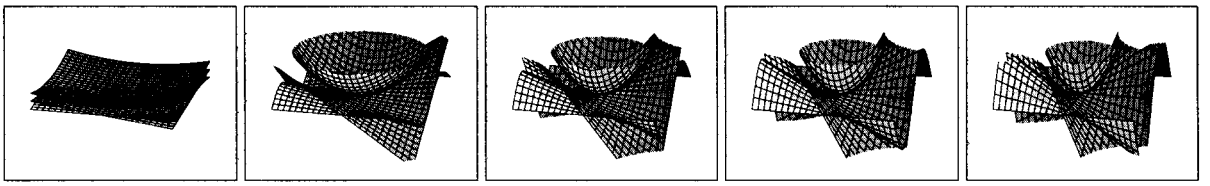
Veamos ahora un ejemplo con una forma paramétrica.

```
> animate3d([t*x, y-t, x*cos(t*y)], x=-1..2*Pi, y=1..4, t=2..5,
> axes=boxed);
```



Por último, tenemos un ejemplo que incluye varias funciones.

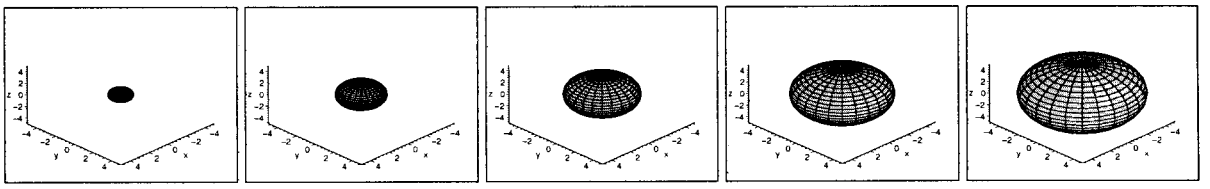
```
> animate3d({t*x*sin(y), t*x*y, t*(x^2 + y^2)}, x=-Pi..Pi, y=-Pi..Pi,
> t=1..20, orientation=[105,60], view=-50..50);
```



13.2.2 Animaciones en 3D por secuencia de gráficas

Otro de los métodos para crear animaciones en tres dimensiones es generar cada una de las gráficas y después mostrarlas en secuencia por medio de la función **display3d**, incluyendo la opción **insequence=true** (de la misma forma que en el caso de dos dimensiones). Por ejemplo, usando la instrucción **sphereplot**, generaremos las gráficas de 5 esferas con centro en el origen, de radio **r=1**, **2**, **3**, **4**, **5**, y las mostraremos en secuencia.

```
> with(plots, sphereplot, display3d);
                              [sphereplot, display3d]

> esferas := [seq(sphereplot(r, theta=0..2*Pi, phi=0..Pi), r=1..5)]:

> display3d(esferas, insequence=true, axes=framed);
```



En el siguiente ejemplo, generaremos una animación para la gráfica de **f(x,y)=(1.3)^x*sin(y)**. En este caso, el parámetro **t** no afecta a la gráfica sino a su orientación, modificaremos el ángulo **theta** de -50 a 300 grados; lo que obtenemos con esto es una animación que muestra la gráfica desde distintos ángulos.

Primero generamos la secuencia. Cada una de las gráficas es de la forma:

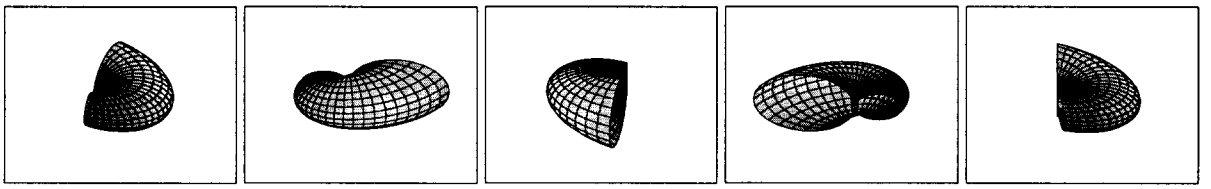
plot3d((1.3)^x*sin(y), x=-1..2, y=0..Pi, coords=spherical, orientation=[ang, 45])

Donde **ang** debe variar de -50 a 300 grados.

```
> anm := [seq(plot3d((1.3)^x*sin(y), x=-1..2, y=0..Pi, coords=spherical,
> orientation=[-50 + 10*t,45]), t=1..35)]:
```

Finalmente mostramos estas gráficas en secuencia.

```
> display3d(anm, insequence=true);
```



Capítulo 14

Álgebra Lineal

Uno de los temas mejor soportados por Maple es el de Álgebra Lineal. Este sistema proporciona dos opciones para realizar cálculos propios de esta materia: el paquete **linalg** y el paquete **LinearAlgebra**. Ambos permiten realizar diversos cálculos sobre matrices y vectores; tales como operaciones aritméticas, cálculo de inversas, transpuestas, determinantes, wronskianos, manipulacion de columnas y renglones; también permiten determinar polinomios característicos, matrices características, valores propios y vectores propios, así como diagonalizacion de matrices, entre otras cosas. Sin embargo, aunque ambos paquetes cuentan con soporte para este tipo de operaciones, en esta versión de Maple se han mejorado notablemente las capacidades y eficiencia de los algoritmos en base a los cuales está construido **LinearAlgebra** (el cual fue concebido como un sustituto de **linalg**), por lo que nos enfocaremos principalmente a éste a lo largo del presente capítulo. No obstante, mencionaremos las características y diferencias principales que existen entre ambos, de tal forma que el usuario pueda elegir cual de los dos paquetes usar.

14.1 Características de linalg y de LinearAlgebra

A continuación revisaremos las principales características de los paquetes antes mencionados. Posteriormente mostraremos las capacidades de **LinearAlgebra** para manejo de matrices y vectores.

14.1.1 El paquete linalg

Este paquete está formado por un conjunto de procedimientos, entre los cuales se incluyen alrededor de cien funciones que permiten realizar una gran diversidad de operaciones de Álgebra Lineal. Las características de este paquete son las siguientes:

- **Los arreglos son la estructura de datos básica.**

  Los vectores y matrices definidos y operados por las funciones de este paquete son manejados en forma de **arreglos**. Estos arreglos deben ser creados a partir de la función **array**, y aunque el paquete proporciona funciones para creación de vectores y matrices, éstas son casos particulares de la función **array**.

- **No proporciona funciones para creación de algunas matrices especiales.**

  linalg no contiene funciones predefinidas para creación, por ejemplo, de matrices cero o matrices identidad; éstas deben ser creadas en forma de arreglos o por medio de las funciones **vector** y ***matrix***. Una alternativa es crearlas a partir de secuencias o algún procedimiento alterno para generación de listas y después convertirlas en vectores o matrices.

- **Evaluación de matrices y vectores.**

  Al ser creados en forma de arreglos, cuando una matriz o vector es asignado a una variable, al colocar este nombre en una línea de comandos, Maple no puede evaluarlo y desplegar su valor, en lugar de esto se despliega el nombre mismo. Para poder desplegar los elementos de una estructura de este tipo se debe usar la función ***eval*** o bien **print**.

- **Álgebra de matrices.**

  Al realizar cálculos algebraicos, Maple no puede evaluar arreglos por medio de los operadores aritméticos normales; es decir, no se puede evaluar una expresión de la forma **A+B**. Este tipo de operaciones deben realizarse por medio de la función **evalm**. En particular la multiplicación de matrices requiere el uso de esta función, así como del operador **&***, el cual indica un producto no conmutativo.

- **Álgebra Lineal abstracta.**

  Es posible realizar cálculos de Álgebra Lineal abstracta, a través del uso de operadores inertes y la función **evalm**.

- **Manejo limitado de matrices grandes con elementos numéricos.**

  Las funciones del paquete ***linalg*** permiten manipular matrices con elementos tanto numéricos como simbólicos; sin embargo, su capacidad para operar matrices muy grandes con elementos numéricos es limitada.

14.1.2 El paquete LinearAlgebra

Este paquete contiene un conjunto de funciones de Álgebra Lineal, las cuales proporcionan las mismas funcionalidades del paquete ***linalg*** (y aún más). Entre sus características más notables se encuentran:

- **Estructuras de datos básicas**

  Las estructuras de datos usadas como base por este paquete son **"Vector"** y **"Matrix"**. Estas estructuras son creadas a partir de las funciones **Vector** y **Matrix**; o bien, por medio de la notación abreviada ***"<a, b, c>"***. Internamente, este tipo de datos están basados en una estructura de datos conocida como ***"rtable"***. Por esta razón, las listas, los arreglos, las matrices y vectores de **linalg** y los arreglos basados en tablas no pueden combinarse con datos creados a partir de **rtables**, sin ser convertidos previamente.

- **Comandos para tipos especiales de matrices y vectores.**

  Este paquete contiene funciones para la creación de tipos especiales de matrices y vectores; tales como identidades, cero y constantes.

- **Álgebra de matrices mejorada.**

  Las matrices y vectores generados por las funciones de este paquete pueden ser operados usando los operadores aritméticos habituales; es decir, las expresiones del tipo **A+B**, **A.B**, **A-B** son evaluadas directamente sin necesidad de usar una función especial para calcularlas. Nótese que el producto es expresado usando el operador punto: **"."**.

- **Soporte más eficiente de matrices numéricas grandes.**

  El paquete está implementado en base a algoritmos numéricos más eficientes, lo que proporciona mejor soporte para manejo de matrices grandes de datos numéricos

14.2 Elección entre linalg y LinearAlgebra

Los puntos más importantes a tener en cuenta al elegir entre estos dos paquetes son:

- El paquete **linalg** es útil para realizar cálculos de álgebra abstracta.
- Comparado con **linalg**, el paquete **LinearAlgebra** incluye varias funciones para creación de matrices especiales, permite realizar álgebra de matrices de manera más fácil y es más poderoso y eficiente al realizar cálculos, especialmente cuando se manipulan matrices numéricas grandes.

14.3 Conversión de datos entre linalg y LinearAlgebra

Como se mencionó anteriormente, estos paquetes usan estructuras de datos diferentes como base para la creación de matrices y vectores, las cuales no pueden ser combinadas para operarlas. Sin embargo, los datos creados con las funciones de uno de ellos pueden ser convertidos para ser usados por funciones del otro paquete. Esto puede hacerse de la siguiente forma:

- Los vectores creados con el paquete **linalg** deben ser convertidos mediante la función:

  convert(..., Vector),

  para ser usados por **LinearAlgebra**.
- Las matrices creadas con **linalg**, deben convertirse usando la instrucción:

  convert(..., Matrix).
- Los vectores creados con **LinearAlgebra** deben ser convertidos con:

  convert(..., vector)

  para poder ser usados por **linalg**.
- Las matrices creadas con **LinearAlgebra** deben convertise mediante:

  convert(..., matrix)

Adicionalmente, una matrix o un vector creados con ***linalg*** puede ser usado como argumento en las funciones **Matrix** y **Vector** de **LinearAlgebra**, respectivamente; con lo cual contamos con otro mecanismo para convertir datos de **linalg** a **LinearAlgebra**.

En las siguientes secciones haremos uso de las funciones contenidas en el paquete **LinearAlgebra**, por lo cual es conveniente cargar éste antes de continuar.

```
> with(LinearAlgebra):
```

14.4 Vectores

A continuación haremos una revisión de las principales instrucciones proporcionadas por Maple, útiles en la creación, manipulación y operación de vectores.

14.4.1 Creación de un vector

Un vector puede ser creado por medio de la función **Vector**. Esta instrucción puede ser usada de varias formas para la creación de vectores, una de esas formas es:

Vector([v1, v2, v3 ,..., vn])

Donde *"v1, v2, v3, ... , vn"* son las entradas del vector; la dimensión de éste es calculada automáticamente con el número de entradas proporcionadas, aunque también puede especificarse como primer argumento, en cualquiera de las siguientes formas:

Vector(1..n, [v1, v2, v3, ..., vn])

Vector(n, [v1, v2, v3, ..., vn])

Una característica interesante de esta función es que nos da la posibilidad de especificar si deseamos crear un vector columna o un vector renglón. La sintaxis es:

Vector[row](...)

Vector[column](...);

Si no se especifica el tipo de vector a crear, esto es equivalente a la forma: **Vector[column](...)**.

Por ejemplo, a continuación crearemos los vectores **v1** y **v2**, y calcularemos su norma por medio de la función **VectorNorm** de **LinearAlgebra**.

```
> v1 := Vector[row]([9, 3, 2.4]);
```

$$v1 := [9, 3, 2.4]$$

```
> v2 := Vector[column](4, [3.5, 0.4, 7.2, Pi/2]);
```

$$v2 := \begin{bmatrix} 3.5 \\ 0.4 \\ 7.2 \\ \frac{\pi}{2} \end{bmatrix}$$

```
> VectorNorm(v1, Euclidean);
```

$$9.78570385818005484$$

```
> VectorNorm(v2, Euclidean);
```

$$\sqrt{64.25 + \frac{\pi^2}{4}}$$

Otra forma en la que podemos crear este tipo de elementos es por medio de la notación abreviada:

<v1, v2, v3, ..., vn>. Para crear vectores columna.

<v1 | v2 | v3 | ... | vn>. Para crear vectores renglón.

Donde **v1, v2, v3, ..., vn** son las entradas del vector. Por ejemplo:

```
> v3 := <Pi/4, alpha, 3, sqrt(Pi)>;
```

$$v3 := \begin{bmatrix} \frac{\pi}{4} \\ \alpha \\ 3 \\ \sqrt{\pi} \end{bmatrix}$$

```
> v4 := <4.5 | 2 | 8.4 | 3>;
```

$$v4 := [4.5, 2, 8.4, 3]$$

Este tipo de estructuras son evaluadas automáticamente al colocar su nombre en la línea de comandos, por ejemplo, para desplegar los elementos de los vectores **v1** y **v4** simplemente los referenciamos:

```
> v1;
```

$$[9, 3, 2.4]$$

```
> v4;
```

$$[4.5, 2, 8.4, 3]$$

Adicionalmente, al crear un vector con la función **Vector**, podemos incluir la opción **'ro'**, la cual indica a Maple que estamos creando un vector de solo lectura, es decir, ninguna de las entradas de este vector podrán ser modificadas.

14.4.2 Creación de un vector a partir de arreglos y listas

Maple nos permite obtener vectores a partir de listas y arreglos previamente definidos, convirtiendo éstos por medio de la función **convert**. Esta función crea de manera predeterminada vectores columna, sin embargo, podemos especificar el tipo de vector que deseamos obtener de la siguiente manera:

convert(lista, Vector[column])

convert(lista, Vector[row])

La primera forma es equivalente a **convert(lista, Vector)**. Por ejemplo, crearemos los siguientes vectores y calcularemos su producto por medio de la función **DotProduct** y la norma de ambos por medio de la función **VectorNorm** (ambos pertenecen a **LinearAlgebra**).

```
> lista := [4, 8, 9, 3];
```

$$lista := [4, 8, 9, 3]$$

```
> ar := array(3..6, [3, 12, 4.6, 8]);
```

$$\begin{aligned} ar := \text{array}(3..6, [\\ (3) = 3 \\ (4) = 12 \\ (5) = 4.6 \\ (6) = 8 \\ ]) \end{aligned}$$

```
> v5 := convert(lista,Vector);
```

$$v5 := \begin{bmatrix} 4 \\ 8 \\ 9 \\ 3 \end{bmatrix}$$

```
> v6 := convert(ar, Vector[row]);
```

$$v6 := [3, 12, 4.6, 8]$$

```
> DotProduct(v5, v6);
```

$$173.400000000000006$$

```
> VectorNorm(v5);
```

$$9$$

```
> VectorNorm(v6);
```

$$12.$$

Nótese que los índices asignados a los arreglos no deben iniciar forzosamente en uno para poder convertirlos en vectores.

Otra forma en la que podemos crear un vector a partir de una lista o un arreglo es incluyendo éstos como argumento en la función **Vector**, como se muestra a continuación:

```
> Vector(4, ar);
```

$$[3, 12, 4.6, 8]$$

```
> Vector(4, lista);
```

$$\begin{bmatrix} 4 \\ 8 \\ 9 \\ 3 \end{bmatrix}$$

En este caso, de manera predeterminada los arreglos son convertidos a vectores renglón, mientras que las listas son convertidas a vectores columna.

14.4.3 Acceso a los elementos de un vector

Los elementos de un vector pueden ser invocados y modificados de la misma forma que en el caso de las listas, es decir haciendo referencia a sus elementos por medio de índices, los cuales siempre inician en uno. Para ejemplificar esto usaremos el vector **v2** creado anteriormente:

```
> v2;
```

$$\begin{bmatrix} 3.5 \\ 0.4 \\ 7.2 \\ \frac{\pi}{2} \end{bmatrix}$$

```
> v2[1]; v2[3];
```

$$3.5$$
$$7.2$$

```
> v2[2] := diff(x^2, x);
```

$$v2_2 := 2\,x$$

```
> v2;
```

$$\begin{bmatrix} 3.5 \\ 2\,x \\ 7.2 \\ \frac{\pi}{2} \end{bmatrix}$$

14.4.4 Vectores creados sin elementos

Podemos crear vectores sin especificar sus elementos y asignar éstos posteriormente, especificando únicamente la dimensión. En este caso el vector es creado con ceros en todas sus entradas. Por ejemplo:

```
> v7 := Vector(3);
```

$$v7 := \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

```
> v7[1] := 8; v7[2] := 4; v7[3] := 6;
```

$$v7_1 := 8$$
$$v7_2 := 4$$
$$v7_3 := 6$$

```
> v7;
```

$$\begin{bmatrix} 8 \\ 4 \\ 6 \end{bmatrix}$$

14.4.5 Vectores aleatorios

Este tipo de vectores pueden ser creados con la función **RandomVector(n)**, donde **n** es la dimensión del vector a crear. Esta función nos da como resultado un vector columna, aunque opcionalmente también puede especificarse el tipo de vector de la siguiente forma:

RandomVector[row](n)

RandomVector[column](n)

Por ejemplo:

```
> RandomVector(5);
```

$$\begin{bmatrix} 30 \\ 62 \\ -79 \\ -71 \\ 28 \end{bmatrix}$$

```
> RandomVector[row](8);
```

$$[16, -34, -62, -90, -21, -56, -8, -50]$$

14.4.6 Vectores generados por funciones

Otra manera de crear vectores es a través del uso de la función **Vector**, de la siguiente manera:

Vector(d, f)

Donde **d** es la dimensión del vector y **f** es una función de una variable que se aplicará a los índices del vector para generar los elementos correspondientes. Por ejemplo:

```
> f := x -> x^2 + 4*x;
```

$$f := x \rightarrow x^2 + 4x$$

```
> d := Vector(4, f);
```

$$d := \begin{bmatrix} 5 \\ 12 \\ 21 \\ 32 \end{bmatrix}$$

```
> e := Vector[row](5, f);
```

$$e := [5, 12, 21, 32, 45]$$

Esta última expresión es equivalente a la siguiente:

```
> e := Vector[row](5, [f(1), f(2), f(3), f(4), f(5)]);
```

$$e := [5, 12, 21, 32, 45]$$

14.4.7 Aplicación de funciones a los elementos de un vector

Una vez definido un vector, podemos usar **map** para aplicar una función dada a cada uno de sus elementos. Por ejemplo:

```
> v9 := Vector([4, Pi, exp(1.1), 4*sin(Pi/4), Pi^2]);
```

$$v9 := \begin{bmatrix} 4 \\ \pi \\ 3.004166024 \\ 2\sqrt{2} \\ \pi^2 \end{bmatrix}$$

```
> map(cos, v9);
```

$$\begin{bmatrix} \cos(4) \\ -1 \\ -0.9905718132 \\ \cos(2\sqrt{2}) \\ \cos(\pi^2) \end{bmatrix}$$

```
> v10 := Vector[row](5, n -> n^2 + x^n);
```

$$v10 := [1 + x, 4 + x^2, 9 + x^3, 16 + x^4, 25 + x^5]$$

```
> map(diff, v10, x);
```

$$[1, 2x, 3x^2, 4x^3, 5x^4]$$

El paquete **LinearAlgebra** contiene otras dos funciones similares a **map**, éstas son: **Map** y **Map2**, las cuales proporcionan algunas características más para realizar mapeos de funciones sobre vectores. Consúltense sus páginas de ayuda.

Las funciones que proporciona Maple para manipulación de vectores nos permiten realizar, entre otras, las operaciones que se describen en las siguientes secciones. Muchas de estas operaciones están soportadas a través de funciones del paquete **LinearAlgebra**; por lo que es conveniente asegurarse de que éste se ha cargado previamente.

14.4.8 Operaciones aritméticas con vectores

Este tipo de operaciones pueden realizarse directamente en la linea de comandos, por medio de los operadores aritméticos usuales. Veamos los siguientes ejemplos:

```
> v := <-3 | 1 | 7>;
```

$$v := [-3, 1, 7]$$

```
> w := <9 | -3 | -21>;
```

$$w := [9, -3, -21]$$

Calcularemos las operaciones: **v + w**, **4*w**, ***w*4*** y **7*v + w**.

```
> v + w;
```

$$[6, -2, -14]$$

```
> w - v;
```

$$[12, -4, -28]$$

```
> 4*w;
```

$$[36, -12, -84]$$

```
> w*4;
```

$$[36, -12, -84]$$

```
> 7*v + w;
```

$$[-12, 4, 28]$$

Como puede observarse, el operador '*' nos permite calcular el producto de un vector por un escalar. Otro operador aplicable en este caso es '.', el cual nos permite calcular el producto punto de los vectores:

```
> v.w;
```

$$-177$$

Estas operaciones también pueden calcularse por medio de las funciones:

Add(v1, v2)

ScalarMultiply(v1, c)

DotProduct(v1, v2)

Las cuales calculan la suma de dos vectores, el producto de un vector por un escalar y el producto punto de dos vectores, respectivamente. **v1** y **v2** representan vectores, mientras que **c** representa un escalar.

A continuación comprobaremos si los vectores **w** y **v** (definidos arriba) son paralelos, para verificar esto debemos encontrar un valor **t** tal que **v=t*w**. Esto podemos hacerlo de la siguiente forma.

Primero definamos el siguiente vector:

```
> m := ScalarMultiply(w, t);
```

$$m := [9t, -3t, -21t]$$

Necesitamos encontrar t tal que $v=m$, para ello debemos resolver el siguiente sistema de ecuaciones:

```
> sis := {v[1] = m[1], v[2] = m[2], v[3] = m[3]};
```

$$sis := \{-3 = 9t, 1 = -3t, 7 = -21t\}$$

```
> sol := solve(sis, t);
```

$$sol := \{t = \frac{-1}{3}\}$$

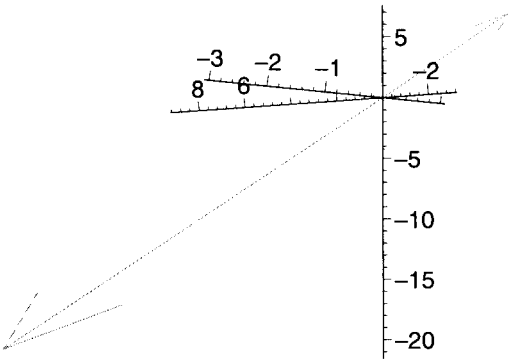
Verifiquemos multiplicando **w** por el valor obtenido y comprobemos si el resultado es igual a **v**:

```
> rhs(sol[1])*w;
```

$$[-3, 1, 7]$$

Por lo tanto los vectores son paralelos. Grafiquemos **v** y **w**.

```
> grafv := plots[arrow](v, shape=arrow):
> grafw := plots[arrow](w, shape=arrow):
> plots[display](grafv, grafw, axes=normal, orientation=[50, 85]);
```



En la gráfica puede observarse que los vectores son paralelos.

A continuación, calcularemos la ecuación de la recta que pasa por los puntos **f** y **g** y desplegaremos su gráfica.

```
> f := <2 | 5 | 8>;
```

$$f := [2, 5, 8]$$

```
> g := <3 | -4 | 2>;
```

$$g := [3, -4, 2]$$

La ecuación de la recta está dada por $\mathbf{x = f + t(g - f)}$. Ésta debe ser convertida a lista para poder colocarla como argumento de **plot3d**.

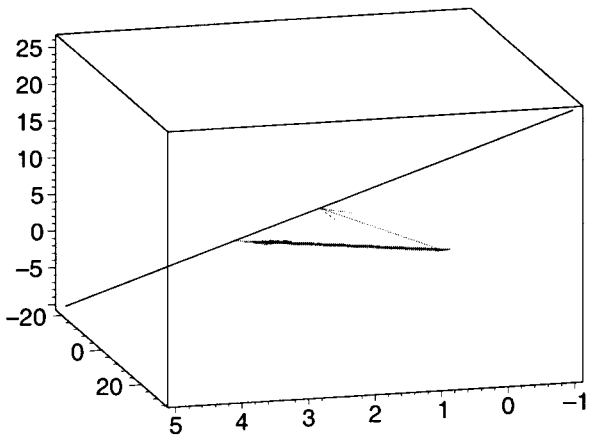
```
> recta := convert(evalm(f + t*(g - f)), list);
```

$$recta := [t + 2, -9t + 5, -6t + 8]$$

Generamos las gráficas de los vectores y de la recta, y las desplegamos.

```
> grf := plots[arrow](f, shape=arrow):
> grg := plots[arrow](g):
> grrec := plot3d(recta, t=-3..3,y=-3..3, axes=boxed):
```

```
> plots[display]({grf, grg, grrec}, orientation=[75, 70]);
```



A continuación, calculamos y graficamos el plano que generan los puntos **P**, **Q** y **R**.

```
> P := <3 | 6 | 7>;
```

$$P := [3, 6, 7]$$

```
> Q := <-4 | 7 | 9>;
```

$$Q := [-4, 7, 9]$$

```
> R := <5 | -9 | 15>;
```

$$R := [5, -9, 15]$$

Primero calcularemos el vector que parte del origen y tienen la misma dirección que el vector que va de **P** a **Q**, lo mismo para el vector que va de **P** a **R**.

```
> u := Q - P; # vector que va de P a Q
```

$$u := [-7, 1, 2]$$

```
> v := R - P; # vector que va de P a R
```

$$v := [2, -15, 8]$$

La ecuación del plano está dada por **x = P + t*u + s*v**.

A continuación generaremos las gráficas de **P**, **Q** y **R**, así como la del plano.

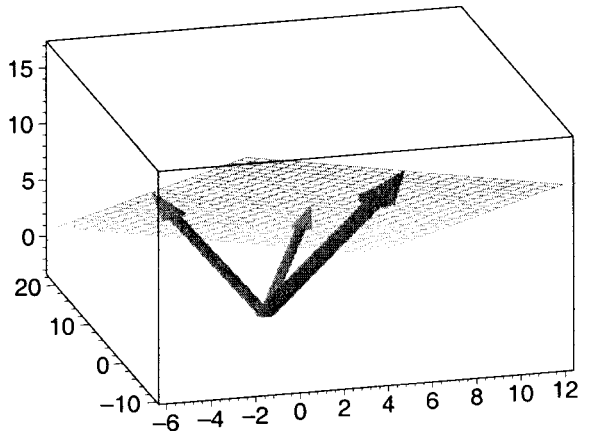
```
> grP := plots[arrow](P):
> grQ := plots[arrow](Q):
> grR := plots[arrow](R):
> plano := convert(evalm(P + t*u + s*v), list);
```

$$plano := [-7t + 2s + 3, t - 15s + 6, 2t + 8s + 7]$$

```
> grplano := plot3d(plano, t=-1..1, s=-1..1):
```

En el siguiente despliegue mostraremos todas estas gráficas, con lo cual podemos darnos cuenta de que, efectivamente, el plano es generado por los vectores **P**, **Q** y **R**.

```
> plots[display]({grP, grQ, grR, grplano}, style=wireframe,
> axes=boxed, orientation=[-105, 60]);
```



Veamos otro ejemplo. Consideremos la siguiente definición:

Sea **V** el conjunto de parejas ordenadas de números reales. Para (x_1, y_1) y $(x_2, y_2) \in \mathbf{V}$, y **c** un número real, definimos:

$$(x_1, y_1) + (x_2, y_2) = (x_1 + y_1, x_2 + y_2)$$

$$c\,(x_1, y_1) = \begin{cases} (0,0) & \text{si } c = 0 \\ (c\,x_1, \frac{y_1}{2}) & \text{si } c \neq 0 \end{cases}$$

¿Bajo estas operaciones, es **V** un espacio vectorial?

Para que $\boldsymbol{V}$ sea un espacio vectorial, dos de las condiciones que deben cumplirse son, para P, Q elementos de **V** y c, d números reales:

$$\text{i)}\quad c\,(P+Q) = c\,P + c\,Q$$

$$\text{ii)}\quad (c+d)\,P = c\,P + d\,P$$

Definamos los siguientes vectores y constantes:

```
> P := <4 | 12>;
```

$$P := [4, 12]$$

```
> Q := <3 | 8>;
```

$$Q := [3, 8]$$

```
> c := 4;
```

$$c := 4$$

```
> d := 3;
```

$$d := 3$$

A continuación creamos la función producto, de acuerdo a la definición del producto por escalar de **V**:

```
> producto := proc(c::numeric, V::Vector)
> if (c = 0)
> then
> <0 | 0>;
> else
> <c*V[1] | V[2]/c>;
> fi;
> end:
```

Ahora calculamos $c(P+Q)$, $cP+cQ$, y los comparamos:

```
> producto(c, P + Q);
```

$$[28, 5]$$

```
> producto(c, P) + producto(c, Q);
```

$$[28, 5]$$

Esta relación si se cumple. Verifiquemos ahora si: $(c+d)P = cP + dP$.

```
> producto(c + d, P);
```

$$\left[28, \frac{12}{7}\right]$$

```
> producto(c, P) + producto(d,P);
```

$$[28, 7]$$

Podemos ver en este caso que la relación no se cumple; por lo tanto, **V** no es un espacio vectorial.

Consideremos ahora el mismo conjunto **V** de arriba, pero ahora definimos las siguientes operaciones, para (x_1, y_1) y $(x_2, y_2) \in \mathbf{V}$ y c un número real:

$$(x_1, y_1) + (x_2, y_2) = (x_1 + 2\,y_1,\ x_2 + 3\,y_2)$$

$$c\,(x_1, y_1) = (c\,x_1, c\,y_1)$$

¿Bajo estas operaciones, es **V** un espacio vectorial?

Esta comprobación podemos hacerla aprovechando la capacidad de Maple para manipular datos simbólicos. Utilizaremos la función **evalb**, la cual nos permite verificar una expresión booleana.

Dos de las condiciones necesarias para que $\boldsymbol{V}$ sea un espacio vectorial, dados **P**, **Q**, $\boldsymbol{R}$ en **V**, son:

$$\text{i)}\quad P + Q = Q + P$$

$$\text{ii)}\quad (P + Q) + R = P + (Q + R)$$

Verifiquemos si éstas se cumplen, para **P**, **Q** y **R** elementos generales de **V**.

```
> P := <x1 | y1>;
```

$$P := [x1,\ y1]$$

```
> Q := <x2 | y2>;
```

$$Q := [x2,\ y2]$$

```
> R := <x3 | y3>;
```

$$R := [x3, y3]$$

```
> suma := proc(V::Vector, W::Vector)
> <V[1] + 2*W[1] | V[2] + 3*W[2]>;
> end;
```

$$suma := \mathbf{proc}(V::Vector, \mathrm{W}::Vector)\,\langle V_1 + 2 * \mathrm{W}_1 \mid V_2 + 3 * \mathrm{W}_2\rangle\ \mathbf{end\ proc}$$

```
> evalb(suma(P, Q) = suma(Q, P));
```

$$false$$

```
> evalb(suma(suma(P, Q), R) = suma(P, suma(Q, R)));
```

$$false$$

De lo anterior podemos ver que, en general, las dos relaciones no se cumplen, por lo tanto $\boldsymbol{V}$ no es espacio vectorial bajo estas operaciones. Comprobemos esto mismo con valores particulares de P, Q y R.

```
> P := <2 | 4>;
```

$$P := [2, 4]$$

```
> Q := <3 | 5>;
```

$$Q := [3, 5]$$

```
> R := <4 | 7>;
```

$$R := [4, 7]$$

```
> suma(P, Q) = suma(Q, P);
```

$$[8, 19] = [7, 17]$$

```
> suma(suma(P, Q), R) = suma(P, suma(Q, R));
```

$$[16, 40] = [24, 82]$$

14.4.9 Funciones de LinearAlgebra para manipulación de vectores

El paquete **LinearAlgebra** proporciona varias funciones útiles en el manejo de vectores, entre ellas se encuentran las que se describen en las siguientes secciones.

Ángulo entre dos vectores

La función **VectorAngle** nos permite calcular el ángulo entre dos vectores. Por ejemplo:

```
> v1 := <6 | 3>;
```

$$v1 := [6, 3]$$

```
> v2 := <-3 | 6>;
```

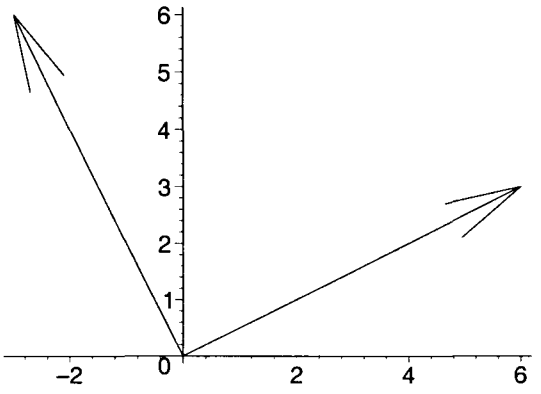
$$v2 := [-3, 6]$$

```
> VectorAngle(v1, v2);
```

$$\frac{\pi}{2}$$

El resultado anterior se debe a que los vectores son perpendiculares. Verifiquemos con una gráfica:

```
> grafv1 := plots[arrow](v1, shape=arrow):
> grafv2 := plots[arrow](v2, shape=arrow):
> plots[display]({grafv1, grafv2}, scaling=constrained);
```



Igualdad de dos vectores

Dados dos vectores, la función **Equal** nos permite determinar si sus respectivos componentes son iguales (también puede ser usada con matrices), su sintaxis es:

Equal(V, W)

Donde **V** y **W** son vectores. Esta función da un resultado de verdadero si ambos argumentos son del mismo tipo (vectores o matrices), tienen la misma dimensión y sus componentes son iguales. En particular, si sus argumentos son vectores, éstos deben tener la misma orientación. Por ejemplo:

```
> a := <1 | 3 | 7>;
```

$$a := [1, 3, 7]$$

```
> b := <1 | 3 | 7>;
```

$$b := [1, 3, 7]$$

```
> c := <4 | 5 | 2>;
```

$$c := [4, 5, 2]$$

```
> Equal(a, b);
```

true

```
> Equal(b,c);
```

false

Esta función proporciona algunas otras opciones útiles en la comparación de vectores. Consúltese su página de ayuda.

Producto punto

Otra de las funciones presentes en este paquete es **DotProduct**, con la cual podemos calcular el producto punto de dos vectores. Por ejemplo, calcularemos el producto punto de los siguientes vectores para verificar si son perpendiculares.

```
> v1 := <6 | 3>;
```

$$v1 := [6, 3]$$

```
> v2 := <-3 | 6>;
```

$$v2 := [-3, 6]$$

```
> DotProduct(v1, v2);
```

$$0$$

El resultado que obtuvimos es cero, por lo tanto los vectores son perpendiculares.

Recuérdese que este producto también puede calcularse por medio del operador '.'; es decir, la instrucción anterior es equivalente a la operación **v1.v2**.

Norma de un vector

También podemos calcular la norma de un vector, utilizando la función **Norm**, o por medio de **VectorNorm**, ambas contenidas en **LinearAlgebra**. La sintaxis es:

Norm(V, Euclidean);

VectorNorm(V, Euclidean);

Por ejemplo:

```
> A := <-7 | 5>;
```

$$A := [-7, 5]$$

```
> Norm(A, Euclidean);
```

$$\sqrt{74}$$

Verifiquemos este resultado calculando manualmente la norma de **A**.

```
> sqrt(A[1]^2 + A[2]^2);
```

$$\sqrt{74}$$

Producto cruz

Otra operación que podemos aplicar sobre vectores está dada por la función **CrossProduct**, la cual nos permite calcular el producto cruz de dos vectores de dimensión tres. Por ejemplo, encontraremos un vector unitario perpendicular a los vectores **W, Z**.

```
> W := <2 | 5 | -4>;
```

$$W := [2, 5, -4]$$

```
> Z := <-4 | 6 | 8>;
```

$$Z := [-4, 6, 8]$$

Calculamos el producto cruz:

```
> P := CrossProduct(W, Z);
```

$$P := [64, 0, 32]$$

A continuación calculamos la norma de **P** y el producto **P * 1/norma(P)**, el resultado es un vector **R**, unitario y perpendicular a **W** y **Z**.

```
> normaP := Norm(P, Euclidean);
```

$$normaP := 32\sqrt{5}$$

```
> R := 1/normaP * P;
```

$$R := \left[\frac{2\sqrt{5}}{5}, 0, \frac{\sqrt{5}}{5}\right]$$

Para verificar que el resultado es un vector unitario y perpendicular calcularemos los productos **W.R** y **Z.R**, y la norma de **R**.

```
> W.R;
```

$$0$$

```
> Z.R;
```

$$0$$

```
> Norm(R, Euclidean);
```

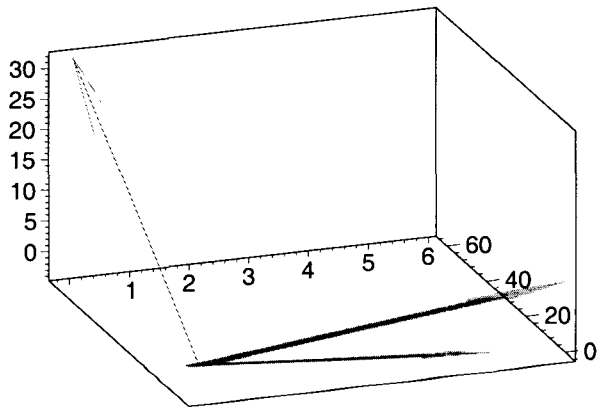
$$1$$

A continuación generaremos las gráficas de los vectores **W**, **Z** y **P**.

```
> grafW := plots[arrow](W):
> grafZ := plots[arrow](Z):
> grafP := plots[arrow](P, shape=arrow):
```

Finalmente mostraremos estas gráficas en un mismo despliegue para comprobar que el vector **P** (y por consiguiente también **R**) es perpendicular a **W** y **Z**.

```
> plots[display]({grafW, grafZ, grafP}, axes=boxed, orientation=[20, 120]);
```



Dimensión de un vector

La función **Dimension** nos permite calcular la dimensión de un vector (también puede usarse para calcular la dimensión de una matriz).

```
> g := x -> 2*x + 4*x^2;
```

$$g := x \to 2x + 4x^2$$

```
> w := Vector(5, g);
```

$$w := \begin{bmatrix} 6 \\ 20 \\ 42 \\ 72 \\ 110 \end{bmatrix}$$

```
> Dimension(w);
```

$$5$$

Este paquete proporciona algunas otras funciones para manipulación de estructuras creadas con **Vector** (o con la notación abreviada "<...>"), así como para la creación de vectores especiales. Entre ellos se encuentran:

- **UnitVector(i, d)**. Construye un **Vector** de dimensión **d**, cuya entrada i-ésima es igual a uno y las demás son cero.
- **VectorAdd(V, W)**. Calcula la suma de los vectores **V** y **W**. Esta función puede ser usada de la misma forma que **Add**; aunque también puede ser usada en la forma: **VectorAdd(V, W, c, d)**, para calcular la combinación lineal **c*V + d*W**.
- **VectorScalarMultiply(V, c)**. Calcula el producto del vector **V** por el escalar **c**.
- **ZeroVector(d)**. Construye un **Vector** de dimensión **d**, con todos sus elementos iguales a cero.

Antes de continuar ejecútese la siguiente instrucción para eliminar cualquier resultado anterior.

```
> restart;
```

Asegúrese de cargar nuevamente el paquete **LinearAlgebra**, con la instrucción **with(LinearAlgebra)**; ya que las funciones de éste serán usadas en las siguientes secciones.

14.5 Matrices

Maple proporciona un conjunto de funciones útiles en el manejo de matrices. En las siguientes secciones haremos una revisión de las capacidades de este sistema, para el manejo y operación de este tipo de datos.

14.5.1 Creación de matrices

Este tipo de dato puede ser creado por medio de la función **Matrix**, usando una de las siguientes expresiones.

Matrix([[a11, a12, ..., a1n], [a21, a22, ..., a2n], ... [am1, am2, ..., amn]])

Matrix(m, n, [[a11, a12, ..., a1n], [a21, a22, ..., a2n], ... [am1, am2, ..., amn]])

Matrix(1..m, 1..n, [[a11, a12, ..., a1n], [a21, a22, ..., a2n], ... [am1, am2, ..., amn]])

En estas expresiones, **aij** representa el elemento del renglón i-ésimo y la columna j-ésima de la matriz. En el primer caso, al no incluirse las dimensiones, éstas son calculadas por el número de elementos proporcionados. Las siguientes dos formas muestran maneras diferentes de especificar las dimensiones.

Al igual que en el caso de los vectores, también podemos usar la siguiente notación abreviada para la creación de este tipo de estructuras:

<<a11, a12, ..., a1n> | <a21, a22, ..., a2n> | ... | <am1, am2, ..., amn>>

para la creación de una matriz por columnas, o bien:

<<a11 | a12 | ... | a1n>, <a21 | a22 | ... | a2n>, ... , <am1 | am2 | ... | amn>>

para construirla por renglones. Por ejemplo:

```
> m := Matrix(1..3, 1..3, [[1, 2, 3], [5, 2, 1], [3, 6, 2]]);
```

$$m := \begin{bmatrix} 1 & 2 & 3 \\ 5 & 2 & 1 \\ 3 & 6 & 2 \end{bmatrix}$$

```
> n := Matrix([[2, 4, 6], [3, 8, 2]]);
```

$$n := \begin{bmatrix} 2 & 4 & 6 \\ 3 & 8 & 2 \end{bmatrix}$$

```
> q := <<2 | 4 | 5>, <6 | 7 | 2>>;
```

$$q := \begin{bmatrix} 2 & 4 & 5 \\ 6 & 7 & 2 \end{bmatrix}$$

```
> r := <<4, 6, 2, 1> | <3, 5, 8, 3> | <9, 2, 4, Pi>>;
```

$$r := \begin{bmatrix} 4 & 3 & 9 \\ 6 & 5 & 2 \\ 2 & 8 & 4 \\ 1 & 3 & \pi \end{bmatrix}$$

Los elementos pueden o no ser especificados durante la creación de la matriz; en caso de que no sean proporcionados, ésta es creada con ceros. Por ejemplo:

```
> s := Matrix(1..3,1..2);
```

$$s := \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

```
> t := Matrix(2,2);
```

$$t := \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Los elementos pueden ser asignados posteriormente. Éstos pueden ser accesados por medio de sus índices, de la misma forma que en las listas. Por ejemplo:

```
> s[1, 1] := 4:  s[1, 2] := 8:  s[2, 1] := 3:  s[2, 2] := 9:
> s[3, 1] := 4:  s[3, 2] := 6:
> s;
```

$$\begin{bmatrix} 4 & 8 \\ 3 & 9 \\ 4 & 6 \end{bmatrix}$$

Nótese (en la última instrucción) que para desplegar los elementos de una matriz es suficiente con invocar ésta a través de su nombre, Maple automáticamente la evalúa y la muestra.

14.5.2 Matrices generadas por funciones

La función **Matrix** nos permite dar como argumento una función de dos variables, la cual se usará para generar los elementos de la matriz, aplicándola sobre los índices de los mismos elementos.

Por ejemplo:

```
> f := (i, j) -> x^i + sin(y^j);
```

$$f := (i, j) \to x^i + \sin(y^j)$$

```
> m := Matrix(3, 3, f);
```

$$m := \begin{bmatrix} x + \sin(y) & x + \sin(y^2) & x + \sin(y^3) \\ x^2 + \sin(y) & x^2 + \sin(y^2) & x^2 + \sin(y^3) \\ x^3 + \sin(y) & x^3 + \sin(y^2) & x^3 + \sin(y^3) \end{bmatrix}$$

Existen otras opciones proporcionadas por la función **Matrix** para la creación de este tipo de estructuras. Por ejemplo, si incluimos la opción **readonly=true**, la matriz será creada como de solo lectura, en cuyo caso no podemos modificar ninguna de sus entradas.

Algunas otras opciones serán tratadas en las siguientes secciones. Consúltese la página de ayuda de esta función para obtener información detallada de las distintas formas que proporciona para crear matrices.

14.5.3 Mapeo de funciones sobre matrices

La función **Map** de **LinearAlgebra** puede ser usado para aplicar una función a cada uno de los elementos de una matriz.

Veamos un ejemplo:

```
> r;
```

$$\begin{bmatrix} 4 & 3 & 9 \\ 6 & 5 & 2 \\ 2 & 8 & 4 \\ 1 & 3 & \pi \end{bmatrix}$$

```
> t := Map(x -> x^2 + 4*x - 5*a, r);
```

$$t := \begin{bmatrix} 32 - 5a & 21 - 5a & 117 - 5a \\ 60 - 5a & 45 - 5a & 12 - 5a \\ 12 - 5a & 96 - 5a & 32 - 5a \\ 5 - 5a & 21 - 5a & \pi^2 + 4\pi - 5a \end{bmatrix}$$

```
> Map(int, t, a);
```

$$\begin{bmatrix} 32a - \frac{5}{2}a^2 & 21a - \frac{5}{2}a^2 & 117a - \frac{5}{2}a^2 \\ 60a - \frac{5}{2}a^2 & 45a - \frac{5}{2}a^2 & 12a - \frac{5}{2}a^2 \\ 12a - \frac{5}{2}a^2 & 96a - \frac{5}{2}a^2 & 32a - \frac{5}{2}a^2 \\ 5a - \frac{5}{2}a^2 & 21a - \frac{5}{2}a^2 & \pi^2 a + 4\pi a - \frac{5}{2}a^2 \end{bmatrix}$$

14.5.4 Creación de matrices especiales

La función **Matrix** soporta varias opciones que podemos usar para crear matrices especiales; también el paquete **LinearAlgebra** proporciona varias otras útiles para este fin. Veamos algunos ejemplos:

- Matrices identidad. Pueden ser creadas con:

  Matrix(n, shape=identity)

  o bien:

  Matrix(1..n, shape=identity)

  Donde **n** determina la dimensión de la matriz. Las matrices identidad también pueden ser creadas por medio de la función **IdentityMatrix(n)** del paquete **LinearAlgebra**, como veremos a continuación:

```
> Matrix(4, shape=identity);
```

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

```
> IdentityMatrix(3);
```

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Matrices de constantes Pueden ser creadas de la siguiente forma:

  Matrix(n, m, c)

  Donde **n**, **m** indican las dimensiones de la matriz y **c** es una constante. También pueden crearse por medio de la función:

  ConstantMatrix(c, n, m)

  Por ejemplo:

```
> Matrix(2, 3, 5);
```

$$\begin{bmatrix} 5 & 5 & 5 \\ 5 & 5 & 5 \end{bmatrix}$$

```
> ConstantMatrix(Pi/2, 2, 4);
```

$$\begin{bmatrix} \frac{\pi}{2} & \frac{\pi}{2} & \frac{\pi}{2} & \frac{\pi}{2} \\ \frac{\pi}{2} & \frac{\pi}{2} & \frac{\pi}{2} & \frac{\pi}{2} \end{bmatrix}$$

- Matrices diagonales. Son creadas mediante:

  Matrix(n, <a1, a2,..., an>, shape=diagonal)

  Matrix(1..n, <a1, a2,..., an>, shape=diagonal)

  Donde **n** determina la dimensión de la matriz y la lista incluye los elementos de la diagonal; si estos no son proporcionados, la matriz es creada con ceros. También se puede usar la función **DiagonalMatrix** para la creación de estas matrices. Por ejemplo:

```
> Matrix(3, <2, 8, 6>, shape=diagonal);
```

$$\begin{bmatrix} 2 & 0 & 0 \\ 0 & 8 & 0 \\ 0 & 0 & 6 \end{bmatrix}$$

```
> DiagonalMatrix([9, 4, 5]);
```

$$\begin{bmatrix} 9 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 5 \end{bmatrix}$$

Al hacer uso de esta función no es necesario indicar la dimensión, es suficiente con proporcionar los elementos de la diagonal.

En este tipo de matrices podemos modificar posteriormente tanto los elementos de la diagonal, como los elementos fuera de ésta. Por ejemplo:

```
> A := DiagonalMatrix([3, 8, 2]);
```

$$A := \begin{bmatrix} 3 & 0 & 0 \\ 0 & 8 & 0 \\ 0 & 0 & 2 \end{bmatrix}$$

```
> A[1, 1] := 5;
```

$$A_{1,1} := 5$$

```
> A[2, 1] := 4;
```

$$A_{2,1} := 4$$

```
> A;
```

$$\begin{bmatrix} 5 & 0 & 0 \\ 4 & 8 & 0 \\ 0 & 0 & 2 \end{bmatrix}$$

- Matrices simétricas. Pueden crearse con:

  Matrix(n, m, shape=symmetric)

De la misma forma que en el caso anterior se pueden inicializar sus elementos durante la creación. Por ejemplo:

```
> B := Matrix(3, 3, shape=symmetric);
```

$$B := \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

```
> B[1, 2] := 4; B[1, 3] := 6;
```

$$B_{1,2} := 4$$

$$B_{1,3} := 6$$

```
> B;
```

$$\begin{bmatrix} 0 & 4 & 6 \\ 4 & 0 & 0 \\ 6 & 0 & 0 \end{bmatrix}$$

Nótese que al modificar los elementos de esta matriz, Maple automáticamente mantiene la simetría.

- Matrices con elementos cero. Este tipo de matrices pueden ser creadas con:

  ZeroMatrix(m, n)

  Donde **m**, **n** determinan las dimensiones de la matriz. Éstas también pueden ser creadas como matrices constantes, como vimos anteriormente. Por ejemplo:

```
> Matrix(3, 4, 0);
```

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

```
> ZeroMatrix(2, 3);
```

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

- Matrices aleatorias. Éstas pueden ser creadas por medio de la función:

  RandomMatrix(m, n)

  Por ejemplo:

```
> RandomMatrix(2, 3);
```

$$\begin{bmatrix} -50 & 62 & -71 \\ 30 & -79 & 28 \end{bmatrix}$$

- Matrices triangulares. Pueden crearse con las siguientes instrucciones:

  Matrix(n, m, [...], shape=triangular[upper])

  Matrix(n, m, [...], shape=triangular[lower])

  Donde **n**, **m** determinan las dimensiones de la matriz (no tiene que ser cuadrada), mientras que la lista determina los elementos con los cuales se inicializará ésta. Si la matriz es cuadrada, es suficiente con colocar una de las dimensiones. Por ejemplo:

```
> Matrix(3, [[2], [4, 5], [2, 8, 9]], shape=triangular[lower]);
```

$$\begin{bmatrix} 2 & 0 & 0 \\ 4 & 5 & 0 \\ 2 & 8 & 9 \end{bmatrix}$$

La función **Matrix** proporciona otras opciones para crear matrices especiales. Consúltese su página de ayuda. Adicionalmente, el paquete ***linalg*** proporciona un conjunto de funciones para creación de matrices especiales, las cuales pueden ser convertidas por medio de **convert**. Entre éstas se encuentran:

- Matriz bloque. La función **blockmatrix** de ***linalg*** nos permite crear una matriz bloque a partir de matrices previamente definidas. Su sintaxis es:

  blockmatrix(m, n, L)

  Donde **m** y **n** indican la cantidad de bloques a usar y **L** es una lista de **m*n** elementos, cada uno de los cuales es una matriz. Por ejemplo:

```
> m1 := matrix([[1, 2, 4], [8, 7, 3], [5, 2, 9]]);
```

$$m1 := \begin{bmatrix} 1 & 2 & 4 \\ 8 & 7 & 3 \\ 5 & 2 & 9 \end{bmatrix}$$

```
> m2 := matrix([[2, 4, 8], [4, 2, 6], [7, 3, 5]]);
```

$$m2 := \begin{bmatrix} 2 & 4 & 8 \\ 4 & 2 & 6 \\ 7 & 3 & 5 \end{bmatrix}$$

```
> m3 := linalg[blockmatrix](2,2,[m1,m2,m1,m2]);
```

$$m3 := \begin{bmatrix} 1 & 2 & 4 & 2 & 4 & 8 \\ 8 & 7 & 3 & 4 & 2 & 6 \\ 5 & 2 & 9 & 7 & 3 & 5 \\ 1 & 2 & 4 & 2 & 4 & 8 \\ 8 & 7 & 3 & 4 & 2 & 6 \\ 5 & 2 & 9 & 7 & 3 & 5 \end{bmatrix}$$

- Matriz bloque diagonal Este tipo de matrices pueden ser creadas por medio de la función **diag**, su sintaxis es:

  diag(B1, B2, ... Bn)

  Donde **"B1, B2, ... , Bn"** son matrices cuadradas o valores constantes. Por ejemplo:

```
> B1 := matrix([[x^2, sin(x), 4], [5, 8, 2], [3, 6, 7]]);
```

$$B1 := \begin{bmatrix} x^2 & \sin(x) & 4 \\ 5 & 8 & 2 \\ 3 & 6 & 7 \end{bmatrix}$$

```
> B2 := matrix([[1, 8, 3], [-3, 2, 6], [4, 5, 2]]);
```

$$B2 := \begin{bmatrix} 1 & 8 & 3 \\ -3 & 2 & 6 \\ 4 & 5 & 2 \end{bmatrix}$$

```
> linalg[diag](B1, Pi/2, Pi^2, B2);
```

$$\begin{bmatrix} x^2 & \sin(x) & 4 & 0 & 0 & 0 & 0 & 0 \\ 5 & 8 & 2 & 0 & 0 & 0 & 0 & 0 \\ 3 & 6 & 7 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{\pi}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \pi^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 8 & 3 \\ 0 & 0 & 0 & 0 & 0 & -3 & 2 & 6 \\ 0 & 0 & 0 & 0 & 0 & 4 & 5 & 2 \end{bmatrix}$$

El paquete **LinearAlgebra** contiene algunas funciones más para creación de matrices especiales, entre las cuales se encuentran: **BezoutMatrix**, **HilbertMatrix**, **SylvesterMatrix**, **VandermondeMatrix** y **ToeplitzMatrix**. Además, también contiene la función **BandMatrix**, la cual permite crear una matriz banda, de la siguiente forma:

```
> lista := [[w, w], [x, x, x], [y, y, y], [z, z]];
```

$$lista := [[w, w], [x, x, x], [y, y, y], [z, z]]$$

```
> LinearAlgebra[BandMatrix](lista);
```

$$\begin{bmatrix} y & z & 0 \\ x & y & z \\ w & x & y \\ 0 & w & x \end{bmatrix}$$

Otra función proporcionada por **LinearAlgebra** es **HankelMatrix**. Esta función permite crear una matriz **M** a partir de los elementos de una lista **L**, de tal forma que el elemento **M[i, j] = L[i + j - 1]**. Véase su página de ayuda para más información.

Consúltese la página de ayuda de **LinearAlgebra** y de **linalg** para una lista completa de estas funciones.

14.5.5 Operaciones aritméticas con matrices

En el caso de las matrices creadas a partir de **Matrix**, éstas pueden ser operadas directamente con los operadores aritméticos '+' y '-', para suma y resta; '*' para producto por un escalar, y '.' para producto de matrices; además de '^' y '**' para multiplicar una matriz por si misma. Estos operadores también pueden ser usados en operaciones que involucren vectores, siempre que estos hayan sido creados con la función **Vector** o a partir de las funciones de **LinearAlgebra**. Por ejemplo:

```
> A := Matrix([[2, 5, 8], [8, 2, 6], [4, 2, 6]]);
```

$$A := \begin{bmatrix} 2 & 5 & 8 \\ 8 & 2 & 6 \\ 4 & 2 & 6 \end{bmatrix}$$

```
> B := Matrix([[x^2, y, 3], [4, 2, 7], [5, 3, 8]]);
```

$$B := \begin{bmatrix} x^2 & y & 3 \\ 4 & 2 & 7 \\ 5 & 3 & 8 \end{bmatrix}$$

```
> A + B;
```

$$\begin{bmatrix} 2+x^2 & 5+y & 11 \\ 12 & 4 & 13 \\ 9 & 5 & 14 \end{bmatrix}$$

```
> A.B;
```

$$\begin{bmatrix} 2x^2+60 & 2y+34 & 105 \\ 8x^2+38 & 8y+22 & 86 \\ 4x^2+38 & 4y+22 & 74 \end{bmatrix}$$

```
> A + B - 4*A.B;
```

$$\begin{bmatrix} -238-7x^2 & -131-7y & -409 \\ -140-32x^2 & -84-32y & -331 \\ -143-16x^2 & -83-16y & -282 \end{bmatrix}$$

Las siguientes expresiones pueden ser usadas para multiplicar una matriz por si misma:

```
> A^3;
```

$$\begin{bmatrix} 816 & 640 & 1388 \\ 1008 & 616 & 1456 \\ 704 & 472 & 1080 \end{bmatrix}$$

```
> B**2;
```

$$\begin{bmatrix} x^4+4y+15 & x^2y+2y+9 & 3x^2+7y+24 \\ 4x^2+43 & 4y+25 & 82 \\ 5x^2+52 & 5y+30 & 100 \end{bmatrix}$$

Estos operadores también nos permiten obtener la inversa de una matriz, por ejemplo:

```
> A := RandomMatrix(7, 5);
```

$$A := \begin{bmatrix} -98 & 59 & 97 & -65 & 16 \\ -41 & -69 & -82 & 5 & -34 \\ 24 & 23 & -66 & 66 & -62 \\ 65 & 25 & 55 & -36 & -90 \\ 1 & 5 & 68 & -41 & -21 \\ -42 & -75 & 26 & 20 & -56 \\ -82 & 38 & 13 & -7 & -8 \end{bmatrix}$$

```
> RowDimension(A);
```

$$7$$

```
> ColumnDimension(A);
```

$$5$$

```
> Dimension(A);
```

$$7, 5$$

Traza de una matriz

Ésta puede ser calculada mediante la función: **Trace(M)**, donde **M** debe ser una matriz cuadrada. Por ejemplo:

```
> m := Matrix([[1, 2, 3], [4, Pi, 5], [Pi^2, 8, exp(1.1)]]);
```

$$m := \begin{bmatrix} 1 & 2 & 3 \\ 4 & \pi & 5 \\ \pi^2 & 8 & 3.004166024 \end{bmatrix}$$

```
> Trace(m);
```

$$4.004166024 + \pi$$

Podemos aprovechar la capacidad que tiene Maple de manipular expresiones simbólicas para verificar el siguiente resultado:

Dadas dos matrices **A**, **B** de n x n, y **a**, **b** escalares, se cumple que:

traza(a*A + b*B) = a*traza(A) + b*traza(B)

Verifiquemos esto para matrices simbólicas de 4 x 4.

```
> A := Matrix(4, 4, [[A11, A12, A13, A14], [A21, A22, A23, A24],
> [A31, A32, A33, A34], [A41, A42, A43, A44]]);
```

$$A := \begin{bmatrix} A11 & A12 & A13 & A14 \\ A21 & A22 & A23 & A24 \\ A31 & A32 & A33 & A34 \\ A41 & A42 & A43 & A44 \end{bmatrix}$$

```
> B := Matrix(4, 4, [[B11, B12, B13, B14], [B21, B22, B23, B24],
> [B31, B32, B33, B34], [B41, B42, B43, B44]]);
```

$$B := \begin{bmatrix} B11 & B12 & B13 & B14 \\ B21 & B22 & B23 & B24 \\ B31 & B32 & B33 & B34 \\ B41 & B42 & B43 & B44 \end{bmatrix}$$

```
> r1 := Trace(a*A + b*B);
```

$$r1 := b\,B11 + a\,A11 + b\,B22 + a\,A22 + b\,B33 + a\,A33 + b\,B44 + a\,A44$$

```
> r2 := expand(a*Trace(A) + b*Trace(B));
```

$$r2 := b\,B11 + a\,A11 + b\,B22 + a\,A22 + b\,B33 + a\,A33 + b\,B44 + a\,A44$$

Una vez realizados los cálculos, podemos comprobar que la igualdad se cumple usando la función **evalb**.

```
> evalb(r1 = r2);
```

$$true$$

Transpuesta de una matriz

La transpuesta de una matriz puede calcularse con la función **Transpose**. Por ejemplo:

```
> n := Matrix([[4, Pi, 5, x^2], [Pi^2, 8, exp(1.1), x^3]]);
```

$$n := \begin{bmatrix} 4 & \pi & 5 & x^2 \\ \pi^2 & 8 & 3.004166024 & x^3 \end{bmatrix}$$

```
> Transpose(n);
```

$$\begin{bmatrix} 4 & \pi^2 \\ \pi & 8 \\ 5 & 3.004166024 \\ x^2 & x^3 \end{bmatrix}$$

Consideremos las siguientes proposiciones:

transpuesta(a*A + b*B) = a*transpuesta(A) + b*transpuesta(B)

transpuesta(transpuesta(A)) = A

Donde **A**, **B** son matrices de m x n, y **a**, **b** son escalares. Verifiquemos esto para matrices simbólicas de dimensión 3 x 4.

```
> A := Matrix(3, 4, [[A11, A12, A13, A14], [A21, A22, A23, A24],
> [A31, A32, A33, A34]]);
```

$$A := \begin{bmatrix} A11 & A12 & A13 & A14 \\ A21 & A22 & A23 & A24 \\ A31 & A32 & A33 & A34 \end{bmatrix}$$

```
> B := Matrix(3, 4, [[B11, B12, B13, B14], [B21, B22, B23, B24],
> [B31, B32, B33, B34]]);
```

$$B := \begin{bmatrix} B11 & B12 & B13 & B14 \\ B21 & B22 & B23 & B24 \\ B31 & B32 & B33 & B34 \end{bmatrix}$$

Utilizaremos la función **equal** de **linalg** para determinar la igualdad de las matrices resultantes.

```
> r1 := Transpose(c*A + d*B);
```

$$r1 := \begin{bmatrix} d\,B11 + c\,A11 & d\,B21 + c\,A21 & d\,B31 + c\,A31 \\ d\,B12 + c\,A12 & d\,B22 + c\,A22 & d\,B32 + c\,A32 \\ d\,B13 + c\,A13 & d\,B23 + c\,A23 & d\,B33 + c\,A33 \\ d\,B14 + c\,A14 & d\,B24 + c\,A24 & d\,B34 + c\,A34 \end{bmatrix}$$

```
> r2 := c*Transpose(A) + d*Transpose(B);
```

$$r2 := c \begin{bmatrix} A11 & A21 & A31 \\ A12 & A22 & A32 \\ A13 & A23 & A33 \\ A14 & A24 & A34 \end{bmatrix} + d \begin{bmatrix} B11 & B21 & B31 \\ B12 & B22 & B32 \\ B13 & B23 & B33 \\ B14 & B24 & B34 \end{bmatrix}$$

```
> Equal(r1, r2);
```

true

Por lo tanto, para este tipo de matrices la proposición es verdadera. Verifiquemos ahora la segunda.

```
> Att := Transpose(Transpose(A));
```

$$Att := \begin{bmatrix} A11 & A12 & A13 & A14 \\ A21 & A22 & A23 & A24 \\ A31 & A32 & A33 & A34 \end{bmatrix}$$

```
> Equal(Att, A);
```

true

Por lo tanto hemos verificado que, para este tipo de matrices, se cumple la segunda proposición.

Cálculo de bases para espacios vectoriales y combinaciones lineales

El paquete **LinearAlgebra** contiene la función **Basis(V)**, la cual nos permite encontrar una base para un espacio vectorial. El argumento **V** puede ser un vector, un conjunto de vectores o una lista de vectores. Por ejemplo:

```
> vecs := [ <1 | 0 | 0>, <1 | 1 | 0>, <1 | 1 | 1>, <1 | 2 | 3>];
```

$$vecs := [[1, 0, 0], [1, 1, 0], [1, 1, 1], [1, 2, 3]]$$

```
> Basis(vecs);
```

$$[[1, 0, 0], [1, 1, 0], [1, 1, 1]]$$

Nótese que **Basis** nos da únicamente los tres vectores linealmente independientes. Veamos el siguiente caso:

```
> v1 := <2 | -4 | 1>;
```

$$v1 := [2, -4, 1]$$

```
> v2 := <0 | 3 | -1>;
```

$$v2 := [0, 3, -1]$$

```
> v3 := <6 | 0 | -1>;
```

$$v3 := [6, 0, -1]$$

```
> lb := [v1, v2, v3];
```

$$lb := [[2, -4, 1], [0, 3, -1], [6, 0, -1]]$$

```
> Basis(lb);
```

$$[[2, -4, 1], [0, 3, -1]]$$

Esta función solo nos da dos de los vectores, lo cual indica que el tercero es combinación lineal de los otros dos, y por lo tanto no pueden formar una base. Verifiquemos esto:

Debemos encontrar dos escalares **c** y **d** tales que **v3 = c*v1 + d*v2**.

```
> ec := v3 = VectorAdd(v1, v2, c, d);
```

$$ec := [6, 0, -1] = [2c, -4c + 3d, c - d]$$

Para encontrar **c** y **d** debemos resolver el siguiente sistema de ecuaciones:

```
> res := solve({2*c = 6, -4*c + 3*d = 0, c - d = -1});
```

$$res := \{c = 3, d = 4\}$$

Comprobemos el resultado obtenido:

```
> assign(res[1], res[2]);
> v3 = c*v1 + d*v2;
```

$$[6, 0, -1] = [6, 0, -1]$$

Por lo tanto **v3** es combinación lineal de **v1** y **v2**, y como consecuencia **lb** no puede ser una base. Consideremos ahora el siguiente ejemplo:

```
> v4 := <1 | 0 | -1>;
```

$$v4 := [1, 0, -1]$$

```
> v5 := <2 | 5 | 1>;
```

$$v5 := [2, 5, 1]$$

```
> v6 := <0 | -4 | 3>;
```

$$v6 := [0, -4, 3]$$

```
> lb2 := [v4, v5, v6];
```

$$lb2 := [[1, 0, -1], [2, 5, 1], [0, -4, 3]]$$

```
> Basis(lb2);
```

$$[[1, 0, -1], [2, 5, 1], [0, -4, 3]]$$

En este caso, **Basis** nos da como resultado los mismos tres vectores, lo cual indica que generan el espacio de vectores de tres dimensiones y que son linealmente independientes. Verifiquemos la independencia lineal.

Sean:

```
> ec1 := v4 = VectorAdd(v5, v6, g, h);
```

$$ec1 := [1, 0, -1] = [2g, 5g - 4h, g + 3h]$$

```
> ec2 := v5 = VectorAdd(v4, v6, j, k);
```

$$ec2 := [2, 5, 1] = [j, -4k, -j + 3k]$$

```
> ec3 := v6 = VectorAdd(v4, v5, a, b);
```

$$ec3 := [0, -4, 3] = [a + 2b, 5b, -a + b]$$

Resolvemos el siguiente sistema para verificar si **v4** es combinación lineal de **v5** y **v6**.

```
> solve({2*g = 1, 5*g - 4*h = 0, g + 3*h = -1});
```

No obtenemos ningún resultado, lo cual indica que el sistema no tiene solución, y por lo tanto **v4** no es combinación lineal de **v5** y **v6**.

De la misma forma podemos comprobar para los vectores **v5** y **v6**.

```
> solve({j = 2, -4*k = 5, -j + 3*k = 1});
> solve({a + 2*b = 0, 5*b = -4, -a + b = 3});
```

Ahora, verifiquemos si **v4**, **v5** y **v6** generan cualquier vector de tres dimensiones.

```
> vx := <x | y | z>;
```

$$vx := [x, y, z]$$

Debemos encontrar escalares **a**, **b**, **h**, tales que: **vx = a*v4 + b*v5 + h*v6.**

```
> rel := vx = VectorScalarMultiply(v4, a) + VectorScalarMultiply(v5, b)
> + VectorScalarMultiply(v6, h);
```

$$rel := [x, y, z] = [a + 2b, 5b - 4h, -a + b + 3h]$$

```
> solve({a + 2*b = x, 5*b - 4*h = y, -a + b + 3*h = z}, {a, b, h});
```

$$\{a = \frac{19x}{27} - \frac{8z}{27} - \frac{2y}{9}, b = \frac{4x}{27} + \frac{4z}{27} + \frac{y}{9}, h = \frac{5x}{27} + \frac{5z}{27} - \frac{y}{9}\}$$

Este resultado comprueba que cualquier vector de tres dimensiones puede ser escrito como combinación lineal de **v4**, **v5** y **v6**; es decir, generan cualquier vector de tres dimensiones. Por lo tanto, hemos comprobado que **lb2** es una base.

Finalmente, calcularemos una base para el espacio formado por las columnas y los renglones de la siguiente matriz. Esto puede obtenerse por medio de las funciones **RowSpace** y **ColumnSpace.**

```
> m := Matrix([[2, 3, 5], [1, 0, 1], [0, 1, 1]]);
```

$$m := \begin{bmatrix} 2 & 3 & 5 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

```
> RowSpace(m);
```

$$[[1, 0, 1], [0, 1, 1]]$$

```
> ColumnSpace(m);
```

$$\left[\begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ -2 \end{bmatrix}\right]$$

Otras dos funciones útiles para este tipo de operaciones son las siguientes:

- **SumBasis**. Calcula una base para la suma directa de espacios vectoriales.
- **IntersectionBasis**. Calcula una base para la intersección de espacios vectoriales.

Bases para el espacio nulo y rango de una transformación lineal

La función **NullSpace(A)** nos permite calcular una base para el espacio nulo (kernel) de una transformación lineal definida por la matriz **A**.

Consideremos la transformacion **T**, de los polinomios de grado tres con coeficientes reales sobre los polinomios de grado dos con coeficientes reales, dada por:

$$T : P^3(R) \longrightarrow P^2(R), T(p) = p'$$

Donde **p** es un polinomio de grado tres con coeficientes reales. Esta trasformación está representada por la siguiente matriz:

```
> A := Matrix([[0,1,0,0],[0,0,2,0],[0,0,0,3]]);
```

$$A := \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 3 \end{bmatrix}$$

Para comprobar esto consideremos el siguiente polinomio:

```
> p := a + b*x + c*x^2 + d*x^3;
```

$$p := a + b\,x + 3\,x^2 + 4\,x^3$$

Definimos la transformación:

```
> T := f -> diff(f, x);
```

$$T := f \to \frac{d}{dx} f$$

Calculamos **T(p)**:

```
> T(p);
```

$$b + 6\,x + 12\,x^2$$

El polinomio **p** esta representado por el vector:

```
> v := <x, y, z, w>;
```

$$v := \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

Ahora aplicamos la transformación multiplicando por la matriz:

```
> vt := Multiply(A, v);
```

$$vt := \begin{bmatrix} y \\ 2\,z \\ 3\,w \end{bmatrix}$$

Por otro lado, una base para $P^2(R)$ es:

```
> bas := <1 | x | x^2>;
```

$$bas := [1,\ x,\ x^2]$$

Multiplicamos el vector **vt**, al que se le aplicó la transformación por medio de la matriz, por esta base.

```
> Multiply(bas, vt);
```

$$y + 2\,z\,x + 3\,w\,x^2$$

De este resultado podemos ver que **A** representa la transformación **T**. Ahora, verifiquemos si **T** es lineal. Debemos comprobar que

T(k*p + q) = k*T(p) + T(q)

Para **p**, **q** polinomios de grado tres y **n** un escalar.

Sean:

```
> p := a1 + b1*x + c1*x^2 + d1*x^3;
```

$$p := a1 + b1\,x + c1\,x^2 + d1\,x^3$$

```
> q := a2 + b2*x + c2*x^2 + d2*x^3;
```

$$q := a2 + b2\,x + c2\,x^2 + d2\,x^3$$

```
> T(k*p + q);
```

$$k\,(b1 + 2\,c1\,x + 3\,d1\,x^2) + b2 + 2\,c2\,x + 3\,d2\,x^2$$

```
> k*T(p) + T(q);
```

$$k\,(b1 + 2\,c1\,x + 3\,d1\,x^2) + b2 + 2\,c2\,x + 3\,d2\,x^2$$

Por lo tanto, **T** es una transformación lineal. Ahora, calcularemos una base para el espacio nulo de **T**, así como la dimensión de este espacio.

```
> NullSpace(A);
```

$$\left\{ \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \right\}$$

Esta base es obvia ya que la derivada de un polinomio es cero solo si éste es una constante. Por otro lado, es obvio que la dimensión de este espacio nulo es 1. Además, si **B** es una base para **P3(R)**.

```
> B := 1 + x + x^2 + x^3;
```

$$B := 1 + x + x^2 + x^3$$

Entonces **T(B)** es una base para el rango de **T**.

```
> T(B);
```

$$1 + 2\,x + 3\,x^2$$

Como podemos ver, la dimensión de esta base es **'3'**, es decir:

dim(P3(R)) = dim(kernel(T)) + dim(rango(T)) = 1 + 3 = 4

Este paquete contiene la función **Rank**, la cual nos permite calcular el rango de una matriz, haciendo eliminación Gaussiana sobre los renglones de ésta. Podemos usar dicha función para determinar la dimensión del rango de **T**.

```
> Rank(A);
```

3

Inversa de una matriz

La inversa de una matriz A de n x n, esta definida como una matriz $A^{(-1)}$, de n x n, tal que:

$$A\,A^{(-1)} = A^{(-1)}\,A = I$$

Una matriz que tiene inversa se dice que es invertible.

El paquete **LinearAlgebra** nos proporciona la función **MatrixInverse**, la cual calcula la inversa de una matriz. Por ejemplo:

```
> A := Matrix([ [1, 4, 5], [2, 8, 3], [3, 6, 9]]);
```

$$A := \begin{bmatrix} 1 & 4 & 5 \\ 2 & 8 & 3 \\ 3 & 6 & 9 \end{bmatrix}$$

```
> Ainv := MatrixInverse(A);
```

$$Ainv := \begin{bmatrix} \frac{-9}{7} & \frac{1}{7} & \frac{2}{3} \\ \frac{3}{14} & \frac{1}{7} & \frac{-1}{6} \\ \frac{2}{7} & \frac{-1}{7} & 0 \end{bmatrix}$$

Verifiquemos este resultado:

```
> A.Ainv;
```

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

```
> Ainv.A;
```

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Consideremos la siguiente matriz y verifiquemos que tiene inversa:

```
> B := Matrix([[2, 9, 3], [1, 8, 4], [3, 6, 8]]);
```

$$B := \begin{bmatrix} 2 & 9 & 3 \\ 1 & 8 & 4 \\ 3 & 6 & 8 \end{bmatrix}$$

```
> Binv := MatrixInverse(B);
```

$$Binv := \begin{bmatrix} \frac{20}{31} & \frac{-27}{31} & \frac{6}{31} \\ \frac{2}{31} & \frac{7}{62} & \frac{-5}{62} \\ \frac{-9}{31} & \frac{15}{62} & \frac{7}{62} \end{bmatrix}$$

A continuación, para las matrices particulares **A** y **B**, verificaremos que: **A*B** es invertible. Además, verificaremos que se cumple la siguiente relación:

$$(A\,B)^{(-1)} = B^{(-1)}\,A^{(-1)}$$

```
> MatrixInverse(A.B);
```

$$\begin{bmatrix} \frac{-417}{434} & \frac{-13}{217} & \frac{107}{186} \\ \frac{-71}{868} & \frac{8}{217} & \frac{3}{124} \\ \frac{397}{868} & \frac{-5}{217} & \frac{-29}{124} \end{bmatrix}$$

Dado que la inversa de **A*B** existe, entonces **A*B** es invertible.

Verifiquemos ahora la relación.

```
> ABinv := MatrixInverse(A.B);
```

$$ABinv := \begin{bmatrix} \frac{-417}{434} & \frac{-13}{217} & \frac{107}{186} \\ \frac{-71}{868} & \frac{8}{217} & \frac{3}{124} \\ \frac{397}{868} & \frac{-5}{217} & \frac{-29}{124} \end{bmatrix}$$

```
> BinvAinv := MatrixInverse(B).MatrixInverse(A);
```

$$BinvAinv := \begin{bmatrix} \frac{-417}{434} & \frac{-13}{217} & \frac{107}{186} \\ \frac{-71}{868} & \frac{8}{217} & \frac{3}{124} \\ \frac{397}{868} & \frac{-5}{217} & \frac{-29}{124} \end{bmatrix}$$

Comprobaremos si **ABinv = BinvAinv**:

```
> Equal(ABinv, BinvAinv);
```

true

Por lo tanto la relación se cumple (al menos para este caso particular).

Extracción de renglones, columnas y submatrices

Este paquete también contiene funciones útiles para extraer columnas, renglones y submatrices de una matriz dada. Consideremos la siguiente matriz:

```
> A := RandomMatrix(6, 6);
```

$$A := \begin{bmatrix} -19 & -70 & -45 & -12 & -66 & 61 \\ 51 & -44 & 72 & 98 & -11 & -70 \\ -35 & -48 & 22 & 56 & 35 & 22 \\ -52 & -52 & -34 & -54 & 61 & -81 \\ 71 & 37 & 69 & -88 & 96 & -99 \\ 89 & 58 & 55 & -90 & 53 & -2 \end{bmatrix}$$

Para extraer un renglón o un conjunto de renglones usamos la función:

Row(A, i)

Row(A, i..j)

En el primer caso se extrae el i-ésimo renglón; mientras que la segunda forma extrae los renglones i-ésimo al j-ésimo. Si se desea extraer renglones no consecutivos, estos deben incluirse en forma de una lista, por ejemplo: **Row(A, [2, 4, 7..11])**. Veamos el siguiente caso.

```
> Row(A, 4);
```

$$[-52, -52, -34, -54, 61, -81]$$

```
> Row(A, [1, 3..5]);
```

$$[-19, -70, -45, -12, -66, 61], [-35, -48, 22, 56, 35, 22], [-52, -52, -34, -54, 61, -81],$$
$$[71, 37, 69, -88, 96, -99]$$

De la misma forma, podemos usar las siguientes funciones para extraer columnas:

Column(A, i)

Column(A, i..j)

La primera forma extrae la i-ésima columna, mientras que la segunda extrae de la i-ésima a la j-ésima.

Por ejemplo:

```
> Column(A, 2);
```

$$\begin{bmatrix} -70 \\ -44 \\ -48 \\ -52 \\ 37 \\ 58 \end{bmatrix}$$

```
> Column(A, [1, 3..5]);
```

$$\begin{bmatrix} -19 \\ 51 \\ -35 \\ -52 \\ 71 \\ 89 \end{bmatrix}, \begin{bmatrix} -45 \\ 72 \\ 22 \\ -34 \\ 69 \\ 55 \end{bmatrix}, \begin{bmatrix} -12 \\ 98 \\ 56 \\ -54 \\ -88 \\ -90 \end{bmatrix}, \begin{bmatrix} -66 \\ -11 \\ 35 \\ 61 \\ 96 \\ 53 \end{bmatrix}$$

Otra de las funciones útiles para este fin es:

SubMatrix(A, r, c)

Esta función nos permite extraer la submatriz de $\boldsymbol{A}$ determinada por los renglones **r** y las columnas **c**. Estos parametros pueden estar dados por un número, un rango o una lista de números y rangos (de la misma forma que en **Row** y **Column**). Por ejemplo:

```
> SubMatrix(A, [1..3, 5], 1..4);
```

$$\begin{bmatrix} -19 & -70 & -45 & -12 \\ 51 & -44 & 72 & 98 \\ -35 & -48 & 22 & 56 \\ 71 & 37 & 69 & -88 \end{bmatrix}$$

Otra manera de generar una submatriz es eliminando columnas y/o renglones de la matriz original. Para este tipo de operaciones se pueden usar las funciones:

DeleteRow(A, r)

DeleteColumn(A, c)

Donde **r** y **c** pueden estar dadas en la misma forma que en el caso de **SubMatrix**. Estas funciones generan una matriz a partir de **A**, en la cual se han eliminado los renglones o columnas indicadas por el segundo argumento. Por ejemplo:

```
> DeleteRow(A, 2..4);
```

$$\begin{bmatrix} -19 & -70 & -45 & -12 & -66 & 61 \\ 71 & 37 & 69 & -88 & 96 & -99 \\ 89 & 58 & 55 & -90 & 53 & -2 \end{bmatrix}$$

```
> DeleteColumn(A, 1..3);
```

$$\begin{bmatrix} -12 & -66 & 61 \\ 98 & -11 & -70 \\ 56 & 35 & 22 \\ -54 & 61 & -81 \\ -88 & 96 & -99 \\ -90 & 53 & -2 \end{bmatrix}$$

```
> DeleteColumn(A, 3);
```

$$\begin{bmatrix} -19 & -70 & -12 & -66 & 61 \\ 51 & -44 & 98 & -11 & -70 \\ -35 & -48 & 56 & 35 & 22 \\ -52 & -52 & -54 & 61 & -81 \\ 71 & 37 & -88 & 96 & -99 \\ 89 & 58 & -90 & 53 & -2 \end{bmatrix}$$

Unión de matrices y matrices aumentadas

La función **Matrix**, en las formas:

Matrix([A, B, ...])

Matrix(n, m, [A, B, ...])

nos permite unir horizontalmente dos o más matrices. Por ejemplo:

```
> A := RandomMatrix(3, 3);
```

$$A := \begin{bmatrix} 78 & 32 & 14 \\ 72 & -23 & 42 \\ 50 & -99 & -68 \end{bmatrix}$$

```
> B := RandomMatrix(3, 4);
```

$$B := \begin{bmatrix} 22 & -84 & 84 & -52 \\ -53 & 42 & -79 & 9 \\ 45 & 33 & -30 & -43 \end{bmatrix}$$

```
> C := RandomMatrix(3, 3);
```

$$C := \begin{bmatrix} 72 & 42 & 77 \\ -78 & -95 & 79 \\ -13 & 29 & 47 \end{bmatrix}$$

```
> Matrix([A, B, C]);
```

$$\begin{bmatrix} 78 & 32 & 14 & 22 & -84 & 84 & -52 & 72 & 42 & 77 \\ 72 & -23 & 42 & -53 & 42 & -79 & 9 & -78 & -95 & 79 \\ 50 & -99 & -68 & 45 & 33 & -30 & -43 & -13 & 29 & 47 \end{bmatrix}$$

El paquete **linalg** contiene la función **stackmatrix**, la cual une las matrices pero verticalmente. Este paquete también proporciona la función **extend**, la cual puede extender una matriz en **m** renglones y **n** columnas.

Operaciones elementales con matrices

LinearAlgebra también proporciona funciones que nos permiten realizar operaciones elementales sobre matrices. Entre ellas tenemos las que se muestran a continuación.

Utilizaremos la siguiente matriz para llevar a cabo los ejemplos:

```
> A := Matrix([[1, 2, 3], [2, -1, 1], [4, 6, 3]]);
```

$$A := \begin{bmatrix} 1 & 2 & 3 \\ 2 & -1 & 1 \\ 4 & 6 & 3 \end{bmatrix}$$

- Intercambio de dos renglones (o columnas) cualesquiera de **A**. Esto puede hacerse por medio de las funciones:

  RowOperation(A, [r1, r2])

  ColumnOperation(A, [c1, c2])

  Éstas generan una matriz basada en **A**, en la cual se han intercambiado los renglones o columnas indicadas en la lista.

```
> RowOperation(A, [1, 3]);
```

$$\begin{bmatrix} 4 & 6 & 3 \\ 2 & -1 & 1 \\ 1 & 2 & 3 \end{bmatrix}$$

- Multiplicación de cualquier renglón (o columna) de **A** por una expresión algebraica. Este puede llevarse a cabo por medio de las funciones:

  RowOperation(A, r, exp)

  ColumnOperation(A, c, exp)

  Éstas funciones generan una matriz a partir de **A**, en la cual se ha multiplicado la columna o renglón indicados por **r** o **c** (según sea el caso) por la expresión dada como tercer argumento.

```
> RowOperation(A, 2, 3*x);
```

$$\begin{bmatrix} 1 & 2 & 3 \\ 6x & -3x & 3x \\ 4 & 6 & 3 \end{bmatrix}$$

```
> ColumnOperation(A, 1, Pi/2);
```

$$\begin{bmatrix} \frac{\pi}{2} & 2 & 3 \\ \pi & -1 & 1 \\ 2\pi & 6 & 3 \end{bmatrix}$$

- Suma de un múltiplo de un renglón o columna a otro renglón o columna. Esta operación puede realizarse por medio de las funciones:

  RowOperation(A, [r1, r2], expr)

  ColumnOperation(A, [c1, c2], expr)

  La primera función da como resultado una matriz basada en **A**, en la cual el renglón **r1** ha sido reemplazado por la expresión:

  Row(A, r1) + expr*Row(A, r2)

De la misma forma, la segunda función da como resultado una matriz basada en **A**, en la que la columna **c1** ha sido reemplazada por:

Column(A, c1) + expr*Column(A, c2)

Por ejemplo, la siguiente instrucción genera una matriz a partir de **A**, en la que el renglón 1 es reemplazado por el renglón 3 multiplicado por 1/8 y sumado al renglón 1.

```
> RowOperation(A, [1, 3], 1/8);
```

$$\begin{bmatrix} \frac{3}{2} & \frac{11}{4} & \frac{27}{8} \\ 2 & -1 & 1 \\ 4 & 6 & 3 \end{bmatrix}$$

Solución de sistemas de ecuaciones lineales

Consideremos el siguiente sistema de m ecuaciones lineales con n incógnitas:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \cdots + a_{2n}x_n = b_2 \\ \vdots \qquad\qquad \vdots \qquad\qquad \vdots \\ a_{m1}x_1 + a_{m2}x_2 + a_{m3}x_3 + \cdots + a_{mn}x_n = b_m \end{cases}$$

Donde a_{ij} y b_i son escalares, para $1 \leq i \leq m$ y $1 \leq j \leq n$.

La matriz de m x n:

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

se le llama *"matriz de coeficientes del sistema"*.

Por otro lado, consideremos los siguientes vectores:

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, B = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$$

El sistema de ecuaciones de arriba lo podemos escribir como:

$$\mathrm{AX} = \mathrm{B}$$

Una forma de resolver este sistema es por medio de la inversa de la matriz B. En caso de que esta matriz exista, entonces la solución del sistema es única y está dada por:

$$X = A^{(-1)}B$$

Por ejemplo, consideremos el siguiente sistema de ecuaciones.

```
> ec1 := x1 + 2*x2 - x3 = 5;
```

$$ec1 := x1 + 2\,x2 - x3 = 5$$

```
> ec2 := x1 + x2 + x3 = 1;
```

$$ec2 := x1 + x2 + x3 = 1$$

```
> ec3 := 2*x1 - 2*x2 + x3 = 4;
```

$$ec3 := 2\,x1 - 2\,x2 + x3 = 4$$

Para crear la matriz de coeficientes podemos usar la función **GenerateMatrix** de **LinearAlgebra**. Su sintaxis es:

GenerateMatrix([eq1, eq2, ..., eqn], [x1, x2, ..., xn])

Donde **eqn** es de la forma:

a*x1 + b*x2 + ... + n*xn

En esta expresión, **"x1, x2, ..., xn"** son las variables de las cuales dependen las ecuaciones. Ésta función nos da una solución de la forma:

M, B

Donde **M** es la matriz de coeficientes del sistema, mientras que **B** es el vector con los términos independientes. Si se incluye la opción **augmented=true**, el resultado es una matriz aumentada, cuya última columna está formada por los términos independientes.

Por medio de esta función podemos crear la matriz de coeficientes y al mismo tiempo el vector con los términos independientes, de la siguiente manera:

```
> (A, B) := GenerateMatrix([ec1, ec2, ec3], [x1, x2, x3]);
```

$$A, B := \begin{bmatrix} 1 & 2 & -1 \\ 1 & 1 & 1 \\ 2 & -2 & 1 \end{bmatrix}, \begin{bmatrix} 5 \\ 1 \\ 4 \end{bmatrix}$$

Ahora creamos el vector **X**:

```
> X := Vector([x1, x2, x3]);
```

$$X := \begin{bmatrix} x1 \\ x2 \\ x3 \end{bmatrix}$$

Primero verificamos si **A** tiene inversa:

```
> Ainv := MatrixInverse(A);
```

$$Ainv := \begin{bmatrix} \frac{1}{3} & 0 & \frac{1}{3} \\ \frac{1}{9} & \frac{1}{3} & \frac{-2}{9} \\ \frac{-4}{9} & \frac{2}{3} & \frac{-1}{9} \end{bmatrix}$$

Entonces, el sistema tiene una solución única, la cual esta dada por

```
> sol := Ainv.B;
```

$$sol := \begin{bmatrix} 3 \\ 0 \\ -2 \end{bmatrix}$$

Para confirmar que esta es la solución debemos verificar que se cumple la relación **A.sol = B**.

```
> A.sol = B;
```

$$\begin{bmatrix} 5 \\ 1 \\ 4 \end{bmatrix} = \begin{bmatrix} 5 \\ 1 \\ 4 \end{bmatrix}$$

Otra función útil para estos casos es **geneqs**, su sintaxis es de la siguiente forma:

GenerateEquations(A, [x1, x2, ... , xn])

GenerateEquations(A, [x1, x2, ... , xn], B)

Donde ***A*** es una matriz de coeficientes, **x1, x2, ... xn** son las variables de las cuales depende el sistema y ***B*** es el vector con los términos independientes. Esta función genera las ecuaciones correspondientes al sistema representado por la matriz de coeficientes.

Otra forma en la que podemos obtener esta solución es por medio de la función **LinearSolve**, su sintaxis es:

LinearSolve(A)

LinearSolve(A, B)

Donde **A** es una matriz de coeficientes y **B** es una matriz o un vector columna (con los términos independientes). Por ejemplo, resolvamos por este método el siguiente sistema:

```
> ec4 := x1 + 2*x2 - 3*x3 = -2;
```

$$ec4 := x1 + 2\,x2 - 3\,x3 = -2$$

```
> ec5 := x1 + x2 + x3 = -2;
```

$$ec5 := x1 + x2 + x3 = -2$$

```
> ec6 := x1 + x2 - x3 = 0;
```

$$ec6 := x1 + x2 - x3 = 0$$

```
> (A, B) := GenerateMatrix([ec4, ec5, ec6], [x1, x2, x3]);
```

$$A, B := \begin{bmatrix} 1 & 2 & -3 \\ 1 & 1 & 1 \\ 1 & 1 & -1 \end{bmatrix}, \begin{bmatrix} -2 \\ -2 \\ 0 \end{bmatrix}$$

La solución está dada por.

```
> LinearSolve(A, B);
```

$$\begin{bmatrix} 3 \\ -4 \\ -1 \end{bmatrix}$$

Matrices escalonadas

Otro método para resolver sistemas de ecuaciones lineales es obteniendo una matriz escalonada a partir de la matriz de coeficientes. **LinearAlgebra** proporciona varias funciones para escalonar matrices por distintos métodos. Por ejemplo, resolvamos el siguiente sistema:

```
> ec7 := 2*x1 + x2 - 3*x3 = 3;
```

$$ec7 := 2\,x1 + x2 - 3\,x3 = 3$$

```
> ec8 := x1 + 3*x2 + x3 = 2;
```

$$ec8 := x1 + 3\,x2 + x3 = 2$$

```
> ec9 := 3*x1 + x2 - x3 = 1;
```

$$ec9 := 3\,x1 + x2 - x3 = 1$$

La función **GaussianElimination** nos permite obtener una matriz escalonada por el método de Eliminación Gaussiana, para la matriz de coeficientes del sistema. Una vez obtenida esta matriz escalonada, la cual llamaremos **A**, podemos obtener las soluciones del sistema $\mathbf{A}\ \mathbf{X} = \mathbf{B}$, por medio de la función:

BackwardSubstitute(A, B)

donde **A** es la matriz de coeficientes y **B** es el vector con los términos independientes. Esta función también puede ser invocada en la forma:

BackwardSubstitute(A)

donde **A** es la matriz aumentada del sistema.

Aplicaremos estas funciones para resolver el sistema planteado.

GaussianElimination nos permite obtener una forma escalonada para la matriz de coeficientes del sistema, así como para la matriz aumentada. En este caso usaremos la matriz aumentada, la cual podemos generar con **GenerateMatrix**.

```
> A := GenerateMatrix([ec7, ec8, ec9], [x1, x2, x3], augmented=true);
```

$$A := \begin{bmatrix} 2 & 1 & -3 & 3 \\ 1 & 3 & 1 & 2 \\ 3 & 1 & -1 & 1 \end{bmatrix}$$

Aplicamos el método de Eliminación Gaussiana para obtener la matriz escalonada:

```
> E := GaussianElimination(A);
```

$$E := \begin{bmatrix} 2 & 1 & -3 & 3 \\ 0 & \frac{5}{2} & \frac{5}{2} & \frac{1}{2} \\ 0 & 0 & 4 & \frac{-17}{5} \end{bmatrix}$$

Y después usamos **BackwardSubstitute** para resolver $\mathbf{A}\ \mathbf{X} = \mathbf{B}$.

```
> BackwardSubstitute(E);
```

$$\begin{bmatrix} \frac{-3}{10} \\ \frac{21}{20} \\ \frac{-17}{20} \end{bmatrix}$$

Por lo tanto la solución es: $x1 = -\frac{3}{10}$, $x2 = \frac{21}{20}$, $x3 = -\frac{17}{20}$.

Esta función también puede obtener una solución para $\mathbf{A}\ \mathbf{X} = \mathbf{B}$, cuando **B** es una matriz. En este caso la solución es una matriz que cumple la relación.

Otra forma de obtener la matriz escalonada es por medio de la función:

ReducedRowEchelonForm(A)

Ésta aplica el método de Gauss-Jordan para escalonar la matriz. Consúltese su página de ayuda.

Otras funciones proporcionadas por este paquete incluyen:

ForwardSubstitute(A, B)

ForwardSubstitute(A)

la cual resuelve el sistema $\mathbf{A}\ \mathbf{X} = \mathbf{B}$, donde **A** debe ser una matriz triangular inferior. La segunda forma calcula también la solución, pero en este caso **A** debe ser la matriz aumentada del sistema.

Otra función que se puede usar para resolver un sistema de ecuaciones es:

LeastSquares(A, B)

Donde **A** es una matriz (las entradas de ésta también pueden ser dadas en forma de una lista o un conjunto), mientras que **B** es una matriz, un vector columna o un conjunto de variables. Esta función calcula el vector que mejor aproxima la solución de la ecuación $\mathbf{A}\ \mathbf{X} = \mathbf{B}$, esta solución aproximada es obtenida por medio del método de mínimos cuadrados. Véase su página de ayuda.

Determinante de una matriz y matrices adjuntas

El determinante de una matriz puede ser calculado por medio de la función **Determinant(A)**, donde **A** es una matriz cuadrada. Por ejemplo:

```
> A := Matrix([[1, 3, 5], [4, 3, 2], [5, 2, 4]]);
```

$$A := \begin{bmatrix} 1 & 3 & 5 \\ 4 & 3 & 2 \\ 5 & 2 & 4 \end{bmatrix}$$

```
> Determinant(A);
```

$$-45$$

A continuación calcularemos el área del paralelogramo generado por los siguientes vectores:

```
> v1 := Vector[row]([3, 4]);
```

$$v1 := [3, 4]$$

```
> v2 := Vector[row]([-4, 5]);
```

$$v2 := [-4, 5]$$

```
> M := Matrix([[v1], [v2]]);
```

$$M := \begin{bmatrix} 3 & 4 \\ -4 & 5 \end{bmatrix}$$

Esta área está dada por: $\left|\det\left(\begin{bmatrix} x1 & y1 \\ x2 & y2 \end{bmatrix}\right)\right|$

donde *x1*, *y1* son las entradas de *v1*, mientras que *x2*, *y2* son las de *v2*.

```
> area := abs(Determinant(A));
```

$$area := 45$$

Grafiquemos los vectores y el paralelogramo. Primero generamos las gráficas de los vectores:

```
> grafv1 := plots[arrow](v1, shape=arrow):
> grafv2 := plots[arrow](v2, shape=arrow):
```

A continuación generamos la gráfica del paralelogramo:

```
> suma := v1 + v2;
```

$$suma := [-1, 9]$$

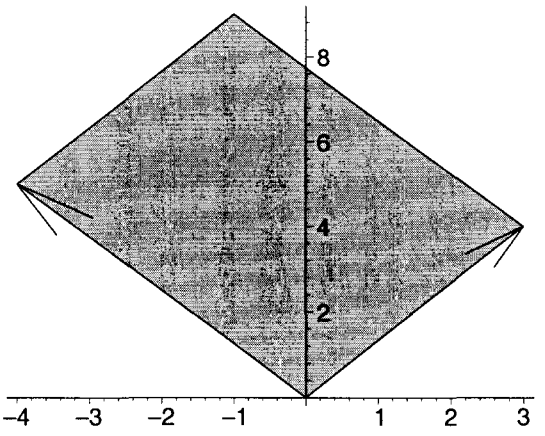
```
> puntos := [[0,0], [v1[1], v1[2]], [suma[1], suma[2]], [v2[1], v2[2]]];
```

$$puntos := [[0, 0], [3, 4], [-1, 9], [-4, 5]]$$

```
> paralel := plots[polygonplot](puntos, color=cyan):
```

Ahora desplegamos las tres gráficas juntas:

```
> plots[display]({grafv1, grafv2, paralel});
```



Por otro lado, la adjunta de una matriz puede ser calculada por medio de la función:

Adjoint(A)

Donde **A** es una matriz cuadrada. Esta función da como resultado una matriz tal que el producto de **A** por la adjunta es igual al producto del determinante de **A** por la matriz identidad. Veamos un ejemplo:

```
> A := Matrix([[2, 4, 3], [4, 2, 5], [3, 2, 5]]);
```

$$A := \begin{bmatrix} 2 & 4 & 3 \\ 4 & 2 & 5 \\ 3 & 2 & 5 \end{bmatrix}$$

```
> Aadj := Adjoint(A);
```

$$Aadj := \begin{bmatrix} 0 & -14 & 14 \\ -5 & 1 & 2 \\ 2 & 8 & -12 \end{bmatrix}$$

Verificaremos que esta matriz cumple la propiedad antes mencionada:

```
> AxAadj := A.Aadj;
```

$$AxAadj := \begin{bmatrix} -14 & 0 & 0 \\ 0 & -14 & 0 \\ 0 & 0 & -14 \end{bmatrix}$$

```
> Adet := Determinant(A);
```

$$Adet := -14$$

```
> Id := IdentityMatrix(3);
```

$$Id := \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

```
> Adet*Id;
```

$$\begin{bmatrix} -14 & 0 & 0 \\ 0 & -14 & 0 \\ 0 & 0 & -14 \end{bmatrix}$$

Por lo tanto la propiedad se cumple.

Valores propios y vectores propios

Los valores propios de una matriz pueden ser calculados por medio de la función:

Eigenvalues(A)

Donde **A** es una matriz cuadrada. Si la matriz **A** contiene valores numéricos, esta función usa un método numérico para determinar los valores propios. Si la matriz contiene expresiones simbólicas, los valores propios son calculados resolviendo el polinomio característico:

Determinant(lambda I - A) = 0

Donde **I** es una matriz identidad de la misma dimension que **A**. Este polinomio característico puede ser calculado con la función **CharacteristicPolynomial**. Su sintaxis es:

CharacteristicPolynomial(A, lambda)

Por ejemplo:

```
> A := Matrix([[1, 4, 3], [1, 5, 2], [3, 6, 4]]);
```

$$A := \begin{bmatrix} 1 & 4 & 3 \\ 1 & 5 & 2 \\ 3 & 6 & 4 \end{bmatrix}$$

```
> CharacteristicPolynomial(A, lambda);
```

$$\lambda^3 - 10\,\lambda^2 + 4\,\lambda + 11$$

```
> evalf(Eigenvalues(A));
```

$$\begin{bmatrix} 9.453812782 - 0.2\,10^{-9}\,I \\ -0.839620113 - 0.7660254040\,10^{-9}\,I \\ 1.385807331 + 0.9660254040\,10^{-9}\,I \end{bmatrix}$$

Por otro lado, los vectores propios de una matriz pueden ser calculados por medio de la función:

Eigenvectors(A)

Donde **A** es una matriz cuadrada. Esta función da como resultado una secuencia de expresiones cuyo primer elemento es un vector con los eigenvalores de **A**, mientras que el segundo elemento es una matriz cuyas columnas son los eigenvectores de **A**; la i-ésima columna de esta matriz es un eigenvector asociado con el i-ésimo eigenvalor del vector devuelto. Por ejemplo, calculemos los vectores propios de la matriz definida anteriormente:

```
> evalf(Eigenvectors(A));
```

$$\begin{bmatrix} 9.453812782 - 0.2\,10^{-9}\,I \\ -0.839620113 - 0.7660254040\,10^{-9}\,I \\ 1.385807331 + 0.9660254040\,10^{-9}\,I \end{bmatrix},$$

$$[0.634780374 + 0.2151270927\,10^{-9}\,I,\ -1.411731932 - 0.8010516764\,10^{-9}\,I,$$

$$0.5269515574 + 0.6707357725\,10^{-9}\,I]$$

$$[0.591578611 - 0.575635464\,10^{-10}\,I,\ -0.1007373862 + 0.2761882708\,10^{-9}\,I,$$

$$-0.6991745564 - 0.1710303189\,10^{-9}\,I]$$

$$[1., 1., 1.]$$

Al invocar esta función se puede incluir la opción **output='values'**, **'vectors'** o **'list'**, dependiendo si se desea obtener los eigenvalores, eigenvectores o ambos, respectivamente.

Otras funciones de LinearAlgebra

En las secciones anteriores solo se han presentado algunas de las funciones contenidas en **LinearAlgebra**, existen otras que también pueden ser usadas en operaciones de matrices y vectores. Entre ellas podemos mencionar las siguientes:

- **IsDefinite**. Determina si una matriz es definida positiva, definida negativa, semidefinida positiva o semidefinida negativa.
- **ConditionNumber**. Calcula el número de condición de una matriz **A**, dado por:
  norm(A)*norm(inverse(A)).
- **IsOrthogonal**. Determina si una matriz es ortogonal.
- **IsUnitary**. Determina si una matriz es unitaria.
- **JordanForm**. Calcula la forma de Jordan de una matriz.
- **MinimalPolynomial**. Calcula el polinomio de menor grado que "anula" a una matriz.
- **MatrixNorm**. Calcula la norma de una matriz.
- **GramSchmidt**. Calcula una lista o conjunto de vectores ortogonales, a partir de una lista o conjunto de vectores linealmente independientes, usando el método de Gram-Schmidt.
- **HermiteForm**. Calcula la forma normal de Hermite de una matriz de n x n de polinomios univariados, sobre el campo de los racionales.
- **SingularValues**. Calcula los valores singulares de una matriz.
- **SmithForm**. Calcula la forma normal de Smith de una matriz.

Consúltese la página de ayuda de estas funciones para obtener más información.

Conclusiones

Este trabajo muestra las características más importantes de Maple 8, con ejemplos que ilustran el uso de la mayoría de las funciones descritas, así como las principales opciones aplicables a éstas. También se presentan los componentes esenciales de la interfaz gráfica, el sistema de ayuda, los elementos sintácticos básicos y las estructuras y tipos usados en el manejo de datos.

Si bien se incluyen pocos ejemplos completos - que muestren las capacidades del sistema para resolver problemas concretos -, cabe señalar que éste en realidad es un trabajo intermedio; la verdadera finalidad del material presentado es convertirse en un libro que, además de describir el sistema y sus principales funciones y características, también ejemplifique el procedimiento para la solución de problemas completos y complejos en diversas áreas de las matemáticas, tales como ecuaciones diferenciales, cálculo y geometría analítica, entre otros.

Este trabajo representa pues una primera etapa en el desarrollo de un texto que proporcione al lector un cúmulo de conocimientos en el uso de Maple, de manera tal que le sean de utilidad para abordar problemas complejos (y simples) de naturaleza científica, aprovechando para ello el poder de las computadoras modernas y el desarrollo actual de los algoritmos en base a los cuales está construido un sistema de álgebra computacional como el descrito aquí.

Esta primera etapa fue planeada para presentar al lector un panorama - medianamente riguroso - de los conocimientos básicos acerca de este sistema. Como parte de una etapa posterior se pretende mostrar la manera en que estos conocimientos se pueden conjuntar para abordar problemas propios de distintas áreas de matemáticas, física y otras disciplinas afines, facilitando las tareas diarias de investigadores, profesores y estudiantes de éstas; proporcionandoles una forma de mecanizar, sobre todo, aquellos procedimientos inmersos en la solución de un problema, susceptibles de ser automatizados.

Una consecuencia de esta automatización puede ser el hecho de poder liberar a la mente de actividades que pueden ser llevadas a cabo de manera eficiente y rápida por una computadora, para ocuparla en la concepción de soluciones a problemas que - hasta ahora - aún no pueden ser resueltos por esta tecnología (al menos no en su totalidad). Finalmente, el objetivo de este tipo de sistemas computacionales - como lo dicen los creadores de Maple - es "la mecanización de las matemáticas", pero solo hasta el punto es que esta "mecanización" se puede concebir como un apoyo al pensamiento humano y no como una alternativa.

Por lo tanto, Maple es presentado aquí como una herramienta de apoyo para la solución de problemas de tipo científico, y no como una manera de sustituir la formación de una persona para el manejo de éstos.

En este mismo sentido, este material es parte de una propuesta para incluir sistemas de este tipo como complemento en los distintos cursos de una carrera científica, que faciliten y refuercen la formación de estudiantes en distintas áreas de las matemáticas, la física y otras afines, sin entrar en conflicto con los métodos tradicionales utilizados actualmente con tal objetivo, y sin intentar sustituir tales cursos.

Bibliografía

[1] Carrillo de Albornoz, Agustín; Llamas Centeno, Inmaculada. *Maple V, aplicaciones matématicas para PC.* Ra-Ma, Madrid, 1995.

[2] Lopez, Robert J. *Maple via calculus: a tutorial approach.* Birkhäuser, Boston, 1994.

[3] Rincón de Rojas, Felix; García López, Alfonsa. *Cálculo científico con Maple.* Ra-Ma, Madrid, 1995.

[4] Redfern, Darren. *Maple V Handbook, Release 5.* Springer-Verlag, New York, 1995.

[5] Heck, Andre. *Introduction to Maple.* Springer-Verlag, New York, 1993.

[6] Friedberg, Sthepen Howard; Insel, Arnold J. *Linear algebra.* Prentice Hall, Englewood Cliffs, New Jersey, 1989.

[7] Atherley, Kate. *An introduction to Maple.* Versión 1.0. Waterloo Maple Software, 1994.

[8] Ibarra Uriarte, Alfonso. *Maple V.*

[9] Char, Bruce W.; Geddes, Keith O.; Gonnet, Gaston H. *Maple V first leaves: a tutorial introduction.* Springer-Verlag, New York, 1992.

[10] Char, Bruce W.; Geddes, Keith O.; Gonnet, Gaston H. *Maple V library reference manual.* Springer-Verlag, New York, 1992.

[11] Char, Bruce W.; Geddes, Keith O.; Gonnet, Gaston H. *Maple V language reference manual.* Springer-Verlag, New York, 1992.

[12] Corless, Robert M. *Essential Maple: an introduction for scientific programmers.* Springer-Verlag, New York, 2002.

[13] Abell, Martha L.; Braselton, James P. *Maple V by example.* AP Professional, Boston, 1994.

[14] Soto Prieto, Manuel Jesús; Vicente Córdoba, José Luis. *Álgebra lineal: con Matlab y Maple.* Prentice Hall, Madrid-México, 1995.

[15] Soto Prieto, Manuel Jesús; Vicente Córdoba, José Luis. *Matemáticas con Maple.* Addison Wesley, Wilmington, Delaware, 1996.

[16] Heal, K. M. *Maple V learning guide.* Springer-Verlag, New York, 1996.

[17] Zachary, Joseph L. *Introduction to scientific programming: computational problem solving using Maple and C.* TELOS, New York, 1996.

[18] Stroeker, Roelof J.; Kaashoek, Johan F. *Discovering mathematics with Maple: an interactive exploration for mathematicians, engineers and econometricians.* Birkhäuser, Boston, Massachusets, 1999.

[19] Kamerich, Ernic. *A guide to Maple.* Springer-Verlag, New York, 1999.

[20] Szabo, Fred. *Linear algebra: an introduction using Maple.* Harcourt/Academic, San Diego, 2002.

[21] Spivak, Michael. *Calculus*, segunda edición. Repla-Reverté, Barcelona, 1988.

[22] Marlin, Joe A.; Kim, Hok. *Calculus I with Maple V.* Departament of Mathematics, North Carolina State University, 1995.

[23] Marlin, Joe A.; Kim, Hok. *Calculus II with Maple V.* Departament of Mathematics, North Carolina State University, 1995.

[24] Marlin, Joe A.; Kim, Hok. *Calculus III with Maple V.* Departament of Mathematics, North Carolina State University, 1995.

[25] *History of Maple.* Departament of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 1993.

[26] Zimmermann, Paul. *A comparison of Maple V.3 and Mathematica 2.2.2.* Laboratoire lorrain de recherche en informatique et ses applications, Nancy, Francia, septiembre de 1994.

[27] Clayton, Martin. *Computer Algebra Software.* CCLRC/Rutherford Appleton Laboratory, Particle Physics & Astronomy Research Council, Starlink Project, Starlink General Paper 47.1, marzo de 1997.