FACULTAT d' ECONOMIA
UNIVERSITAT DE VALENCIA

LICENCIATURA EN C.C. ACTUARIALES Y FINANCIERAS

MATEMÁTICAS AVANZADAS PARA ACTUARIOS

CURSO 2003-2004

PRÁCTICAS.- (Documento 1)

DEPARTAMENTO: **ECONOMIA FINANCERA**
PROFESOR : **MANUEL VENTURA MARCO**
http://www.uv.es/~ventura
e-mail: manuel.ventura@uv.es

INTRODUCCIÓN A LAS PRÁCTICAS

■ Presentación

El "Core Syllabus for Actuarial Training in Europe" propone entre las materias de la etapa preliminar de formación en:

1. Matemáticas
...
4. Informática
...

Sin embargo, el estado actual (lease "planes de estudio") de las titulaciones que dan acceso a la Licenciatura en Ciencias Actuariales y Financieras no garantiza el conocimiento de gran parte de los métodos que se requieren, tanto en uno como en otro caso. Con las prácticas de Matemáticas Avanzadas para Actuarios pretendemos cubrir, con las limitaciones y el tiempo de que disponemos, dichas lagunas. Por eso, y conscientes de que existian otras alternativas, hemos tomado partido por un sistema integrado de algebra computacional: *Mathematica*, que a la vez que nos permite cubrir con solvencia los objetivos y contenidos de 1., también nos permite introducir al estudiante en parte de los contenidos que se indican en 4.

■ Objetivos Generales

1. Una sólida formación en los conceptos y métodos matemáticos que sustentan la comprensión de otras materias de la titulación.

2. Comprensión del proceso de modelización matemática de los problemas financieros y actuariales.

3. Introducir los conceptos y métodos básicos de análisis numérico.

4. Utilización de un sistema de software matemático, en nuestro caso *Mathematica* v. 4.2, en sus vertientes de cálculo simbólico, cálculo numérico, representación gráfica y lenguaje de programación. En su caso, utilización puntual y opcional de otros paquetes y lenguajes informáticos, como Excel combinado con Visual Basic para aplicaciones en Excel.

5. Alternativas y estrategias de resolución, comentando y comparando resultados.

■ Metodología docente

Las clases prácticas de Matemáticas Avanzadas para Actuarios del curso 2003-2004 consistirán en la emisión semanal o quinquenal por parte del profesor de una colección de ejercicios de carácter aplicado al campo económico, financiero y actuarial (se procurará seguir en la medida de lo posible el ritmo de las clases teóricas, siempre de acuerdo con el temario). Dichas colecciones llevarán una fecha de referencia que indicará el día en que serán tratadas en el aula. Con las aclaraciones previas que en cada caso se requieran y el apoyo docente, tanto en el aula como en tutorias y vía correo electrónico, el estudiante deberá proceder a la pertinente modelización y resolución, preferentemente mediante soporte informático (principalmente el paquete de sotware *Mathematica*), y a la posterior discusión de la solución. Con un desfase máximo de dos semanas desde la fecha de referencia, el estudiante hará entrega al profesor, al menos en un 50%, de un informe sobre los ejercicios de la colección(planteamiento y modelización, resolución - con alternativas, en su caso -, discusión de resultados).

El material del curso puede conseguirse en el Servicio de Repografia (Planta Baja del Aulario Sur) o en la dirección electrónica

http://www.uv.es/~ventura/Docencia/MatAvAc/maa.htm

■ Evaluación

En general, la asignatura se superará aprobando una prueba escrita (excepcionalmente puede ser oral) al final de semestre. Dicha prueba consistirá en dos partes: la primera estará compuesta de cuestiones y ejercicios a contestar en papel, y una segunda parte que requerirá el uso del software informático que, en su caso, se utilice como soporte en las clases prácticas.

No obstante, dado que la voluntad es valorarar la participación activa en las clases prácticas, los estudiantes podrán estar exentos en parte o en todo de la segunda prueba. Para que dicha exención sea efectiva deberá acreditarse (mediante firma en la lista habilitada al efecto) la asistencia al menos a 12 de las clases prácticas y la presentación de las colecciones regulares de ejercicios en las condiciones establecidas en el apartado de metodología docente. En función del porcentaje y nivel de correción de los problemas entregados se facilitará con caracter previo al examen una valoración cuya incidencia en la nota final estará en un 30% o 40%.

REFERENCIAS BIBLIOGRÁFICAS

- **Alberca, P. (2000): "Prácticas con Mathematica. Álgebra y Cálculo, Cuaderno I". Ed. Aljibe, Archidona. Págs. 11 a 24 y 73 a 76.**

  S 681.3.06 ALB

- **Alberca, P. (2000): "Prácticas con Mathematica. Ampliación de Cálculo, Cuaderno I". Ed. Aljibe, Archidona. Págs. 20 a 23, 35 a 39 y 56.**

  S 681.3.06 ALB

- **Huang , C.J. and Crooke, P.S. (1997): "Mathematics and Mathematica for Economists". Blackwell. Págs. 1 a 35, 69 a 80, 492 a 508, 510 a 512, 544 a 547, 549 a 551, 605 a 608 y 625 a 629.**

  S 681.3.06 HUA

- **Pérez, C. (1995): "Cálculo simbólico y numérico con Mathematica". Ed. RA-MA, Madrid. Págs. 65 a 77, 86, 108 a 115, 279 a 299, 585 a 592, 609 a 625, 629 a 648, 671 a 679, 682 a 687, 691 a 694 y 697 a 706.**

  S 681.3.06 MATHEPER

- **Cortés, R. y otros autores (2003):"Breve Manual de *MATHEMATICA*". Ed. Universidad Politécnica de Valencia, Valencia.**

RECURSOS ELECTRÓNICOS (ENLACES)

http://www.wolfram.com

http://www.addlink.es

http://library.wolfram.com/infocenter

http://documents.wolfram.com/v4

http://www.wolfram.com/products/mathreader/windows.html

Este último enlace conecta con la página web a partir de la que puede descargar *MathReader*, un programa de aplicación que sólo permite visionar e imprimir los documentos creados con *Mathematica* (notebooks). Tenga en cuenta que con MathReader no puede crear ni editar documentos, ni mucho menos realizar cálculos.

INTRODUCCIÓN A MATHEMATICA

La formación universitaria en el ámbito de las Ciencias Actuariales y Financieras requiere el uso de software matemático como un instrumento activo que capacite para la modelización y resolución de los problemas propios de la disciplina. Entre las principales ventajas de "hacer matemáticas" por ordenador cabe destacar el hecho de que podemos visualizar y experimentar, cuestión altamente complicada con la docencia habitual de resolución de ejercicios en pizarra. Por otro lado, un curso de matemáticas avanzado comporta un número significativo de tediosos y complicados cálculos que oscurecen la comprensión de los principios matemáticos, económicos, financieros y actuariales que subyacen. El software matemático permite obviar los cálculos y concentrarse sobre los aspectos conceptuales y operativos necesarios pàra comprender los principios subyacentes.

En su origen, los ordenadores surgieron para atender con una doble finalidad:

1.- Archivar y ordenar información.
2.- Realizar cálculos numéricos.

Los primeros paquetes individuales que permitían realizar tareas y cálculos específicos de orden numérico, algebraico y/o gráfico aparecen sobre 1960. Por su parte, el nacimiento de los PC's suele datarse en 1981 y desde entonces se produce una proliferación en el uso de microordenadores. Además, a partir de la decada de los 80 del siglo XX se producen importantes avances en el cálculo simbólico y exacto gracias al surgimiento de los denominados lenguajes de 5ª generación, como es el caso de LISP y posteriormente C, lo que permite el desarrollo de programas informáticos de matemática simbólica. En principio estos programas son para máquinas grandes y medianas (workstations), dado que los mismos tienen mayor capacidad de memoria, considerando que el uso de la misma en el cálculo simbólico es exponencial, siendo lineal o polinómico en el cálculo numérico.

Entre las diversas opciones de software, algunas de ellas menos exigentes en recursos, cada una tiene su ventaja comparativa, por lo que no debe sorprender que se utilice más de un paquete. Por ejemplo, las hojas de cálculo: Excel, Lotus 1-2-3, QuattroPro, etc. además de fáciles de usar son muy versatiles para manejar datos y, en especial, para cálculos recursivos con los mismos. Ronald Shone en "Economic Dynamics" (CUP, 1997, pág.17) distingue cuatro clases básicas de software:

1.- Hojas de Cálculo.
2.- Matemáticos: *Mathematica*, Maple, MatLab, Derive, MathCad, etc.
3.- Estadísticos: SPSS, Statgraphics, SAS, etc.
4.- Econometricos: EViews, TSP, Microfit, etc.

Cabe añadir los lenguajes de programación: Frontan, Pascal, C, etc., que en general requieren mayores conocimientos técnicos. En la actualidad, en el contexto de los entornos multimedia son más utilizados Visual Basic o Visual C++ y Java por la facilidad con que permiten crear aplicaciones concretas. De cara a una adecuada formación técnico-científica en el campo financiero-actuarial, lo que ahora se recomienda es iniciarse combinando

un paquete fácil para manejar datos: Excel, un paquete de matemáticas o modelización: M*athematica* (programación de alto nivel), y un lenguaje de bajo nivel: Visual Basic (para desarrollos gráficos y programación interactiva en aplicaciones de Internet) y luego ir evolucionando según las necesidades.

El paquete *Mathematica* es un potentísimo sistema general de software matemático y de aplicaciones, desarrollado desde 1986 por Stephen Wolfram, cuya primera versión data de 1988 (la primera para PC's data del año siguiente), siendo *Mathematica* v2.2. la primera para Windows. En la actualidad existe toda una industria alrededor de este paquete y sus aplicaciones: Wolfram Research, Inc. cuyo campo principal es la Ingeniería de Software. La última versión es *Mathematica* 5, aunque nosostros trabajaremos con la versión *Mathematica* 4.2. Existen versiones para DOS, Windows y UNIX. También hay una versión específica para estudiantes y otra para profesores. Además, existen varias revistas específicas, entre ellas The Mathematica Journal, y varios cientos de libros especializados en *Mathematica* o alguno de los paquetes con aplicaciones adicionales, parte de ellos desarrollados por empresas independientes, entre ellos The Mathematica Book, cuyo autor es el propio S. Wolfram.

Es importante tener presente que *Mathematica* es un sistema completamente integrado de álgebra por ordenador que permite realizar:

* Cálculo numérico
* Cálculo simbólico
* Representación gráfica
* Programación de alto nivel.

De esta forma, es posible disponer de un único y potente sistema que puede tratar conjuntamente todos los aspectos del cálculo científico-técnico de una manera coherente y unificada, desde operaciones aritméticas simples (como una calculadora convencional) hasta sofisticadas operaciones programadas, gracias a su estructura de alto nivel que aprovecha su uso como lenguaje interpretado (lenguaje interpretado: cada entrada produce inmediatamente salida, lenguaje compilado). En el ámbito de la Universidad se utiliza tanto para la docencia como para la investigación, aunque su uso se extiende también a profesionales y técnicos de otras instituciones públicas, empresas y centros de estudios e investigación. En principio su impacto se dio fundamentalmente en Física, Ingeniería y Matemáticas, aunque hoy en día su uso se extiende también a otras disciplinas científico-técnicas: Biología, Economía, Psicología, etc. De hecho, constituye una de las herramientas técnicas habituales en muchas investigaciones y proyectos, tanto a nivel de desarrollo como de difusión de resultados mediante la publicación de documentos técnicos. En lo que concierne al campo económico-financiero, *Mathematica* ha desempeñado un papel más que significativo en el desarrollo de modelos financieros avanzados, siendo también apliamente utilizado en estadística, econometría y en estudios de planificación y análisis, potenciado a su vez el desarrollo de la ciencía computacional y de software aplicada a dichas disciplinas.

Por su facilidad de uso, su enfoque intuitivo y su excelente documentación, *Mathematica* es tal vez el más extendido y consolidado paquete de software matemático en el mundo universitario, aunque no es el único. En la actualidad existe una diversidad de paquetes de cálculo, los más conocidos son: **Macsyma**, primer paquete de software que combinaba cálculo numérico y cálculo simbólico (programado en lenguaje LISP en el M.I.T. en la los años sesenta); **DERIVE** aparece en 1988 como evolución de μMATH (programa en lenguaje LISP de finales de los 70 y principios de los 80 del s. XX para uso en microordenadores); **MatLab** (que significa Laboratorio de Matemáticas), páquete en C y Fortran de análisis numérico y para representación gráfica con amplias capacidades, sobre todo en cálculo matricial, cuyos orígenes se remontan a 1968, y está orientado fundamentalmente a científicos e ingenieros (en la actualidad el más recomendado para iniciarse en tareas de investigación); **Maple**, programado en C, nace en 1983 diseñado para la enseñanza en la Universidad de Waterloo (Canadá), pero en la actualidad ofrece un potente entorno integrado de cálculo numérico, simbólico y gráfico que lo hace ideal para los propósitos de algunos científicos; y **Gauss,** un entorno compatible y de fácil uso, orientado a objetos matemáticos para programación matricial y representación gráfica, fundamentalmente en Estadística y Econometría.

Entorno Operativo de *Mathematica*

■ Inicio de sesión

Para entrar en el programa desde el escritorio de Windows tiene que picar (hacer click) en la tecla de Inicio y deplazarse con el puntero del ratón a Programas de Windows. Cuando se le despliege el menú de programas (en nuestro caso, es el configurado en las Aulas Informáticas del Campus dels Tarongers), tiene que desplázarse al submenú Aplicaciones Docentes, y una vez se despliege éste donde aparece escrito *Mathematica* 4.2 precedido de su icono. Si entonces pica con el puntero del ratón aparecerá la pantalla de inicio en *Mathematica*, en la que suele haber tres elementos: a la izquierda una hoja de trabajo (luego veremos que se denominan notebooks) de nombre Untitled-1, a la derecha una pequeña ventana con simbología matemática (luego se verá que generalmente corresponde a la palette BasicInput), y (opcionalmente si no ha sido desativada en sesiones previas) semioculta entre las dos otra ventana cuyo título es Welcome to *Mathematica*®. Este último elemento, cuya denominación es MathematicaNavigator.nb contiene tres botones que nos llevan respectivamente a un tutorial de 10 minutos (está en inglés, pero vale la pena que entre en el mismo para aprender lo básico sobre como funciona *Mathematica*), al Help Browser o lanzadera de la ayuda de *Mathematica* en Windows, y a la página web de Wolfram Research. Si lo que queremos es empezar a trabajar, podemos ampliar la hoja de trabajo Untitled-1 haciendo click con el ratón en el botón ampliar  de la barra de título de la propia hoja y luego mover el ratón por dicha hoja hasta que el puntero aparezca como un segmento horizontal con bifurcaciones en los extremos; entonces podemos empezar a teclear (o insertar cualquier objeto) y veremos que en el extremo derecho de la hoja aparece un corchete de color azul.

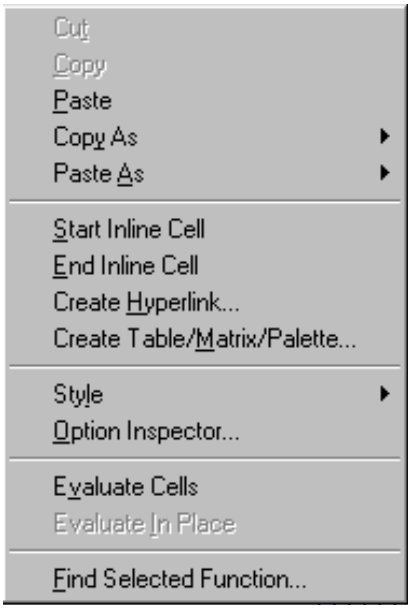
■ Menús de *Mathematica* en Windows

Las versiones del programa *Mathematica* para Windows disponen de una barra con menús desplegables, nueve en *Mathematica* 4.2: File, Edit, Cell, Format, Input, Kernel, Find, Windows y Help. Para acceder a cada uno de ellos basta con desplazar el puntero del ratón hasta que se enmarque (a modo de botón) el elegido y entonces picar sobre el mismo con el ratón.

Si se desplaza al **menú File** y pica en el mismo para que se despliege, podrá observar una serie de seis bloques de opciones, el primero de ellos constituido por las opciones básicas para manejar archivos: New, Open, Close, Save, Save As, Save As Special, Revert (recupera la última versión guardada), Open Special e Import. Es interesante, como luego veremos, el tercer bloque, sobre todo la opción Palettes. El resto de bloques son para enviar el fichero por correo electrónico (no siempre funciona, hace falta que esté bien configurada la opción MAPI de Windows), para imprimir, para recuperar los últimos archivos guardados y, por supuesto, la opción Exit para salir de *Mathematica*.

Por su parte, el **menú Edit** consta de seis bloques, constituyendo los tres primeros las opciones típicas de los menús de edición en entorno Windows: Undo (deshacer), Cut, Copy, ... Los otros tres bloques son más de características específicas para editar en *Mathematica*. El **menú Cell** consta de cinco bloques dedicados a gestionar las celdas o unidades básicas de información de *Mathematica*. El **menú Format**, con seis bloques, está dedicado a todas aquellas acciones que permiten dar formato a las celdas (fuente, aspecto, tamaño, color, etc.) y al documento en general, tanto para trabajar en pantalla como para imprimir. El **menú Input** contiene cinco bloques que agrupan una miscelania de acciones relacionadas con la entrada de información al sistema, permitiendo crear tablas y Arrays; incluiyendo opciones para gráficos, sonidos e Hyperlinks. El **menú Kernel**, con tres bloques, es fundamerntal debido a que desde el mismo podemos gestionar la conexión entre el front end (interfaz de relación entre el sistema y el usuario) y el kernel (núcleo del sistema que realiza cálculos y representaciones gráficas). El **menú Find** es otro de los típicos menús en Windows, sobre todo en el primero de sus tres bloques. Igualmente, los dos últimos menús: **Window** y **Help** son también clásicos en el software de aplicaciones en Windows, por lo que no merecen mayor atención por el momento.

En *Mathematica* también se dispone de un **menú contextual** al que se accede pulsando el botón derecho del ratón:



Es muy útil, ya que cómo puede observarse contiene las instrucciones de uso más habitual, estando disponibles las mismas dependiendo de si la celda está activada o no.

■ Las Celdas de *Mathematica*

La unidad básica de información en Windows se denomina celda y se identifica a modo de corchete de color azul a la derecha de la pantalla (en función del tipo de celda la parte superior del corchete es diferente). A cada celda se le puede asignar un estilo, bien desde la la opción Style del menú Format o del menú contextual, o desde el menú desplegable de la barra de herramientas (si no la tiene en pantalla puede activarla desplegando el menú Format y, en el último bloque, picando en Show Toolbar). De entrada las celdas tienen estilo Input, dado que en principio el programa espera que usted entre información ejecutable para que el programa haga algún cálculo o representación gráfica. No obstante, podemos modificar el estilo picando en el corchete que define la celda, con lo que el mismo se enmarcará de negro apareciendo el corchete de color amarillo o naranja, y entonces elegir el nuevo estilo, por ejemplo Text si quiere escribir algún comentario, Title si quiere poner un título, etc.

El menú Cell ofrece una serie de opciones que permiten manipular las celdas previamente seleccionadas picando con el ratón sobre el corchete que las delimita. Para crear una nueva celda simplemente desplácese con el puntero del ratón hasta el final de la celda en la que se encuentra y cuando el cursor se convierta en un segmento horizontal haga click con el botón izquierdo y verá que aparece una línea horizontal de color negro, entonces pulse la tecla Enter (el retorno de carro). Otra opción es situarnos en la última línea de la celda, apretar la tecla Fin, apretar la tecla de desplazamiento a la derecha (la que tiene una flecha horizontal con la punta mirando al lado derecho) y cuando aparezca la línea negra horizontal al final de la celda, darle a Enter.

■ Estructura de *Mathematica*

El sistema en sí tiene dos partes (técnicamente se dice que es modular): lo que vemos en pantalla es el F**ront End**, que constituye el área de trabajo mediante la cual *Mathematica* inteactua con el usuario (interfaz), y luego está el **Kernel**, que es la parte que se activa cuando ejecutamos una orden y que procede a realizar los cálculos y representaciones gráficas. El Kernel se carga la primera vez que ejecutamos una celda con estilo Input y luego se queda en memoria. La comunicación entre ambos se produce mediante un protocolo de enlace denominado *MathLink,* que sirve también para enlazar con otros lenguales y programas. Así pues, la arquitectura modular de Mathematica le permite funcionar como un potente componente de software.

El Front End más habitual consiste en un documento o fichero de *Mathematica* denominado **Notebook** que tiene la extensión .nb en las versiones más recientes y .ma en las antiguas (ficheros ASCII). En realidad es un conjunto jerarquizado (de modo automático) e interactivo de celdas que junto con las entradas y salidas de resultados combinan texto, tablas, gráficos, cálculos y otros elementos tales como hyperlinks. *Mathematica* se constituye así en más que un programa de cálculo junto con un procesador de textos, ya que permite generar documentos interactivos con calidad de publicación. De hecho, ¡Lo que está viendo es un notebook!.

■ Uso básico de Mathematica

En la actualidad *Mathematica* admite dos formas de introducir Input (ordenes): comando y paletas. El **modo comando o textual** (text-based interface) consiste en teclear toda la orden o conjunto de instrucciones. Por ejemplo, si queremos decirle a Mathematica que integre la función x^2 para valores de $x \in [1, 5]$, haremos:

```
Integrate[x^2, {x, 1, 5}]
```

Aquí puede observarse la estructura del tipo de orden más habitual en *Mathematica*: las funciones predefinidas o propias de *Mathematica (Built-in function)*. Empieza siempre por mayúscula (*Mathematica* es "Case Sensitive"; es decir, sensible a mayusculas y minusculas. Además, advierte si ya existe una variable o función con nombre similar) y los argumentos van encerrados entre corchetes [ ]. Dentro de los corchetes los argumentos se separan por comas y en su caso se agrupan con paréntesis {}. Así, en primer lugar se teclea la expresión u expresiones (en este caso entre llaves y separadas por comas) con las que se quiere trabajar, luego una coma y a continuación se define la variable o variables respecto de las cuales se plantea la orden, que en el caso de que tenga límites se expresa entre llaves colocando primero la variable y separando con comas los límites para los valores de relevantes de la misma. En ocasiones los comandos admiten opciones (ya veremos en cada caso las que nos interesan). Por otra parte, advertir que hay funciones que requieren un número fijo de argumentos, pero hay otras que pueden aplicarse con un número variable de argumentos (**`Log[7]`** da el logaritmo natural o neperiano de 7 pero **`Log[10,7]`** da el logaritmo decimal de 7) e incluso en algunas funciones no es necesario el argumento (**`Random[]`** es uno de los ejemplos más relevantes, generando un número aleatorio real uniformemente distribuido en el rango 0 y 1). Además, casi todas las funciones predefinidas tienen asociado un comando equivalente, en realidad una versión simple de las mismas: expresión // comando.

Aunque debe advertirse que *Mathematica* requiere una sintaxis muy precisa y consistente, al menos si trabajamos en forma estándar (Véase en el menú Cell las opciones del primer bloque), muchos elementos son comunes o similares con los de otros sistemas, paquetes y lenguajes. Así, los operadores aritméticos básicos se expresan, por lo general, son los habituales:

OPERACIÓN	SÍMBOLO
Adición	+
Sustracción	-
Multiplicación	* o SPACE
División	/
Potencia	^

El orden de prelación de los mismos a la hora de agrupar operaciones es también el convencional. En todo caso, si no se tiene claro dicho orden, siempre puede recurrirse a utilizar paréntesis para agrupar términos. En realidad los operadores se corresponden con funciones propias de *Mathematica.* Para conocer la función equivalente a un símbolo o forma especial puede utilizarse la función **`Alias[]`**. Así, por ejemplo:

```
In[1]:= Alias["*"]

Out[1]= Times
```

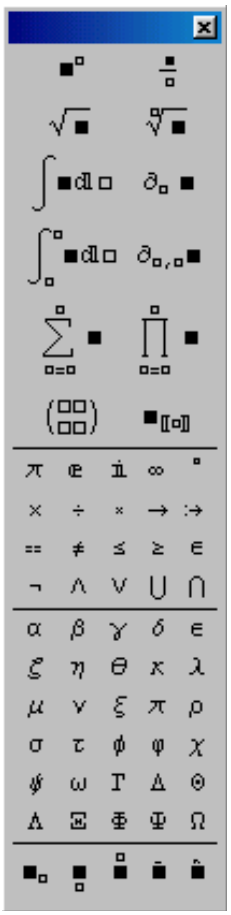
Además, el comando FullForm da como resultado la forma completa (el lenguaje en sí de *Mathematica*) de cualquier expresión:

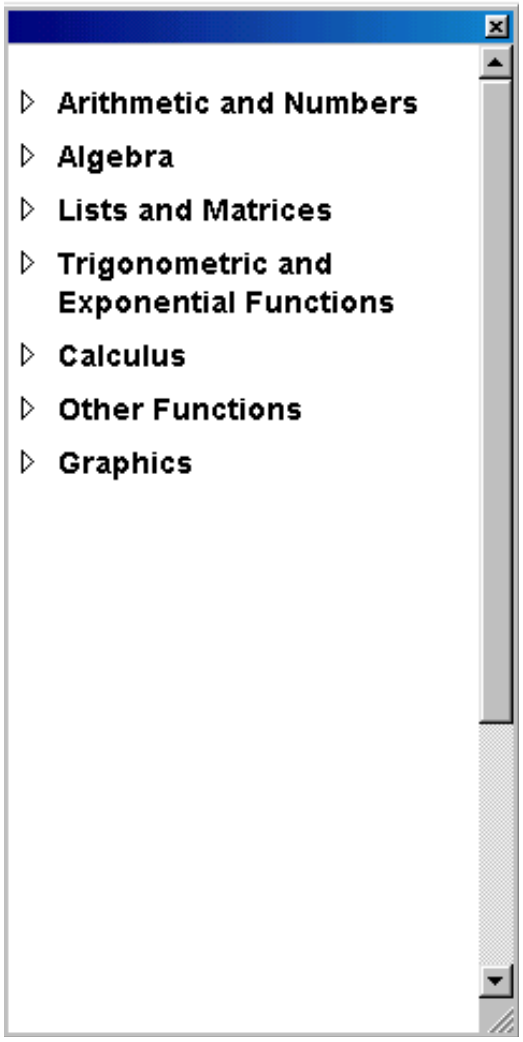
```
In[2]:= C * (1 + i) ^ t // FullForm

Out[2]//FullForm=
        Times[C, Power[Plus[1, i], t]]
```

Trabajar en **modo paletas** permite introducir notación matemática estándar (símbolos). *Mathematica* tiene un conjunto de paletas predefinidas (véase la opción Palettes del menú File, tercer bloque), pero también tiene otra opción que permite crear al usuario sus propias paletas (aunque de momento no entraremos en este punto). Si una paleta no está activada (Véase segundo bloque en menú Window), hay que ir al menú File y

situando el puntero del ratón en la opción Palettes se despliega un submenú. Entre las establecidas por defecto, las dos que más nos interesan son BasicInput y BasicCalculations. Pinchando con el puntero del ratón se activarán si no lo estaban.





Puede observar que se trata de Notebooks independientes constituidas por tablas con botones que al pulsarse causan una determinada acción. Así, por ejemplo, si va a BasicInput y hace click en el cuarto botón de la primera columna obtendrá el siguiente resultado en pantalla

$$\int_{\square}^{\square} \square \, d\square$$

Ahora tiene que rellenar los cuadrados huecos que aparecen. Para el hueco del integrando (función a integrar), suponiendo que es la misma que antes, sitúe el cursor en el cuadrado corespondiente y haga click, con lo que el mismo aparecerá otro cuadrado inscrito menor, y a continuación en la paleta de BasicInput (recupérela desde el menú Window si no la tiene en pantalla) haga click en el primer botón de la primera columna. El resultado en pantalla será:

$$\int_{\square}^{\square} \square^{\square}\, \mathrm{d}\square$$

Ahora ya puede rellenar:

$$\int_{1}^{5} x^2\, \mathrm{d}x$$

Para *Mathematica* esto significa lo mismo que habíamos tecleado en modo textual. Ahora si queremos ejecutarlo para obtener el resultado caben tres opciones: pulsar la combinación de teclas SHIFT (Mayúsculas)+ENTER (Intro) o ↵ (retorno de carro) [en versiones antiguas del programa era la tecla INSERT], ir al menú Kernel y en Evaluation hacer click en la opción , o en el menú contextual picar la opción Evaluate Cells. En cualquier el input se envía al kernel (núcleo), que se carga si es la primera vez de la sesión, y observará que *Mathematica* etiqueta la entrada asignando un núnero a dicha celda, In[1]:= si es la primera vez que ejecutamos y **I[*n*]**:= en general (n es un natural), y a continuación crea una celda de salida con el resultado a la que etiqueta asignándole el mismo número que a la correspondiente celda de entrada, Out[1]= si es el primer resultado de la sesión y en general **Out[*n*]:**=. Veámoslo:

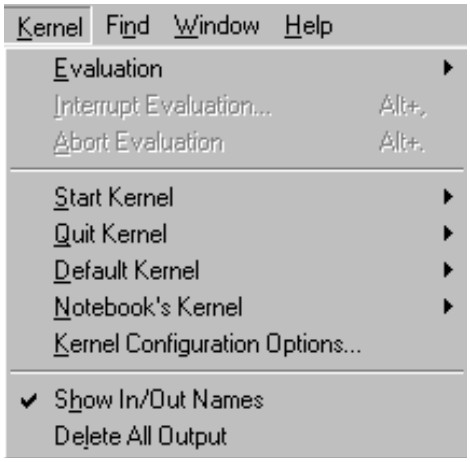
```
In[3]:=
```

$$\int_{1}^{5} x^2\, \mathrm{d}x$$

```
Out[3]=
```

$$\frac{124}{3}$$

No obstante, las etiquetas de Input y Output pueden eliminarse desactivándolas desde la primera opción del tercer bloque del menú Kernel:



Si no queremos que aparezca en pantalla el resultado (lo que se conoce como omitir u obviar output) debemos finalizar el input con ";". Si el kernel no entiende el input produce un beep (zumbido). Puede consultarse en el menú Help la opción Why the Beep?... la causa del mismo. También produce mensajes de advertencia y error. Para interrumpir y abortar el proceso iniciado mediante el kernel , debe activarse desde el menú Kernel las opciones Interrupt Evaluation (equivalentemente: ALT+,) y Abort Evaluation (equivalentemente: ALT+.).

Respecto al uso de entradas y resultados previos como argumentos, simplemente hay que introducir **In[n]** y **Out[n]**. No obstante, también hay símbolos especiales que sirven como referente:

% indica la salida anterior, equivale a **Out[-1]**

%% indica la penúltima salida, equivale a **Out[-2]**

%n recupera la salida n-ésima, equivale a **Out[n]**

Por último, si quiere guardar su trabajo en un fichero, lo puede hacer desde el icono de la barra de herramientas (estándar en Windows) o desde el menú File haciendo click en la opción Save. Si simplemente quiere salir del programa el procedimiento más sencillo consiste en ir al menú File y clickar en la opción Exit.

■ Ayuda en *Mathematica*

Para recabar ayuda puede teclear el nombre del objeto (función, operador, etc.) en cuestión precedido de:

Comando	Información
?objeto	Básica
??objeto	General
?*objeto	Lista

Por ejemplo:

```
In[4]:= ? DSolve

        DSolve[eqn, y, x] solves a differential equation for the function
           y, with independent variable x. DSolve[{eqn1, eqn2, ...  }, {y1,
           y2, ...  }, x] solves a list of differential equations. DSolve[eqn,
           y, {x1, x2, ... }] solves a partial differential equation. More...

In[5]:=
        ?? DSolve

        DSolve[eqn, y, x] solves a differential equation for the function
           y, with independent variable x. DSolve[{eqn1, eqn2, ...  }, {y1,
           y2, ...  }, x] solves a list of differential equations. DSolve[eqn,
           y, {x1, x2, ... }] solves a partial differential equation. More...

        Attributes[DSolve] = {Protected}

        Options[DSolve] = {DSolveConstants → C}
```

```
In[6]:=
?*Solve
```

```
System`
DSolve  LinearSolve  MainSolve  NDSolve  NSolve  Solve
```

```
In[7]:=
?Solve*
```

```
System`
Solve  SolveAlways  SolveDelayed
```

u opcionalmente puede ir al menú Help y picar en Help Browser. En cualquier caso la información está en inglés, así que ¡a traducir!.

Para conocer las opciones que ofrece una función existe a su vez la función **Options[]**.

Números, símbolos y operadores

En lo referente a tipos de números *Mathematica* admite enteros, reales aproximados, racionales, y complejos. *Mathematica* está programado para trabajar simbólicamente, usando aritmética racional. De ahí que de inicio la precisión que aplica en los cálculos es infinita, por lo que ofrece resultados exactos. Esto quiere decir que si al teclear 2/3 en una calculadora el resultado que ofrece es 0.6666667 (aritmética de punto flotante), en *Mathematica* 2/3 es un símbolo en el sistema de los números racionales (aritmética exacta) y no realiza aproximación por expansión decimal. Pero además, los racionales los simplifica automáticamente y los irracionales los trata simbólicamente:

```
In[8]:= n = 45 / 360
```

$$\text{Out[8]} = \frac{1}{8}$$

```
In[9]:= Sqrt[5]
```

$$\text{Out[9]} = \sqrt{5}$$

Respecto de los números complejos, la notación para la parte imaginaris es b*I. Debe recordar que I = $\sqrt{-1}$, lo que puede comprobar en *Mathematica* mediante la siguiente instrucción

```
In[10]:= TrueQ[Sqrt[-1] == I]
Out[10]= True
```

A veces nos puede convenir que *Mathematica* trabaje numéricamenten (que convierta el resultado en un número real aproximado); para lo que recurrimos a la función predefinida **N[ ]**, que puede utilizarse con un único argumento:

```
In[11]:= N[Sqrt[5]]
Out[11]= 2.23607
```

o con dos, indicando el segundo los dígitos de precisión que requerimos:

```
In[12]:= N[Sqrt[5], 18]
Out[12]= 2.23606797749978970
```

La precisión aplicada en un cálculo puede obtenerse mediante la instrucción

```
In[13]:= Precision[%]
Out[13]= 18
```

Una forma alternativa de indicarle a *Mathematica* que queremos como resultado un real aproximado es la siguiente:

```
In[14]:= Sqrt[5.]
Out[14]= 2.23607
```

Por otra parte, *Mathematica* admite input y genera output en notación decimal científica.

```
In[15]:= 3.25*^-5
Out[15]= 0.0000325
```

```
In[16]:= 3.25*^5
Out[16]= 325000.
```

```
In[17]:= 25! // N
Out[17]= 1.55112×10^25
```

■ Símbolos del sistema y operadores

Respecto a símbolos especiales básicos reservados para constantes tenemos: **Pi, E, I, Infinity, Degree**. Aunque pueden teclearse tal cual, lo habitual es recurrir a la paleta BasicInput o teclear la forma abreviada: **Pi** se teclea [ESC]pi[ESC] y se consigue π, **E** se teclea [ESC]ee[ESC] y se consigue **e** , **I** se teclea [ESC]ii[ESC] y se consigue **i** , **Infinity** se teclea [ESC]inf[ESC] y se consigue ∞, y **Degree** (recuerde: Pi/180) se teclea [ESC]Deg[ESC] y se consigue °.

Ya se han mencionado los operadores aritméticos básicos. La notación de los operadores lógicos en *Mathematica* es: **&&** para la conjunción (y lógica), **||** para la disyunción (o lógica), **Xor[ ]** para la exclusión (o lógica excluyente), **!** para la negación, **True** para verdadero y **False** para falso.

En relación con símbolos de comparación la notación es: **==** (teclear dos veces igual) para igualdad, **<=** para menor o igual, **>=** para mayor o igual, **>** para extrictamente mayor, **<** para extrictamente menor ,y **!=** para distinto.

Variables y funciones

■ Variables

En el entorno de *Mathematica* se entiende por variable cualquier expresiíon que no sea un número, una palabra reservada, o una función del propio sistema. En principio, como en otros lenguajes de alto nivel (MatLab, por ejemplo), no se requiere declarar el tipo de variable (el propio sistema las identifica) ni dimensionarlas (más propio de lenguajes de programación de bajo nivel), aunque en ocasiones es aconsejable. Las variables pueden tomar cualquier valor que corresponda a los tipos de número: Real, Rational, Integer, Complex. También puede ser una cadena de caracteres: String, o incluso un símbolo: Symbol. Además, en un sentido más amplio, pueden asociarse a cualquier objeto: formulas, expresiones, gráficos, sonidos, etc. En lo que respecta a su denominación, existen algunas prácticas habituales (convenios de programación y/o usuarios):

las variables genéricas: x, y, z, u, v, t, ... con minúsculas.

Las variables concretas pueden ir en mayusculas: K, L, M, ...

No obstante, ¡Ojo con los símbolos reservados del sistema!, hay que evitarlos (No podemos utilizar I para denotar al interés ya que identifica la raíz cuadrada de - 1, tampoco E para Efectivo porque ya representa al número o base exponencial, etc.), siendo recomendable usar una palabra entera simple o compuesta que empiece por una letra seguida de combinaciones de letras, dígitos numérico o carácteres.

La asignación de valores o expresiones algebraicas a variables puede ser mediante el operador **`Set[variable, valor]`**, cuyo símbolo equivalente es **`=`** (es equivalente **`x = 1`** que **`Set[x,1]`**), o el operador **`SetDelayed[variable, valor]`**, cuyo símbolo equivalente es **`:=`**. Con la asignación inmediata, símbolo **`=`**, *Mathematica* procede a evaluar el lado derecho y asigna a la variable el resultado de dicha evaluación, pero con la asignación diferida, símbolo **`:=`** , el lado derecho no se evalua mientras la variable no sea utilizada dentro de una determinada instrucción u orden. Para ver la diferencia veamos el siguiente ejemplo:

```
In[18]:= i = 0.015;
         vi = 1 / (1 + i);
         vd := 1 / (1 + i);

In[21]:= vi

Out[21]= 0.985222

In[22]:= vd

Out[22]= 0.985222
```

pero si ahora modificamos el valor de i

```
In[23]:= i = 0.012;
         vi

Out[24]= 0.985222

In[25]:= vd

Out[25]= 0.988142
```

Una asignación puede eliminarse mediante la función **Clear[]**, de hecho se recomienda empezar toda aplicación en *Mathematica* utilizando dicha función respecto de los objetos (variables, funciones, etc.) que vayan a utilizarse; también el operdor **UnSet[variable,valor]**, cuyo símbolo equivalente es **=.** y que borra valores y definiciones; o mediante la función **Remove[]**, que borra variables y funciones, por lo que si los utilizamos a posteriori *Mathematica* no los reconocerá. Tanto **Clear[]** como **Remove[]** admiten comodines si el argumento va entre comillas: " ...*...".

Por defecto *Mathematica* considera todas las variables como globales. A no ser que se indique lo contrario se entiende que las variables una vez definidas están referenciada como un objeto determinado (tiene validez en todo el sistema) y que su valor es el último que se le ha asignado.

■ Funciones

Hay que diferenciar entre las Built-in functions o funciones propias del sistema *Mathematica*, cuya denominación atiende a la notación matemática estándar, y las funciones definidas por el usuario.

■ Funciones del sistema (Built-in functions)

Ya hemos hablado de ellas en la sección de Uso básico de *Mathematica*. Sólo resaltar que permiten estructuras anidadas:

```
In[26]:= N[Exp[Log[E]], 20]

Out[26]= 2.7182818284590452354
```

Un buen resumen de las funciones predefinidas puede encontrarse en la paleta BasicCalculations. Alternativamente a aplicar funciones, lo que se denomina también como notación normal, es usar el operador // seguido del nombre de la función, lo que se denomina notación postfija.

■ Funciones de usuario

Las funciones definidas por el usuario (funciones propias de usuario) tienen la siguiente estructura

```
nombre[argumento1_, argumento2_, ..., argumentok_] :=
```

Aquí la inicial del nombre puede ir en mayuscula o minúscula, aunque suele ser una práctica habitual el utilizar minúscula en la definición de los objetos propios. Los argumentos pueden ser números, símbolos o funciones (estructuras anidadas), pudiendo ser la asignación inmediata, **=**, o diferida, **:=**, que se prefiere, ya que recalcula cada vez que llamamos a la función según el valor que tengan asignadas en ese momento las variables. Si no se

les ha asignado valor a las variables, con el operador **ReplaceAll**, cuyo símbolo equivalente es **/.** , se sustituye en la función la variable por el valor asignado.

Se pueden utilizar funciones para definir otras funciones, lo que es habitual, por ejemplo, en funciones recursivas. Se puede definir funciones condicionalmente (funciones por partes o subdominios) mediante el operador condicional **/;**. Ejemplo:

```
In[27]:= f[x_Integer /; x > 0] := x^0.75;
```

de forma que fuera de su subdomínio de definición no funciona la función. Así, con respecto a la función que acabamos de definir

```
In[28]:= f[5]
Out[28]= 3.3437
```

pero

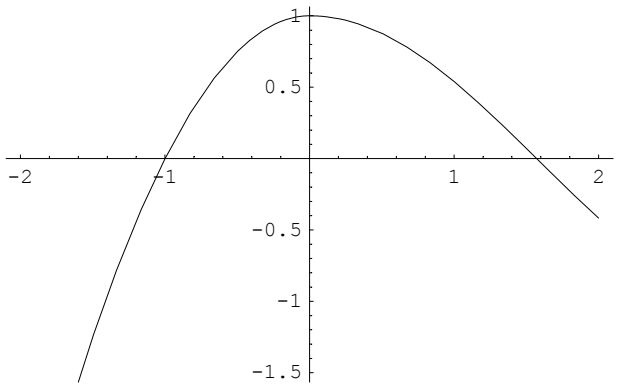
```
In[29]:= f[1 / 4]
Out[29]= f[1/4]
```

o incluso

```
In[30]:= f[0]
Out[30]= f[0]
```

También pueden utilizarse funciones predefinidas como **If[]** , **Which[]**, etc. propias de la programación:

```
In[34]:= Clear[f, x];
         f[x_] := If[x ≥ 0, Cos[x], 1 - x^2];
         Plot[f[x], {x, -2, 2}, AxesOrigin → {0, 0}];
```



Una vez hemos definido una función podemos documentarla para ofrecer información al potencial usuario. El procedimiento es:

```
f::usage = "f[argumento1,argumento2,...,argumentok] devuelve ..."
```

Para obtener información al respecto de la función que hemos definido

```
? f
```

Listas y tablas

Para *Mathematica* una lista es cualquier colección de objetos (números, funciones, símbolos, gráficos, etc.) delimitada por llaves {} y con comas para separar los elementos de la misma. A su vez, una lista puede ser un elemento de otra lista. Con las listas podemos realizar operaciones, asignaciones y aplicarles funciones matemáticas (funciones listables), aunque si se trata de una función que haya definido el usuario primero hay que asignarle a dicha función el atributo de listable

```
AppendTo[Attributes[f], Listable]
```

```
f[lista]
```

o alternativamente aplicarle el operador funcional

```
Map[f, lista]
```

Para referirnos a un elemento de la lista, utilizamos la denominación de la lista y entre dobles corchetes [[ ]] el ordinal del mismo o la función predefinida **`Part[expresion,ordinal]`**. En la sección List and Matrices de la Paleta BasicCalculations están las principales funciones y procedimientos al respecto. Algunas de las funciones para gestionar listas son: **`Length[lista]`** para conocer la longitud de una lista, **`Complement[lista1,lista2]`** para conocer los elementos de una lista que no están en otra, etc. También podemos realizar operaciones con listas, como la intersección de dos listas

```
Intersection[lista1, lista2]
```

o equivalentemente

```
lista1⋂lista2
```

la unión de dos listas

```
Union[lista1, lista2]
```

o equivalentemente

```
lista1⋃lista
```

Para añadir un nuevo elemento a una lista que ya existe hay dos opciones:

```
Union[lista1, nuevoelemento]
```

o

```
AppendTo[lista, terminoadicional]
```

Otra función habitual para operar con listas, que ha menudo se confunde con la unión, es

```
Join[lista1, lista2]
```

En este caso se juntan dos listas con todos sus elementos, aunque se repitan en una y otra. Es obvio que si utilizamos **Union[ ]** o **Join[ ]** con sólo dos elementos (uno en cada argumento) que sean distintos entre sí el resultado es el mismo:

```
In[37]:= lista1 = {2};
         lista2 = {a};
         lista3 = Union[lista1, lista2]

Out[39]= {2, a}

In[40]:= lista1 = {2};
         lista2 = {a};
         lista3 = Join[lista1, lista2]

Out[42]= {2, a}
```

Las listas son de importancia en sí mismas, pero son también importantes debido a que vectores, matrices y arrays se pueden definir en *Mathematica* como listas. No obstante, *Mathematica* tiene también funciones predefinidas para crear listas y obtener resultados. Entre ellas la más recurrida es **Table[ ]**, que permite generar listas siguiendo una determinada estructura (técnicamente se dice que los elementos de la lista siguen un determinado patrón), también generar matrices como resultado de evaluar una expresión con dos subíndices, y también sirve para dar valores a una función

```
f[x_] :=;
nombre = Table[{x, f[x]}, {x, x0, x1, dx}]
```

Para que realmente tenga la apariencia de una tabla con encabezado de columnas

```
TableForm[nombre, TableHeadings → {{}, {"x", "f(x)"}}]
```

TableHeadings es una opción del comando TableForm. El primer juego de llaves, en este caso vacias {}, es para etiquetar las filas.

Un ejemplo habitual es el TRIÁNGULO DE TARTAGLIA

```
In[43]:= TableForm[Table[Binomial[n, k], {n, 0, 10}, {k, 0, n}]]

Out[43]//TableForm=
1
1    1
1    2    1
1    3    3    1
1    4    6    4    1
1    5    10   10   5    1
1    6    15   20   15   6    1
1    7    21   35   35   21   7    1
1    8    28   56   70   56   28   8    1
1    9    36   84   126  126  84   36   9    1
1    10   45   120  210  252  210  120  45   10
```

Si hemos definido una matriz mediante el procedimiento de lista, **M = {lista}**, pero queremos visualizarla en la forma estándar debemos aplicarle la función **MatrixForm[M]**, o equivalentemente

```
M // MatrixForm.
```

Representación Gráfica

■ Representación gráfica 2D

■ Gráficos de funciones

Una de las principales utilidades de Mathematica son las posibilidades de representación gráfica que ofrece, con la ventaja de que los mismos aparecen directamente en pantalla como output. La función **Plot[ ]** es la predefinida para representar en dos dimensiones funciones de una variable. Sus argumentos fundamentales son las funciones a representar (entre llaves y separadas por comas si son mas de una, como en una lista) y la variable con los límites del dominio común (infeior y superior) entre llaves y separado por comas. Como argumentos opcionales podemos introducir

```
PlotStyle → {},
```

que permite definir el estilo de línea mediante las funciones predefinidas: **Thicknees[ ]** para grosor; **RGB[ ]** para una combinación de los colores rojo, verde y azul; **CMYK[ ]** para una combinación más atrevida de colores , **GrayLevel[ ]** para escala de grises, etc. Todas estás funciones admiten como argumentos valores numéricos comprendidos en el rango 0 y 1. Por su parte **Dashing[ ]** determina las marcas del trazado (el punteado). Para dar formato al fondo del gráfico está la opción

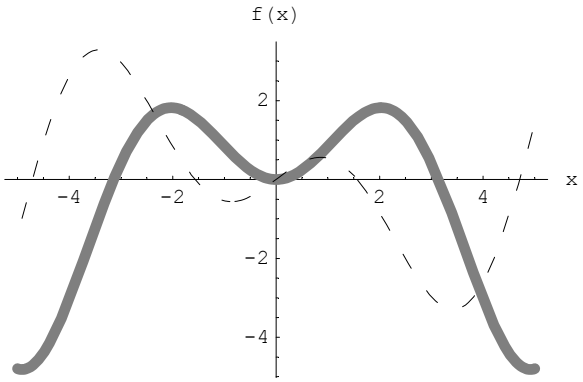
```
Background → { }
```

Mediante **AxesLabel → {"nombreOX","nombreOY"}** podemos etiquetar los ejes, mediante **AxesOrigin → {x0, y0}** podemos definir el orígen de coordenadas para la representación gráfica, y mediante **Axes → False** podemos eliminar los ejes. Por su parte, mediante la opción

```
PlotRange → { }
```

se define el rango de valores para el eje de ordenadas y/o abcisas, ya que de lo contrario el mismo viene dado automáticamente por el programa en función del eje de abcisas (con **PlotRange → All** entran todos los puntos de la curva). El siguiente es un ejemplo ilustrativo:

```
In[44]:= Plot[{x * Sin[x], x * Cos[x]}, {x, -5, 5},
          PlotStyle → {{Thickness[0.02], GrayLevel[0.5]}, {Dashing[{0.03, 0.05}]}},
          AxesLabel → {"x", "f(x)"}, PlotRange → All]
```




```
Out[44]= - Graphics -
```

Argumentos opcionales pueden consultarse mediante el comando Options en forma de reglas establecidas por defecto.

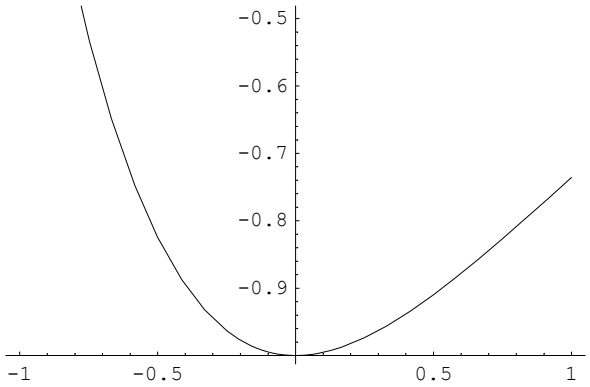
```
In[45]:= Options[Plot]

Out[45]= {AspectRatio → 1/GoldenRatio, Axes → Automatic, AxesLabel → None,
          AxesOrigin → Automatic, AxesStyle → Automatic, Background → Automatic,
          ColorOutput → Automatic, Compiled → True, DefaultColor → Automatic,
          Epilog → {}, Frame → False, FrameLabel → None, FrameStyle → Automatic,
          FrameTicks → Automatic, GridLines → None, ImageSize → Automatic,
          MaxBend → 10., PlotDivision → 30., PlotLabel → None, PlotPoints → 25,
          PlotRange → Automatic, PlotRegion → Automatic, PlotStyle → Automatic,
          Prolog → {}, RotateLabel → True, Ticks → Automatic,
          DefaultFont :→ $DefaultFont, DisplayFunction :→ $DisplayFunction,
          FormatType :→ $FormatType, TextStyle :→ $TextStyle}
```

Puede observarse que hay opciones de marco: **Frame -> True**, **FrameLabel->{"xetiqueta","yetiqueta"}**. Que con **AspectRatio** se determina la escala de los ejes, con **PlotLabel** se le da título al gráfico, con **DisplayFuction -> Identity** se suspende la representación gráfica, y con **GridLines->Automatic** se inserta una retícula.

Puede representarse directamente la función derivada o la función integral

```
In[46]:= Plot[Evaluate[Integrate[x * E^-x, x]], {x, -1, 1}]
```




```
Out[46]= - Graphics -
```

■ Gráficos paramétricos

El procedimiento consiste en crear una función de las variables respecto del parámetro y luego aplicar la función predefinida

```
{x, y} = γ[t_] := {, };

ParametricPlot[γ[t], {t, tmin, tmax}, AspectRatio → 1]
```

■ Gráficos de datos

`ListPlot[ ]` representa un gráfico a partir de los datos contenidos en una

```
lista = {};
ListPlot[lista, PlotStyle → PointSize[], PlotJoined → True]
```

con **`PointSize[ ]`**, cuyo argumento debe estar comprendido enter 0 y 1, se da tamaño a los puntos a representar. Por su parte, con la opción **`PlotJoined → True`** se crea una línea que va uniendo los puntos.

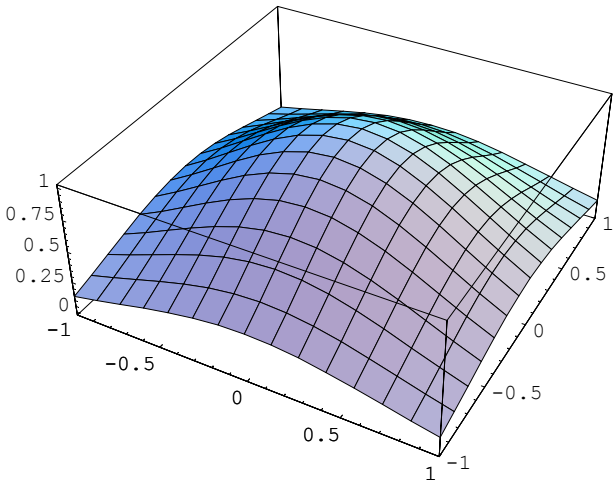
■ Representación 3D

Las funciones de dos variables pueden representarse en gráficos 3D mediante el comando

```
Plot3D[función, {x, xmin, xmax}, {y, ymin, ymax}]
```

que funciona de modo similar al comndo Plot con la salvedad de que sólo es válido para una única función. Por ejemplo:

In[47]:=
```
Plot3D[Exp[-x^2 - y^2], {x, -1, 1}, {y, -1, 1}]
```



Out[47]= - SurfaceGraphics -

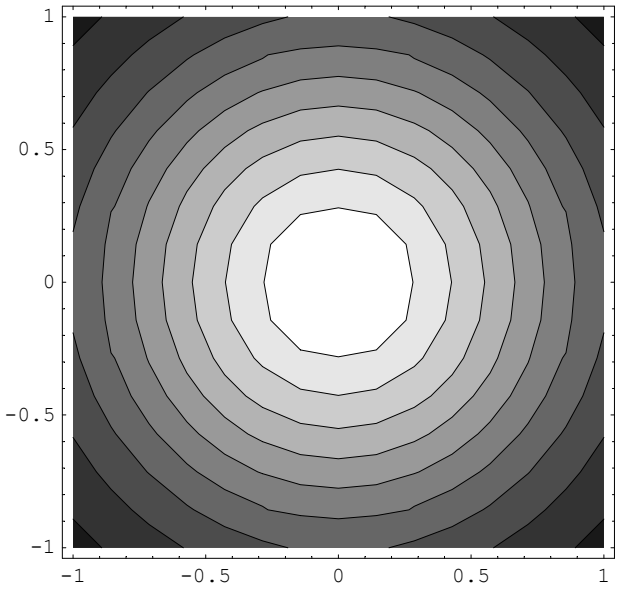
■ Lineas de contorno (Curvas de nivel)

Con funciones de dos variables también es interesante la representación de las líneas de contorno o de nivel

```
ContourPlot[función, {x, xmin, xmax}, {y, ymin, ymax}]
```

Por ejemplo:

In[48]:=
```
ContourPlot[Exp[-x^2 - y^2], {x, -1, 1}, {y, -1, 1}]
```



Out[48]= - ContourGraphics -

■ Combinación de varios gráficos en uno

Representar más de una función requiere el uso de la función

```
Show[gráfico1, gráfico2, ...]
```

que muestra juntos varios gráficos anteriores.

Módulos y paquetes

Un módulo es una agrupación de instrucciones o sentencias (separadas por ;) mediante una función predefinida de *Mathematica* para que actuen de forma conjunta con una finalidad específica. Dicha función es **Module[ ]**, y permite ejecutar sentencias considerando las variables definidas como locales. Hay que tener presente que siempre guardan como resultado el valor de la última sentencia, por lo que si queremos algún resultado la última sentencia debe ser **Return[ ]** o **Print[ ]**.

```
Module[{variables}, sentencias]
```

```
Module[{variables = valores}, sentencias]
```

Vamos a ilustrarlo con un ejemplo:

```
In[49]:= factores[i_, t_] := Module[{acumulacion, descuento},
           {acumulacion = (1 + i) ^ t, descuento = (1 + i) ^ (-t)};
           Print["El factor de acumulación correspondiente a una tasa anual de ",
            i, " y a un plazo de ", t, " años es ", acumulacion,
            " y el factor de descuento es ", descuento, "."]]
```

```
In[50]:= factores[0.1, 10]
```

```
El factor de acumulación correspondiente a una tasa anual de 0.1
  y a un plazo de 10 años es 2.59374 y el factor de descuento es 0.385543.
```

Mathematica es un sistema extensible al que podemos añadir mayor funcionalidad mediante modulos y bloques (similares a los módulos). Un package o paquete es una colección de funciones, módulos, bloques o programas construidos con *Mathematica*. Hay packages que vienen con el propio *Mathematica* (Consulte la secuencia Adds-On / Standar Packages en el Help Browser del menú Help), otros pueden adquirirse a Wolfram Research o en algún distribuidor de software técnico-científico previo pago (es el caso de Finance Essential) o vienen con algún manual (MathEco.m en Huang y Crooke), pero también podemos construir nuestros propios packages.

Los packages se guardan con la extensión .m en un SubDirectorio (generalmente el que contiene los packages que lleva el propio sistema). El procedimiento simple para generarlos es seleccionar las celdas implicadas y en el menú File, desplegar la opción Save As Special y elegir Package Format.
Para cargar en el Kernel un package primero hay que especificar la ruta donde está guardado

```
AppendTo[$Path, "C:directorio"]
```

y luego cargarlo con la función

```
Needs["Contexto`Paquete`"]
```

o recurriendo al operador **Get**, o su símbolo equivalente <<, que permite cargar cualquier fichero

```
<< RutaAcceso/nombrearchivo.m
```

Un ejemplo:

```
valoractual[C_, n_, i_] := C * (1 - ((1 + i) ^ (-n))) / i;
valorfinal[C_, n_, i_] := C * (((1 + i) ^ n) - 1) / i;

Save["a:\RConstante.m", {valoractual, valorfinal}]
```

para recuperarlo

```
In[51]:= << a:\RConstante.m

In[52]:= valoractual[100, 2, 0.025]

Out[52]= 192.742

In[53]:=
         valorfinal[100, 2, 0.025]

Out[53]= 202.5
```