Guía rápida para el nuevo usuario de *Mathematica* 5.0®

Por Eugenio Manuel Fedriani Martel (efedmar@dee.upo.es)

y Alfredo García Hernández-Díaz (agarher@dee.upo.es)

Universidad Pablo de Olavide

Mayo de 2004

Introducción

Estos apuntes son una aproximación al programa *Mathematica* en su versión 5.0. No se le suponen conocimientos previos al lector sobre el programa, pero puede complementarse con otros muchos manuales relativos al mismo.

En el programa *Mathematica* se pueden distinguir dos grandes partes. Una de ellas, llamada núcleo (*Kernel*), es la encargada de ejecutar todos los comandos y realizar los cálculos necesarios. La otra parte es la interfaz del usuario (*Front-End*). Existe un tipo especial de *Front-End* que permite generar documentos interactivos en los que se mezclan gráficos y textos y en el que se incluirán todos los comandos a evaluar por el núcleo; a ese tipo de documentos se los denomina *Notebooks*. La forma de interaccionar entre estas dos partes la dirige el usuario, es decir, que el núcleo no realiza ninguna acción hasta que el usuario no se lo indique y para ello se puede pulsar las teclas *Shift* y *Enter* simultáneamente. En caso de pulsar la tecla *Enter* únicamente, la entrada no se evaluará, simplemente se cambia de línea para poder introducir otro *input*. Una vez pulsado *Shift* y *Enter*, se evaluarán todos los *inputs* introducidos.

Una vez cargado el núcleo, se está en condiciones de comunicarse con el programa. A un *input* dado por el usuario, *Mathematica* devolverá un *output* que numerará (ambos con el mismo número) secuencialmente a lo largo de una sesión de trabajo. Esto permite hacer referencia en cualquier momento a un resultado obtenido o a una variable definida anteriormente. El símbolo % se utiliza para referirse al *output* inmediatamente anterior; pueden usarse %%, %%%, %%%%, etc. para el penúltimo, antepenúltimo, etc. Otra forma alternativa es usar *%n* donde *n* es el número del output. Además, cada entrada y salida lleva un corchete situado a la derecha de la pantalla delimitando lo que denominaremos celdas (*Cell*). Esta distinción por celdas nos permitirá, entre otras cosas, cambiar el estilo y formato de éstas, suprimirlas fácilmente y agruparlas o desagruparlas por bloques de trabajo. También conviene saber que los

inputs se pueden recalcular cuantas veces se desee (por si queremos cambiar un dato, por ejemplo) y la nueva salida sustituirá a la antigua asignándoles a ambos una nueva numeración.

Otro punto a tener en cuenta es que cuando escribamos comandos deberemos tener cuidado con:

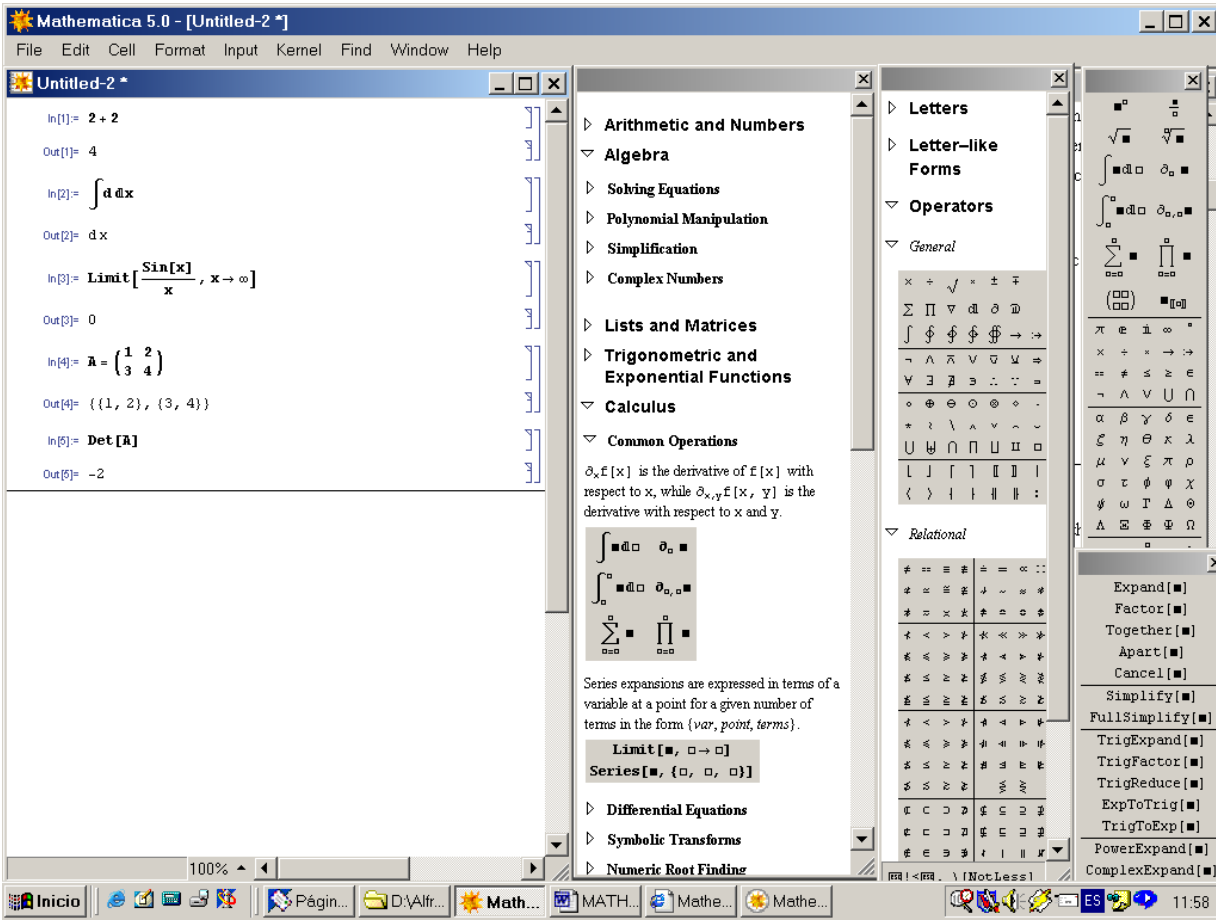
Las mayúsculas y minúsculas. *Mathematica* distingue unos caracteres de otros. Todas las funciones, opciones, variables y constantes incorporadas al programa empiezan con mayúscula (conviene nombrar las que definamos nosotros con minúscula).

Los espacios. Un espacio entre dos variables se interpreta como un signo de multiplicación. Por esto, nunca debemos dejar un espacio entre caracteres cuando demos un nombre a una constante, variable o función.

Paréntesis, corchetes y llaves. Tienen funciones muy diferentes. Los paréntesis se utilizan para agrupar e indican prioridad en las operaciones a efectuar. Los corchetes son exclusivos de las funciones y delimitan el argumento de las mismas (¡no usarlos como un segundo nivel de paréntesis!) y, por último, las llaves se utilizan para definir listas (vectores y matrices, por ejemplo) de elementos.

Una de las novedades más importantes que se introdujo en la versión 3.0 y que ha continuado hasta la versión actual es lo que se denominan paletas (*palettes*). Las paletas son pequeñas ventanas que podemos activar (o desactivar) y que contienen algunas de las operaciones, órdenes e instrucciones más usuales que necesitaremos durante nuestro trabajo. Inicialmente son siete paletas que contienen todo tipo de operaciones, desde las más básicas hasta otras más complejas de Cálculo Algebraico, Cálculo Integral o de Cálculo Diferencial. Pero no solo nos va a servir para tenerlas más a mano, sino que además nos ofrecen la posibilidad de escribir, y por tanto resolver, de la misma forma que escribimos en un folio. Es decir, podemos resolver la integral definida de x entre 0 y 2 tecleando **Integrate[x, {x, 0, 2}]** o usar la Paleta 3 denominada *BasicCalculations* (la podemos encontrar en *File/Palettes/3BasicCalculations*) y escribir $\int_0^2 x\,dx$, lo cual es mucho más intuitivo.

Éste podría ser el aspecto normal de parte de una ventana de *Mathematica*, donde podemos apreciar algunos de los aspectos que ya hemos comentado hasta este punto.



A la izquierda tenemos el *Notebook,* donde hemos realizados algunas operaciones sencillas. Con cada *input* que hemos introducido, y una vez que hemos pulsado *Shift+Enter*, el núcleo nos devuelve un *output* y todos ellos van enumerados consecutivamente. Al realizar la primera operación se activó el núcleo (luego la suma 2+2 no es que haya tardado unos segundos en realizarla sino que primero se cargó el núcleo y luego realizó la operación). El núcleo solo es visible en una pestaña de la barra de tareas, si se pulsa apreciaremos que no ocurre nada, es decir, el núcleo no es visible.

A la derecha tenemos activadas cuatro paletas. La de la izquierda es la Paleta 3 (*BasicCalculations*), la del centro es la Paleta 6 (*CompleteCharacters*) y las dos últimas son: arriba la Paleta 4 (*BasicInput*) y abajo la Paleta 2 (*AlgebraicManipulation*). Estas paletas son las que hemos utilizado para facilitarnos la escritura de la integral, el límite y la matriz que aparecen en el *Notebook*.

Edición de documentos de *Mathematica* (.nb)

Existen diferentes editores de texto específicos para poder visualizar documentos de trabajo de *Mathematica* con la idea de que todo el mundo pueda editarlos, leerlos o modificarlos sin necesidad de adquirir la versión completa del programa. Estos programas nos permitirán únicamente visualizar los *Notebooks* (archivos .nb), es decir, no se podrán utilizar para realizar

operación alguna. Los editores de texto para *Mathematica* no contienen el núcleo de *Mathematica*. El editor que recomendamos usar se denomina *MathReader* y ha sido creado por la misma compañía que *Mathematica* (Wolfram Research, Inc.). Éste se puede descargar gratuitamente en la dirección www.wolfram.com/products/ y nos servirá para visualizar, entre otros, artículos de Economía y Empresa como los disponibles en la página http://library.wolfram.com/infocenter/BySubject/BusinessAndEconomics.

Creación de documentos con *Mathematica*

Otra de las aplicaciones que incorpora *Mathematica* es la posibilidad de crear documentos de trabajo como artículos o libros (por ejemplo, este manual ha sido escrito íntegramente con *Mathematica*). Para ello, *Mathematica* incorpora opciones de formato como la elección del estilo de presentación, la fuente para el texto, el tamaño de la letra, el estilo de letra (negrita, subrayado...), la justificación del texto y muchas opciones más que se pueden consultar en el menú *Format*.

Para su correcta utilización, conviene saber que *Mathematica* distingue entre celdas (*Cells*) del tipo *input*, *output*, celdas de texto o, por ejemplo, una celda para el título del libro (estilo *Title*). De hecho, si escribimos texto dentro de una celda del tipo *input*, *Mathematica* intentará evaluar este texto y, al no comprender lo que escribimos, dará varios errores. Es decir, cuando queramos escribir texto deberemos dividir una celda en dos celdas distintas, la primera para el texto (estilo *Text*) a la que le daremos el formato que deseemos y la segunda para las operaciones y funciones que queremos que calcule *Mathematica* (estilo *input*) las cuales llevan un estilo por defecto, pero que también podemos modificar. Todas las operaciones que conciernen a las celdas (tipo de celda, agrupación o división) se encuentran en el menú *Cell*. También es posible introducir texto dentro de una celda evaluable utilizando (* comentario *).

Uso de la ayuda de *Mathematica*

La ayuda disponible en *Mathematica* es bastante completa y se puede encontrar dentro del menú *Help.* Una vez desplegado dicho menú podemos seleccionar las opciones *Help Browser* o *Master Index,* donde se puede acceder a abundante documentación (ejemplos, enlaces, etc.) que nos pueden servir de gran ayuda.

Desde la ayuda también se hace referencia a numerosos paquetes que, aunque pueden formar parte de la distribución estándar de *Mathematica* 5.0, no pueden ser utilizados hasta que el usuario los cargue previamente (como ya se verá, mediante << o el comando Needs).

Calculadora

Mathematica reconoce los operadores habituales de suma, diferencia, producto, cociente y potenciación:

Operación	**Notación en *Mathematica***
Suma	x+y
Resta	x-y
Producto	x*y o bien x y (un espacio)
Cociente	x/y
Potenciación	x^y

Otras operaciones numéricas usuales son el cálculo del valor absoluto, la raíz cuadrada, la parte entera, el factorial…

Operación	**Notación en *Mathematica***
Valor absoluto de x	Abs[x]
Raíz cuadrada de x	Sqrt[x]
Parte entera de x	Floor[x]
Factorial de x	x! o Factorial[x]
Número aleatorio real entre 0 y 1	Random[]
Máximo y mínimo de una lista de valores	Max[x1, x2,…], Min[x1, x2,…]
Descomposición en factores primos de x	FactorInteger[x]

Ya que éstas son las primeras funciones que aparecen, vamos a realizar algún comentario. En primer lugar destacar que, tal como se comentó al principio, todas las funciones comienzan con mayúsculas (incluso cuando el nombre está formado por varias palabras) y, en segundo lugar, que los argumentos de las distintas funciones siempre van entre corchetes. Además, la mayoría de las operaciones anteriores se encuentran también en varias de las paletas, como la Paleta 3 (*BasicCalculations*) y la Paleta 4 (*BasicInputs*).

Con respecto a la precisión en el cálculo, *Mathematica* tiene precisión *infinita;* es decir, que las operaciones son realizadas en forma exacta o bien con la precisión que le indique el operador. Esto hace que no haya ninguna limitación en cuanto al tamaño máximo de los números enteros que es capaz de manejar salvo la única limitación de la memoria disponible del PC. Probemos con 2^300 :

```
2^300

203703597633448608626844568840937816105146839366593625063614044935
 438129976333670618339737 6
```

Nótese cómo indica que el resultado ocupa más de una línea.

Si queremos aproximar expresiones racionales periódicas o expresiones irracionales, debemos utilizar el comando **N[expresión, número de cifras]**, donde el primero de los argumentos (expresión) corresponde a la cantidad numérica que deseamos aproximar y el segundo de los argumentos (número de cifras) a la cantidad de cifras con la que quieres que se exprese el resultado (parte entera+decimales). Esta orden también la tenemos disponible en la paleta *BasicCalculations/Arithmetic and Numbers*.

Por ejemplo, obtengamos el desarrollo de 123/9990 y de la raíz cuadrada de 3 con 20 cifras exactas:

```
123 / 99990

  41
-----
33330
```

```
N[123 / 99990]

0.00123012
```

```
N[Sqrt[3], 20]

1.7320508075688772935
```

La orden Sqrt[ ] (del Inglés *square root*) es el comando usado para calcular la raíz cuadrada. También aparece en la paleta *BasicCalculations/Arithmetic and Numbers*.

También podemos calcular aproximaciones numéricas indicándole a *Mathematica* que alguno de los números es real y no entero; para ello le pondremos el punto de los decimales y *Mathematica* devolverá el resultado con una precisión por defecto.

Obsérvese que 123/99 no produce la misma salida que 123./99:

```
123/99
```

$$\frac{41}{33}$$

```
123./99
```

```
1.24242
```

Constantes incorporadas

Mathematica tiene un gran número de constantes usuales predefinidas, algunas de ellas son:

Constante	**Notación en *Mathematica***
π	Pi o Esc+p+Esc
E	E o Esc+e+e+Esc
i (unidad imaginaria)	I o Esc+i+i+Esc
∞ (infinito)	Infinity

Si bien "infinito" no es una constante, ya que no es un valor numérico, *Mathematica* la incluye como constante predefinida. Por supuesto, también es posible escribir estas constantes seleccionándolas de las paletas.

Busquemos el valor de π con 20 cifras exactas. Para ello utilizaremos de nuevo la orden **N[]** que sirve para calcular aproximaciones, como ya se ha visto.

```
N[π, 20]
```

```
3.1415926535897932385
```

Funciones usuales

Veamos ahora una tabla de las funciones más usuales y su sintaxis en *Mathematica*:

Función	Notación en Mathematica
$\sqrt{x}$	Sqrt[x]
e^x	E^x o bien Exp[x]
ln(x)	Log[x]
$\log_a(x)$	Log[a,x]
sen(x)	Sin[x]
cos(x)	Cos[x]
tg(x)	Tan[x]
cotg(x)	Cot[x]
sec(x)	Sec[x]
cosec(x)	Csc[x]
arcsen(x)	ArcSin[x]
sh(x)	Sinh[x]
arcsh(x)	ArcSinh[x]

Muchas de estas funciones se encuentran en la paleta *BasicCalculations/ Trigonometric and Exponential Functions.*

Comandos algebraicos usuales

Mediante el comando **`Expand[]`** podemos obtener el desarrollo de las expresiones introducidas (por ejemplo, la aplicación de la propiedad distributiva, el desarrollo del cuadrado del binomio...). Esta orden también se encuentra en la paleta *Algebraic Manipulation*. Pasemos a explicar algunas de las operaciones algebraicas que se encuentran en dicha paleta:

Orden	Significado
Expand[x]	Forma expandida (efectúa sumas, productos, potencias…).
Factor[x]	Factoriza x (escribe x como producto de factores mínimos).
Together[x]	Escribe todos los términos de x con un denominador común.
Apart[x]	Separa x en términos con denominadores lo más simples posible.
Cancel[x]	Cancela factores comunes que posean numerador y denominador.
Simplify[x]	Simplifica x siguiendo reglas algebraicas estándar.
FullSimplify[x]	Simplifica x usando reglas algebraicas más potentes.
TrigExpand[x]	Expande expresiones trigonométricas en suma de términos.
TrigFactor[x]	Factoriza expresiones trigonométricas en producto de términos.

Veamos algunos ejemplos:

```
Expand[(1 – x)^3]

1 - 3 x + 3 x^2 - x^3
```

```
Factor[%]

-(-1 + x)^3
```

```
Expand[(1 + x + 3 y)^4]
Factor[%]
Factor[x^10 – 1]
Expand[%]
```

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

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

$$(-1 + x)\,(1 + x)\,(1 - x + x^2 - x^3 + x^4)\,(1 + x + x^2 + x^3 + x^4)$$

$$-1 + x^{10}$$

Resolución de ecuaciones y sistemas de ecuaciones

Para determinar los ceros de una función, podemos usar los comandos **Solve[ecuación, variables]** y **Nsolve[ecuación, variables] (**Paleta *BasicCalculations/Algebra/SolvingEquations***)** siempre que la función sea polinómica o tenga una forma sencilla de expresión: **Solve[ ]** da el valor exacto de las raíces de la función y **NSolve[ ]** da el valor aproximado de las mismas. En el caso de que la función sea más compleja, se pueden calcular aproximaciones de las raíces utilizando la orden **FindRoot[ ]**, que utiliza métodos numéricos para su resolución.

Ejemplo.- Calculemos las raíces de $f(x)=x^2+5x$:

```
Solve[x^2 + 5 x == 0, x]
```

$$\{\{x \to -5\}, \{x \to 0\}\}$$

Mathematica proporciona la respuesta en el formato que hemos visto. Nótese que debe escribirse el doble signo igual entre la función y el valor cero para indicar que es una ecuación. La **x** que figura después de la coma identifica la variable o incógnita a despejar.

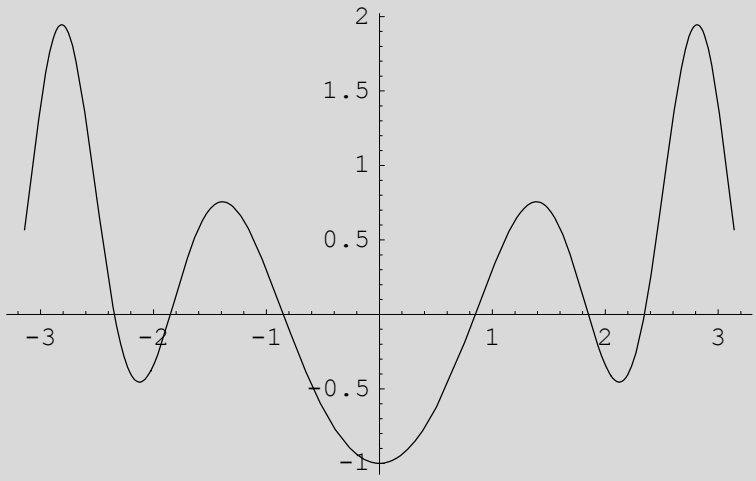
Si intentamos hallar, mediante el comando anterior, las raíces de $g(x)=x^3-7x^2+6x+4$ obtendremos una expresión complicada, que es el valor exacto de cada una de ellas (le sugerimos probarlo). Si solo deseamos una aproximación de las mismas, usaremos el comando **NSolve[]**:

```
NSolve[x^3 – 7 x^2 + 6 x + 4 == 0, x]

{{x → -0.433665}, {x → 1.57414}, {x → 5.85952}}
```

El comando **FindRoot[]** es útil para aproximar algún cero de la función dada. Podemos utilizarla en una de las dos siguientes versiones: **FindRoot[función,{x,x0}]** que utiliza el Método de Newton (o método de la tangente) tomando como punto incial **x=x0** o **FindRoot[función,{x,x0,x1}]** que utiliza una variante del Método de la secante tomando como puntos iniciales **x=x0** y **x=x1**.

Como este comando usa métodos aproximados iterativos, es posible que si damos valores muy alejados para los puntos iniciales, *Mathematica* no pueda encontrar una solución (o la solución que buscamos). Es conveniente hacer primero una gráfica de la función para "ver" dónde se anula la función (más adelante explicaremos algo más sobre las representaciones gráficas). Por ejemplo, hallemos un cero de la función h(x)=sen(x^2)-cos(x):

```
Plot[Sin[x^2] – Cos[x], {x, –Pi, Pi}]
```




```
- Graphics -
```

Observamos que la función presenta un cero (corta al eje de abscisas) en el intervalo (0,1). Por lo tanto hacemos:

```
FindRoot[Sin[x^2] – Cos[x], {x, 0, 1}]

{x → 0.849369}
```

Repitiendo el procedimiento, podremos hallar cualquier otra raíz que nos propongamos.

Para **sistemas de ecuaciones algebraicas** bastará escribir las ecuaciones entre llaves e indicar las variables del sistema.

```
Solve[{2 x + 3 y == 4, x + y == 3}, {x, y}]

{{x → 5, y → -2}}
```

Vectores y matrices

Para definir un vector de tres coordenadas basta con escribir los elementos entre llaves y separados por comas. Igualmente, para definir una matriz bastará escribirla como un vector de vectores que corresponderán a las filas de la matriz. También se puede usar la paleta *BasicCalculations/ Lists and Matrices,* como se verá luego. Definamos dos vectores:

```
v1 = {1, 2, 3}
v2 = {2, 6, −3}

{1, 2, 3}
```

```
{2, 6, -3}
```

Nótese que se ha asignado a dos variables el valor de vectores con el signo "=".

Podemos hacer el producto escalar entre ellos:

```
v1.v2

5
```

o también

```
Dot[v1, v2]

5
```

También el producto vectorial:

```
Cross[v1, v2]

{-24, 9, 2}
```

Y ojo con el resultado de esta multiplicación (elemento a elemento):

```
v1 * v2

{2, 12, -9}
```

Finalmente, para obtener el elemento i-ésimo de un vector haremos **`Part[v,i]`**. Por ejemplo, el segundo elemento de **`v1`** es

```
Part[v1, 2]

2
```

Una vez que hemos visto cómo se introducen vectores manualmente, veamos que también es posible generarlos con la orden **`Table[]`**, cuya definición es

Table[*expr*,{*imax*}]	genera un vector con *imax* copias de *expr*.
Table[*expr,*{*i,imax*}]	genera un vector variando *expr* desde *i=1* hasta i=*imax*.
Table[*expr*,{*i,imin*,*imax*}]	genera un vector variando *expr* desde *i=imin* hasta *i=imax*.
Table[*expr*,{*i,imin*,*imax*,*di*}]	genera un vector variando desde *imin* hasta *imax* con saltos *di*.

Veamos algunos ejemplos:

```
Table[n^2, {n, 4}]

{1, 4, 9, 16}
```

```
Table[Sin[x] / x, {x, 1, 3, 0.2}]

{Sin[1], 0.776699, 0.703893, 0.624734, 0.541026,
 0.454649, 0.367498, 0.281443, 0.19827, 0.119639, 0.04704}
```

```
Table[2 a, {a, 2, 6}]

{4, 6, 8, 10, 12}
```

Pasemos ahora a aspectos básicos del Cálculo Matricial. Definamos un par de matrices:

```
a = {{1, 2, 5}, {−2, 5, 7}, {1, 0, 3}}
b = {{1, 1, 1}, {6, 3, 2}, {1, −3, 0}}

{{1, 2, 5}, {-2, 5, 7}, {1, 0, 3}}
```

```
{{1, 1, 1}, {6, 3, 2}, {1, -3, 0}}
```

Podemos expresarlas en el formato tradicional utilizando la orden **`MatrixForm[]`**:

```
MatrixForm[a]
MatrixForm[b]
```

$$\begin{pmatrix} 1 & 2 & 5 \\ -2 & 5 & 7 \\ 1 & 0 & 3 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 1 & 1 \\ 6 & 3 & 2 \\ 1 & -3 & 0 \end{pmatrix}$$

La multiplicación no es más que **`a.b`**, pero si queremos que el resultado tenga la forma tradicional haremos:

```
MatrixForm[a.b]
```

$$\begin{pmatrix} 18 & -8 & 5 \\ 35 & -8 & 8 \\ 4 & -8 & 1 \end{pmatrix}$$

Además, como es de esperar, podemos multiplicar matrices con vectores y matrices con escalares:

```
MatrixForm[a.v1]
```

$$\begin{pmatrix} 20 \\ 29 \\ 10 \end{pmatrix}$$

```
MatrixForm[2 a]
```

$$\begin{pmatrix} 2 & 4 & 10 \\ -4 & 10 & 14 \\ 2 & 0 & 6 \end{pmatrix}$$

También es posible introducir matrices mediante el uso de las paletas. En varias de las paletas (*BasicInput, BasicCalculation*s y otras) aparece la opción correspondiente a una matriz 2x2. Una vez seleccionada esta opción, podemos añadirle filas y columnas pulsando *Ctrl+Enter* y *Ctrl+","* respectivamente.

En la siguiente tabla se resumen algunas de las operaciones matriciales más usuales, como el cálculo del determinante, los autovalores o valores propios, los autovectores o vectores propios, la inversa, etc.

Operación	**Notación en Mathematica**
Inversa	**Inverse[a]**
Determinante	**Det[a]**
Valores propios	**Eigenvalues[a]**
Vectores propios	**Eigenvectors[a]**
Polinomio característico	**CharacteristicPolynomial[a,x]**
Rango	**MatrixRank[a]**
Traspuesta	**Transpose[a]**
Traza	**Tr[a]**
Reducida por filas	**RowReduce[a]**
Matriz Diagonal	**DiagonalMatrix[{a11,a22,...}]**
Matriz Identidad nxn	**IdentityMatrix[n]**
Espacio nulo de a	**NullSpace[a]**

Det[a]

```
16
```

MatrixForm[Inverse[a]]

$$\begin{pmatrix} \frac{15}{16} & -\frac{3}{8} & -\frac{11}{16} \\ \frac{13}{16} & -\frac{1}{8} & -\frac{17}{16} \\ -\frac{5}{16} & \frac{1}{8} & \frac{9}{16} \end{pmatrix}$$

Eigenvalues[a]

$$\left\{\frac{1}{2}\left(7+\sqrt{17}\right), 2, \frac{1}{2}\left(7-\sqrt{17}\right)\right\}$$

Eigenvectors[a]

$$\left\{\left\{-3+\frac{1}{2}\left(7+\sqrt{17}\right), -5+\frac{3}{4}\left(7+\sqrt{17}\right), 1\right\},\right.$$
$$\left.\{-1, -3, 1\}, \left\{-3+\frac{1}{2}\left(7-\sqrt{17}\right), -5+\frac{3}{4}\left(7-\sqrt{17}\right), 1\right\}\right\}$$

MatrixRank[a]

```
3
```

MatrixForm[Transpose[a]]

$$\begin{pmatrix} 1 & -2 & 1 \\ 2 & 5 & 0 \\ 5 & 7 & 3 \end{pmatrix}$$

RowReduce[a]

```
{{1, 0, 0}, {0, 1, 0}, {0, 0, 1}}
```

```
MatrixForm[%]
```

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Merece especial atención la orden **`Eigensystem[]`**, ya que ésta proporciona directamente los autovalores y autovectores de la matriz en la misma salida. Concretamente, **`Eigensystem[]`** proporciona un primer vector con los autovalores de la matriz y un segundo vector cuyos elementos son autovectores asociados a los autovalores anteriormente ordenados y que, si es posible, forman una base del espacio. En el caso de que no se pueda obtener una base formada por autovectores, *Mathematica* añadirá vectores nulos hasta completar la base (luego no sería una base). Así, tanto **`Eigenvectors[]`** como **`Eigensystem[]`** proporcionan autovectores linealmente independientes en número máximo.

Para la matriz "a" que estamos usando como ejemplo, la orden **`Eigensystem[a]`** proporciona la misma información que la aplicación sucesiva de **`Eigenvalues[a]`** y **`Eigenvectors[a]`**:

```
Eigensystem[a]
```

$$\left\{\left\{\frac{1}{2}\left(7+\sqrt{17}\right), 2, \frac{1}{2}\left(7-\sqrt{17}\right)\right\},\right.$$
$$\left\{\left\{-3+\frac{1}{2}\left(7+\sqrt{17}\right), -5+\frac{3}{4}\left(7+\sqrt{17}\right), 1\right\},\right.$$
$$\left.\left.\{-1, -3, 1\}, \left\{-3+\frac{1}{2}\left(7-\sqrt{17}\right), -5+\frac{3}{4}\left(7-\sqrt{17}\right), 1\right\}\right\}\right\}$$

Para una matriz no diagonalizable, no nos puede proporcionar una base formada por autovectores:

```
c = {{1, 1, 3}, {0, 1, 2}, {0, 0, 2}}
MatrixForm[c]
```

```
{{1, 1, 3}, {0, 1, 2}, {0, 0, 2}}
```

$$\begin{pmatrix} 1 & 1 & 3 \\ 0 & 1 & 2 \\ 0 & 0 & 2 \end{pmatrix}$$

```
Eigensystem[c]

{{2, 1, 1}, {{5, 2, 1}, {1, 0, 0}, {0, 0, 0}}}
```

Nota: En *Mathematica* todo lo que esté escrito entre llaves y debidamente separado por comas es una lista, como un vector o una matriz, y responderá a las propiedades de éstos.

Definición de variables y funciones

Como hemos visto con los vectores, podemos asignarle valores a las variables para así facilitarnos el trabajo. Si queremos que a partir de este momento la variable *x* valga 7 haremos:

```
x = 7

7
```

A partir de este momento cualquier cálculo que realicemos en donde intervenga la ***x***, ésta equivaldrá al valor dado:

```
x + 4

11
```

```
x^2 + x – 3

53
```

Es importante destacar que los valores asignados a las variables son **permanentes**. Una vez que se haya asignado un valor a una variable concreta, el valor permanecerá hasta que no “liberemos” o "limpiemos" a esta variable. Por supuesto, el valor desaparecerá cuando reiniciemos el núcleo o empecemos una nueva sesión de *Mathematica*. Para liberar a las variables bastará con

```
Clear[x]
```

O bien

```
x =.
```

Cuando definimos una función en *Mathematica*, debemos especificar el nombre de la misma y su variable independiente. Por ejemplo:

```
f[x_] := x^2 + 5 x
```

Obsérvese que esta entrada no produce una salida. Esto se debe a los dos puntos delante del signo de igualdad. Lo que hacemos al añadir los dos puntos equivale a un pequeño programa que se ejecutará cada vez que lo llamemos. En cambio, si no añadimos los dos puntos, *Mathematica* ejecutará la función inmediatamente.

Notemos también que la variable ***x*** lleva un guión bajo "_" delante del signo igual. Esto es para que el programa entienda que se trata de una variable muda, es decir, ***x*** puede llevar cualquier nombre o valor. Es importante recordar esto para evitar posibles errores, ya que si se hubiese escrito **`f[x]:=x^2`** se habría asignado el valor **`x^2`** al objeto **`f[x]`** en vez de a la función **`f`** y no entenderá, por ejemplo, **`f[3]`** ó **`f[y]`**.

Podemos calcular el valor numérico de la función **`f`** que hemos definido, para distintos valores de ***x***:

```
f[3]
```

```
24
```

```
f[-3/2]
```

$$-\frac{21}{4}$$

También podemos trabajar con argumentos matriciales o simbólicamente, como vemos en los ejemplos siguientes:

```
Clear[a]
f[a]
```

$$5\,a + a^2$$

```
f[a + h]
```

$$5\,(a+h)+(a+h)^2$$

```
Expand[%]
```

$$5\,a+a^2+5\,h+2\,a\,h+h^2$$

Funciones definidas a trozos

Las funciones definidas a trozos pueden introducirse de diferentes formas. Veámoslas con la función **f(x)=2x**, si **x<=-1**, **f(x)=-x**, si **-1<x<3**, **f(x)=x^2-4**, si **x<=-1**. Como usaremos el nombre **f** para una nueva función, es conveniente "limpiarla" antes de volver a utilizarla:

```
Clear[f]
```

Primera forma:

```
f[x_] := 2 x /; x <= −1;
f[x_] := −x /; −1 < x < 3;
f[x_] := x^2 − 4 /; x >= 3;
```

Los signos de menor o igual y de mayor o igual se escriben como **<=** y **>=** respectivamente. Esto define cada uno de los trozos de **f**, pero se puede definir toda la función de una vez. El ";" al final de cada línea sirve para que, aunque se ejecuten, no produzcan salidas.

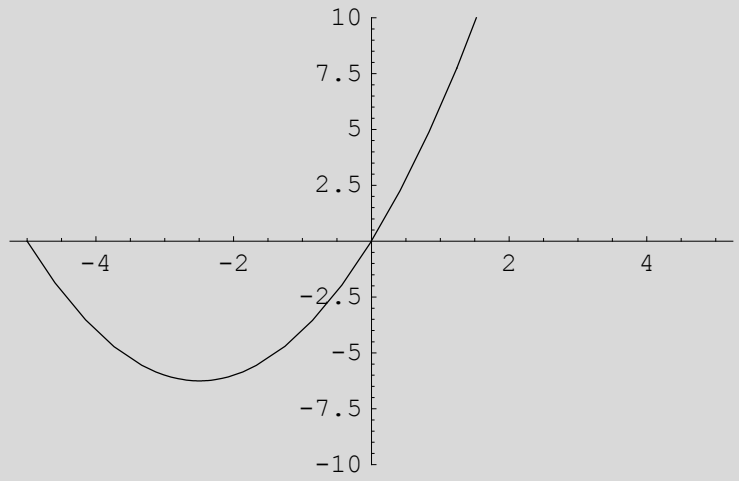
Segunda forma:

Usando el comando **Which[]:**

```
f[x_] := Which[x <= 1, 2 x, −1 < x < 3, −x, x >= 3, x^2 + 4]
```

La gráfica de esta función (véase la sección Representación Gráfica) es:

```
Plot[f[x], {x, −5, 5}, PlotRange −> {−10, 10}]
```




```
- Graphics -
```

Hemos agregado **`PlotRange->{-10,10}`** al comando **`Plot[]`**. Con esto hacemos que represente los valores de la función que están comprendidos entre los valores pedidos, es decir, los valores del recorrido comprendidos entre -10 y 10. Ésta y otras opciones gráficas se verán con más detalle en capítulos posteriores.

Nota: las líneas verticales que aparecen **no forman parte de la gráfica de la función.** Para obtener una gráfica donde estas verticales no aparezcan, podemos utilizar el comando **`Show[]`**, pero no vamos a entrar ahora en más detalles de los necesarios.

Cálculo de límites

Una vez que hemos aprendido a definir funciones, ya podemos manipularlas cómodamente y lo primero que vamos a aprender es a calcular límites. Podemos calcular el valor de la función cos(x^2) en el punto x=0 simplemente reemplazando el valor x por 0 (no es necesario ni conveniente hacer esto para calcular límites, pero así aprendemos a realizar sustituciones).

```
Cos[x^2] /. x −> 0
```

```
1
```

Pero si hacemos esto mismo para algunas funciones especiales, como por ejemplo **`sen(x)/x`**, obtenemos una indeterminación:

```
Sin[x] / x /. x –> 0
```

```
– Power::infy : Infinite expression 1/0 encountered. More…
– ∞::indet : Indeterminate expression 0 ComplexInfinity encountered. More…
```

```
Indeterminate
```

Para encontrar el valor correcto al que tiende esta función en el punto, necesitamos tomar límite. La orden es (también en la Paleta 3, *BasicCalculations/Calculus*)

Limit[expr,x→x0]	Límite de **expr** cuando **x** tiende a **x0.**
Limit[expr,x→x0,Direction→1]	Límite lateral por la izquierda.
Limit[expr,x→x0,Direction→-1]	Límite lateral por la derecha.

Así, haremos

```
Limit[Sin[x] / x, x –> 0]

1
```

Si un límite no existe o vale infinito, obtendremos una salida que así lo indique:

```
Limit[Sin[x] / x^2, x –> 0]

∞
```

Los límites laterales de la función **1/x** en el punto **x=0** serán

```
Limit[1 / x, x –> 0, Direction –> 1]

-∞
```

```
Limit[1 / x, x → 0, Direction → –1]

∞
```

Curiosidad: Obsérvese qué ocurre cuando calculamos el límite de la función **sen(1/x)** cuando **x** tiende a cero.

Derivación

Para calcular la derivada de una función podemos utilizar los símbolos de derivadas parciales que aparecen en la paleta *BasicInput* o directamente alguna de las siguientes órdenes

D[f[x],x]	Derivada (o derivada parcial) de **f** con respecto a **x**.
D[f[x],{x,n}]	Derivada parcial n-ésima de **f** con respecto a **x**.
D[f[x1,x2,...],x1,x2,…]	Derivada parcial de f con respecto a x1, x2, …

La última de las tres funciones es válida para funciones de varias variables. Dedicamos una sección posterior al estudio de la derivación, integración, etc. de este tipo de funciones.

Definamos una nueva función **f** y derivémosla:

```
Clear[f]
f[x_] := x^2 - 3 x + 4
D[f[x], x]

-3 + 2 x
```

También es posible calcular la derivada de una función con **f'[x]**. Es más, también la función **Derivative[]** sirve para el cálculo de derivadas aunque la estructura es algo distinta (véase la Ayuda). Adelantamos únicamente que **Derivative[1][f]** equivale a **f'[x]**.

```
f'[x]
Derivative[1][f]

-3 + 2 x
```

```
-3 + 2 #1 &
```

Obsérvese que la segunda salida es distinta a la segunda. En realidad se ha obtenido el mismo resultado ya que *Mathematica* nos está indicando con los símbolos **#1&** que la constante 2 va multiplicada por la primera de las variables de la función **f**. Esto se debe a que *Mathematica* ha transformado la orden **Derivative[1][f]** en **D[f[#]&,{#,1}]** para resolverla. Así, podemos calcular la derivada primera de la función seno sin especificar la variable ejecutando

```
Derivative[1][Sin]

Cos[#1] &
```

Podemos derivar funciones más complicadas, incluso dependientes de varios parámetros. Definamos, por ejemplo, una función dependiente de dos parámetros, a y b (en caso de que a ó b ya hubiesen sido utilizadas antes y se les hubiese asignado valores se deberían de liberar para lo que usaríamos la orden **Clear[a]**, **Clear[b]** o, directamente, **Clear[a,b]**). La función y su derivada son

```
Clear[a, b]
g[x_] = a Log[x Cos[x]] + b Sin[a b / x]
D[g[x], x]

a Log[x Cos[x]] + b Sin[a b / x]
```

$$-\frac{a\,b^2\,\text{Cos}\left[\frac{a\,b}{x}\right]}{x^2}+\frac{a\,\text{Sec}[x]\,(\text{Cos}[x]-x\,\text{Sin}[x])}{x}$$

Podemos simplificar haciendo:

```
Simplify[%]

a (x - b^2 Cos[a b / x] - x^2 Tan[x]) / x^2
```

O de esta otra manera (ver detalles de estas funciones en la Ayuda)

```
FullSimplify[%]

a (x - b^2 Cos[a b / x]) / x^2 - a Tan[x]
```

Otras de las funciones más frecuentes que nos encontraremos en problemas de varias variables es el cálculo del vector gradiente, **Grad[función,Cartesian[x,y,z]]**. Esta orden se encuentran en el paquete **Calculus`VectorAnalysis`** (que tenemos que cargar previamente):

Veamos un ejemplo:

```
f[x_, y_, z_] := x^2 y^3 + z
<< Calculus`VectorAnalysis`
Grad[f[x, y, z], Cartesian[x, y, z]]

{2 x y^3, 3 x^2 y^2, 1}
```

Integración

Para resolver un problema de integración, como viene siendo habitual, tenemos dos maneras de abordarlo: o bien usamos la orden **`Integrate[]`**, o bien los símbolos de la paleta *BasicInput*. La orden **`Integrate[]`** tiene tres usos distintos:

`Integrate[f,x]` para una primitiva de **`f`** con respecto a la variable **`x`**.
`Integrate[f,{x,xmin,xmax}]` para la integral definida de **`f`** con respecto a **`x`** en el intervalo **`xmin`** y **`xmax`**.
`Integrate[f,{x,xmin,xmax},{y,ymin,ymax}]` para la integral múltiple (doble) de **`f`** con respecto a **`x`** e **`y`** (se verá más adelante).

Veamos si es posible calcular la integral indefinda de **`g(x)=alog(xcos(x))+bsen(ab/x)`** (definida en la sección anterior) con la paleta indicada y con la orden anterior:

$$\int g[x]\,dx$$

$$-a\,x+\frac{1}{2}\,i\,a\,x^2-a\,b^2\,\text{CosIntegral}\left[\frac{a\,b}{x}\right]-a\,x\,\text{Log}[1+e^{2\,i\,x}]+a\,x\,\text{Log}[x\,\text{Cos}[x]]+\frac{1}{2}\,i\,a\,\text{PolyLog}[2,-e^{2\,i\,x}]+b\,x\,\text{Sin}\left[\frac{a\,b}{x}\right]$$

```
Integrate[g[x], x]
```

$$-a\,x+\frac{1}{2}\,i\,a\,x^2-a\,b^2\,\text{CosIntegral}\left[\frac{a\,b}{x}\right]-a\,x\,\text{Log}[1+e^{2\,i\,x}]+a\,x\,\text{Log}[x\,\text{Cos}[x]]+\frac{1}{2}\,i\,a\,\text{PolyLog}[2,-e^{2\,i\,x}]+b\,x\,\text{Sin}\left[\frac{a\,b}{x}\right]$$

Algo feo, pero qué le vamos a hacer. Calculemos una integral impropia dependiente de un parámetro q:

```
Integrate[2 * Exp[-q * x^2], {x, -Infinity, Infinity}]
```

$$2\,\text{If}\left[\text{Re}[q] > 0,\ \frac{\sqrt{\pi}}{\sqrt{q}},\ \text{Integrate}\left[e^{-q\,x^2},\ \{x,\ -\infty,\ \infty\},\ \text{Assumptions} \to \text{Re}[q] \le 0\right]\right]$$

La salida obtenida quiere decir que si la parte real de q es mayor que cero entonces la integral es igual a $2\frac{\sqrt{\pi}}{\sqrt{q}}$ y, si no, no sabe hacerlo.

También es posible calcular integrales numéricas utilizando la orden **NIntegrate[]** que nos proporcionará un valor aproximado en lugar del exacto:

```
NIntegrate[Exp[x^2], {x, -2, 2}]

32.9053
```

Probemos con la integral de **g(x)=aLog(xCos(x))+bSin(ab/x)** en algún intervalo razonable. Por ejemplo, podemos definir una función que dependa de los parámetros **a** y **b**:

```
funcion[a_, b_] := NIntegrate[g[x], {x, Pi/6, Pi/3}]
```

Al intentar evaluar la integral para unos valores concretos de **a** y **b** vemos que **no funciona**:

```
funcion[1, 2]

- NIntegrate::inum : Integrand g[x] is not numerical at {x} = {0.785398}. More…

NIntegrate[g[x], {x, π/6, π/3}]
```

¿Por qué? Esto ocurre porque la dependencia en **a** y **b** no está explícita en **g(x)** y no la entendió. Para evitar estos errores definamos una función **g2(x)** que tenga también como variables a los parámetros **a** y **b**:

```
g2[x_, a_, b_] = a Log[x Cos[x]] + b Sin[a b / x]
```

$$a\,\text{Log}[x\,\text{Cos}[x]] + b\,\text{Sin}\left[\frac{a\,b}{x}\right]$$

Ahora sí es posible definir nuestra "función", ya que la dependencia en **a** y **b** es explícita:

```
funcion2[a_, b_] := NIntegrate[g2[x, a, b], {x, Pi/6, Pi/3}]
```

Ahora evaluamos

```
funcion2[1, 2]

0.109469
```

De esta manera podemos hacerlo para otros valores de a y b:

```
funcion2[−1, 0]

0.330295
```

El símbolo "**:=**" es más que una asignación. Si nos damos cuenta, además de servir como asignación, hace que la operación definida no sea ejecutada inmediatamente sino cuando se le llame más adelante. O sea, es como un pequeño programa que luego será ejecutado.

Funciones de varias variables: derivación parcial e integración múltiple

Vamos a profundizar en algunos de los conceptos que hemos estudiado hasta el momento (cálculo de límites, integración y derivación), pero ahora centrándonos en funciones de varias variables. Apreciaremos que las funciones a utilizar son las mismas, pero que habrá que añadir algún argumento más.

Tomemos la función de dos variables f(x,y)=2xy+sen(2x-3y^2) y realicemos algunas operaciones con ella. Es muy conveniente limpiar antes la "f" si ya se había definido anteriormente. Empecemos calculando los límites unidimensionales y reiterados en el punto (x,y)=(0,0):

```
f[x_, y_] := 2 * x * y + Sin[2 x - 3 y^2]
f1[y_] = Limit[f[x, y], x -> 0]
f2[x_] = Limit[f[x, y], y -> 0]

-Sin[3 y^2]
```

```
Sin[2 x]
```

Los límites reiterados son entonces

```
Limit[f2[y], y -> 0]
Limit[f1[x], x -> 0]

0
```

```
0
```

Con respecto a las derivadas parciales de primer orden tenemos

```
D[f[x, y], x]
D[f[x, y], y]

2 y + 2 Cos[2 x - 3 y^2]
```

```
2 x - 6 y Cos[2 x - 3 y^2]
```

Y las de segundo orden (véase la sección sobre Derivación)

```
D[f[x, y], {x, 2}]
D[f[x, y], {y, 2}]
D[f[x, y], x, y]
D[f[x, y], y, x]
```

$$-4\,\text{Sin}[2\,x-3\,y^2]$$

$$-6\,\text{Cos}[2\,x-3\,y^2]-36\,y^2\,\text{Sin}[2\,x-3\,y^2]$$

$$2+12\,y\,\text{Sin}[2\,x-3\,y^2]$$

$$2+12\,y\,\text{Sin}[2\,x-3\,y^2]$$

Para orden superior podemos usar o bien **`D[f[x,y],{x,n}]`** o bien **`D[f[x,y],x,x,x,...]:`**

```
D[f[x, y], {x, 5}]
```

$$32\,\text{Cos}[2\,x-3\,y^2]$$

```
D[f[x, y], x, x, x, x, x]
```

$$32\,\text{Cos}[2\,x-3\,y^2]$$

Finalmente, si queremos calcular la integral doble $\int_1^2\int_0^3 f(x,\,y)\,dy\,dx$, haremos

```
Integrate[f[x, y], {x, 1, 2}, {y, 0, 3}]
```

$$\frac{1}{6}\left(81-\sqrt{6\pi}\,\text{Sin}[1]\left(\text{Cos}[3]\,\text{FresnelS}\left[3\sqrt{\frac{6}{\pi}}\,\right]-\text{FresnelC}\left[3\sqrt{\frac{6}{\pi}}\,\right]\text{Sin}[3]\right)\right)$$

```
Simplify[%]
```

$$\frac{1}{6}\left(81-\sqrt{6\pi}\,\text{Sin}[1]\left(\text{Cos}[3]\,\text{FresnelS}\left[3\sqrt{\frac{6}{\pi}}\,\right]-\text{FresnelC}\left[3\sqrt{\frac{6}{\pi}}\,\right]\text{Sin}[3]\right)\right)$$

Veamos si con una aproximación numérica obtenemos algo más legible:

```
N[%]

13.8634
```

Este resultado se podía haber obtenido directamente utilizando la integración numérica con la orden `NIntegrate[]`.

Para los casos en los que el recinto de integración no es rectangular como, por ejemplo, $\int_1^2 \int_x^{x^2} x\,y\,dy\,dx$ haremos

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

 27
 8
```

Series

En esta sección vamos a distinguir entre dos problemas bien distintos: calcular la suma de una serie conocida y generar la serie de potencias de una función conocida en un punto dado.

Para calcular la suma de una serie usaremos

Sum[*f*,{*i*,*imax*}]	evalúa $\sum_{i=1}^{imax} f$
Sum[*f*,{i,*imin*,*imax*}]	evalúa $\sum_{i=imin}^{imax} f$
Sum[*f*,{i,*imin*,*imax*,*di*}]	evalúa $\sum_i f$ donde **i = imin, imin+di**, **imin+2di, . . . , imax**.
Sum[f,{i,imin,imax},{j,jmin,jmax}]	evalúa $\sum_{i=imin}^{imax} \sum_{j=jmin}^{jmax} f$

Por ejemplo,

```
Sum[1 / n, {n, 10}]

 7381
 2520
```

```
Sum[a^2 – 3 a + 7, {a, 2, 18, 2}]

933
```

```
Sum[1 / (a + b), {a, 6}, {b, 10}]
```

$$\frac{1946843}{240240}$$

```
N[%]
```

```
8.10374
```

Para generar la serie de potencias de una función dada usamos:

`Series[f,{x,x0,n}]` genera la serie de potencias (desarrollo de Taylor) de **`f`** en el punto **`x=x0`** de orden **`n`** (incluido el resto).

`SeriesCoefficient[`*`serie`*`,n]` coeficiente del n-ésimo término de ***`serie`***.

Calculemos el polinomio de Taylor de orden tres en el punto **`x=0`** de f(x)=sen(x), g(x)=ln(x+1) y h(x)=e^{x^2+1} :

```
Series[Sin[x], {x, 0, 3}]
```

$$x - \frac{x^3}{6} + O[x]^4$$

```
Series[Log[x + 1], {x, 0, 3}]
```

$$x - \frac{x^2}{2} + \frac{x^3}{3} + O[x]^4$$

```
Series[Exp[x^2 + 1], {x, 0, 3}]
```

$$e + e\,x^2 + O[x]^4$$

El coeficiente del término 20 de la serie de potencias de la función $\mathbf{f(x)=\frac{sen(x\text{^}3-2\,x)}{x+2}}$ en el punto **`x=3`** es

```
SeriesCoefficient[Series[Sin[x^3 – 2 x] / (x + 2), {x, 3, 50}], 20];
```

```
N[%]

-2.96413×10^9
```

Ecuaciones diferenciales

Para las ecuaciones o sistemas de ecuaciones diferenciales, el procedimiento es análogo a las algebraicas. Resolvamos la ecuación diferencial de variable separadas **y'=2xy**. En primer lugar podemos (aunque no es necesario) definir la ecuación diferencial a resolver:

```
Eq1 = D[y[x], x] == 2 x y[x]

y'[x] == 2 x y[x]
```

También es posible definir la ecuación anterior como **y'[x]==2 x y**. De nuevo nos aparece un ejemplo donde se distinguen los significados tan distintos que tienen el signo de igualdad y el doble signo de igualdad. Una vez hecho esto, podemos pasar a resolverla como sigue:

```
DSolve[Eq1, y[x], x]

{{y[x] → e^x^2 C[1]}}
```

En caso de que queramos indicarle valores iniciales para obtener una solución particular, se añade como una nueva ecuación entre llaves de la siguiente manera:

```
DSolve[{Eq1, y[0] == 1}, y[x], x]

{{y[x] → e^x^2}}
```

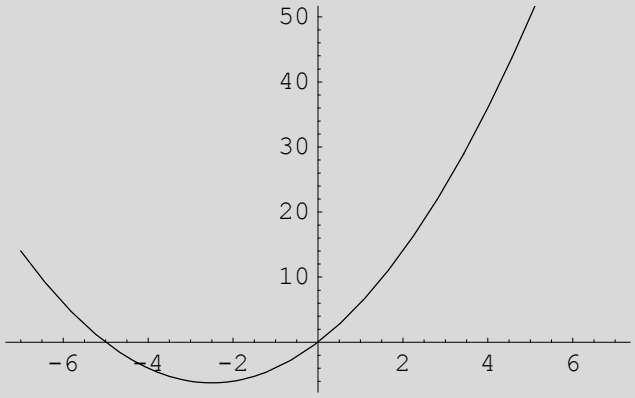
La función **NDSolve[]** ya debemos intuir cuándo podemos y debemos utilizarla para ecuaciones diferenciales ordinarias, así como para poder lograr la resolución de sistemas de ecuaciones diferenciales.

Representación gráfica

Representación gráfica en 2D

Como ya habíamos anticipado, para la representación gráfica de funciones de una variable debemos usar el comando **`Plot[función,{x,xmin,xmax}]`** (también en la paleta *Basic-Calculations/Graphics)* donde **`x`** es la variable independiente y **`xmin`** y **`xmax`** son los extremos del intervalo del dominio de la **`función`** que deseamos representar. Por ejemplo:

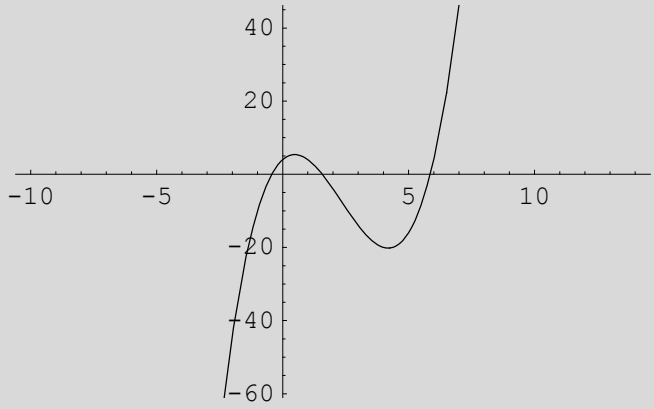
```
Plot[x^2 + 5 x, {x, −7, 7}]
```




```
- Graphics -
```

Representemos gráficamente otra función:

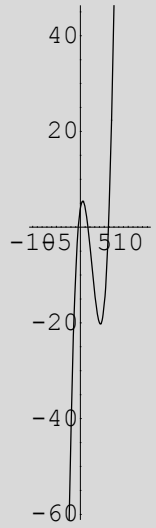
```
g[x_] := x^3 − 7 x^2 + 6 x + 4
Plot[g[x], {x, −10, 14}]
```




```
- Graphics -
```

Observemos que *Mathematica* realiza el gráfico con distintas escalas en ambos ejes. Para que tome la misma unidad en los ejes, se debe agregar la orden **AspectRatio->Automatic** (esta opción se verá con profundidad en la próxima sección) como vemos en el siguiente ejemplo:

```
Plot[g[x], {x, −10, 14}, AspectRatio → Automatic]
```



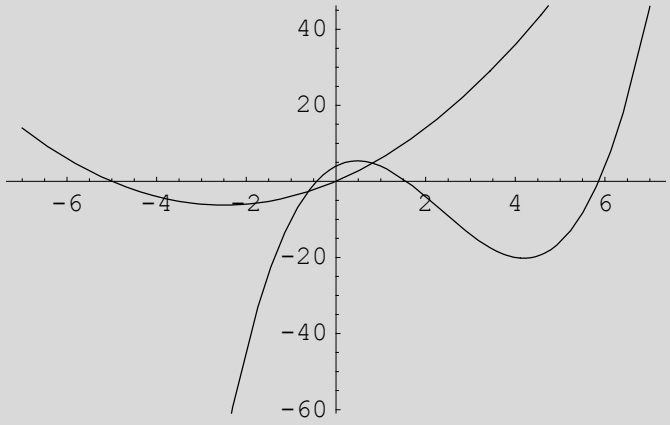

```
- Graphics -
```

Como vemos, la gráfica de la función ahora es "más real", aunque hay detalles que no se aprecian con suficiente nitidez. Por tanto, es importante decidir cuándo usar la orden **AspectRatio->Automatic** y cuándo dejar que *Mathematica* "decida" cuál es la escala más conveniente.

Observación: Después de cada gráfico, *Mathematica* imprime **-Graphics-**. Para evitar que se produzca este mensaje, se puede agregar al final de la línea del comando **Plot** un punto y coma.

Podemos pedir que represente ambas funciones en el mismo dibujo; para ello, usamos el comando **Plot**, pero como primer argumento usaremos ahora una lista de funciones:

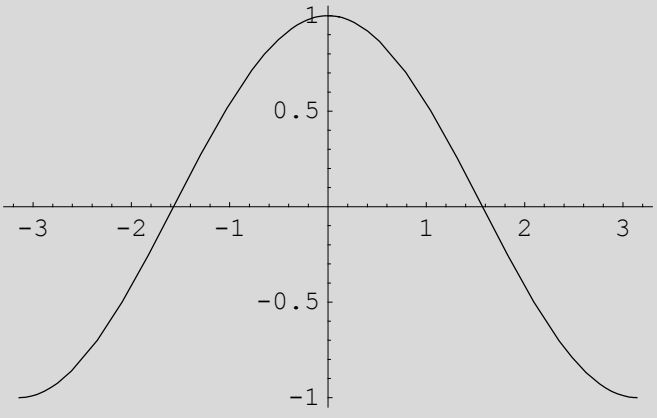
```
Plot[{x^2 + 5 x, g[x]}, {x, −7, 7}]
```



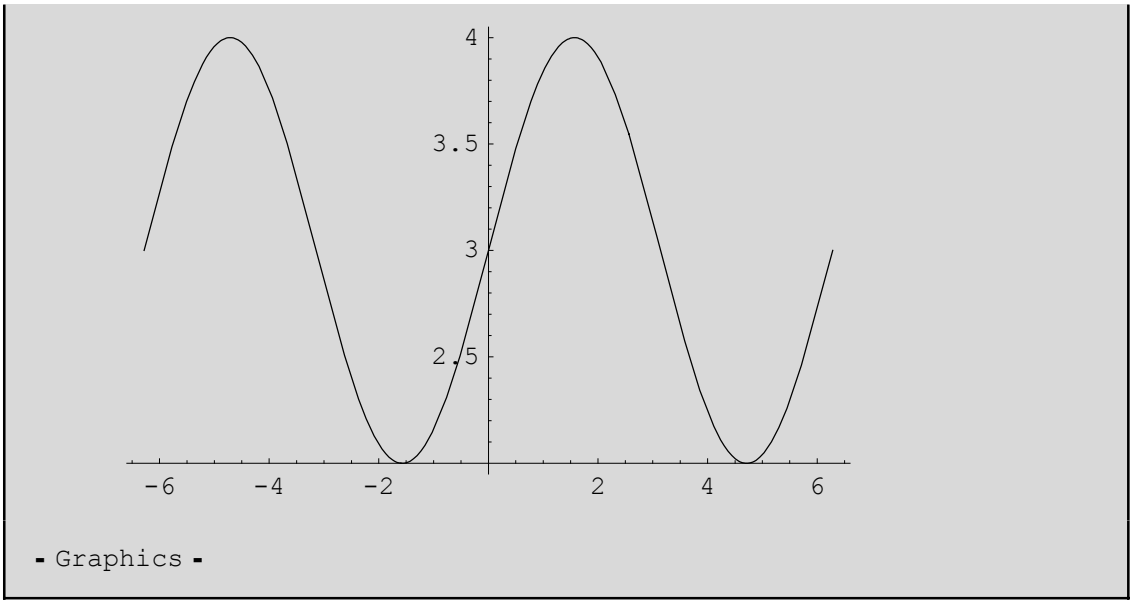
- Graphics -

Existe otra manera de superponer gráficas; consiste en asignarle a cada una un nombre y usar el comando **Show[]** como se muestra en el siguiente ejemplo:

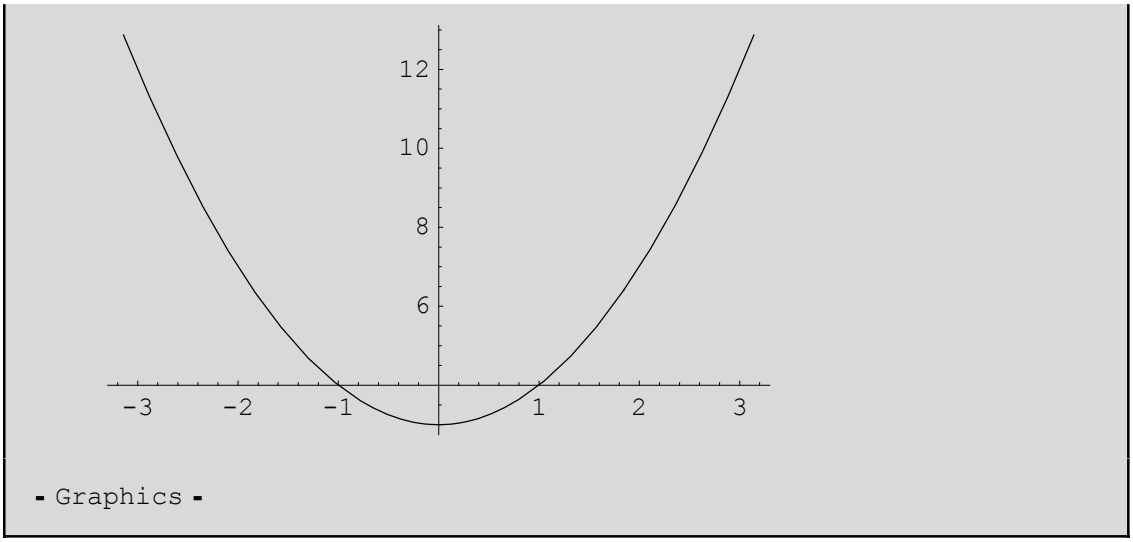
```
aa = Plot[Cos[x], {x, −Pi, Pi}]
bb = Plot[Sin[x] + 3, {x, −2 Pi, 2 Pi}]
cc = Plot[x^2 + 3, {x, −Pi, Pi}]
dd = Show[aa, bb, cc]
```



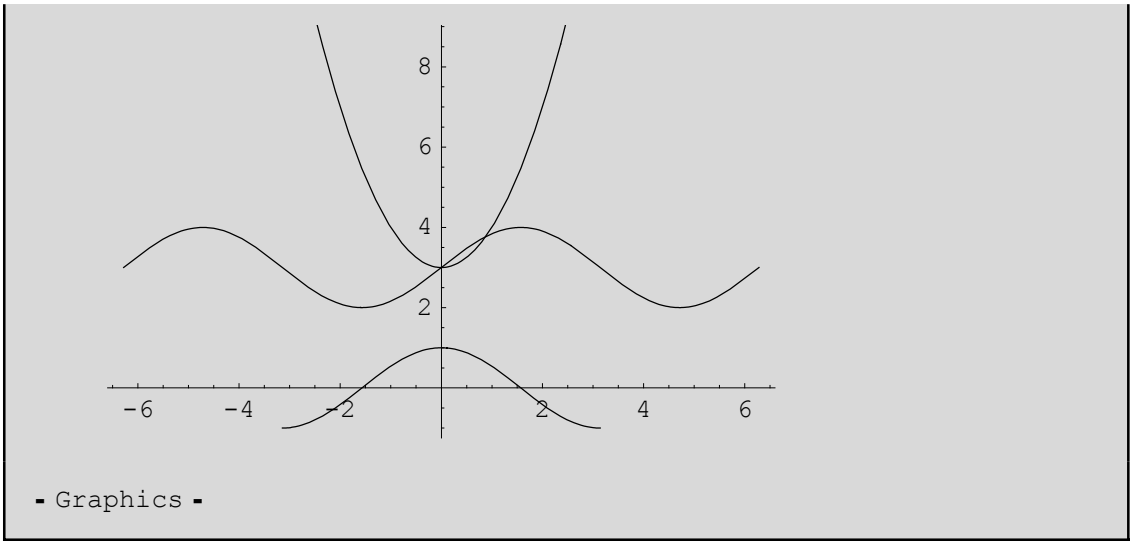
- Graphics -



- Graphics -



- Graphics -



- Graphics -

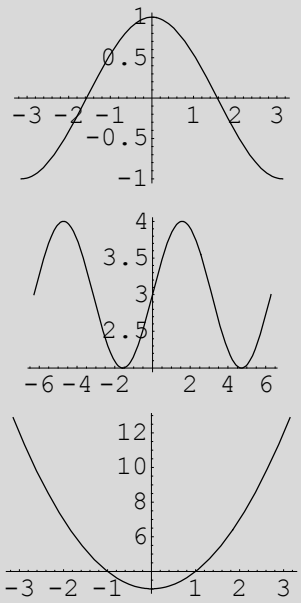
El comando **`Show[]`** permite mostrar los gráficos en diferentes disposiciones mediante el comando **`GraphicsArray[]`**. Así, si queremos los tres gráficos en fila:

```
Show[GraphicsArray[{aa, bb, cc}]]
```


```
- GraphicsArray -
```

Si los queremos en columna:

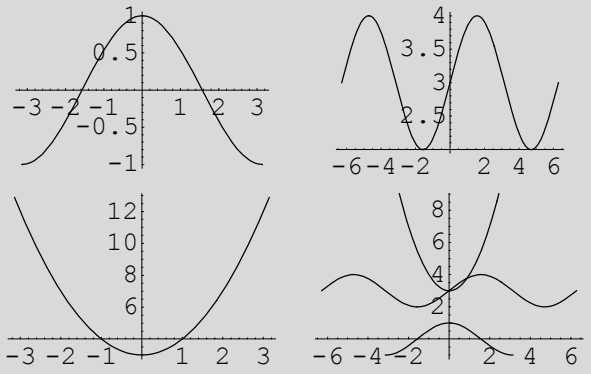
```
Show[GraphicsArray[{{aa}, {bb}, {cc}}]]
```




```
- GraphicsArray -
```

Si queremos los cuatro gráficos en dos columnas:

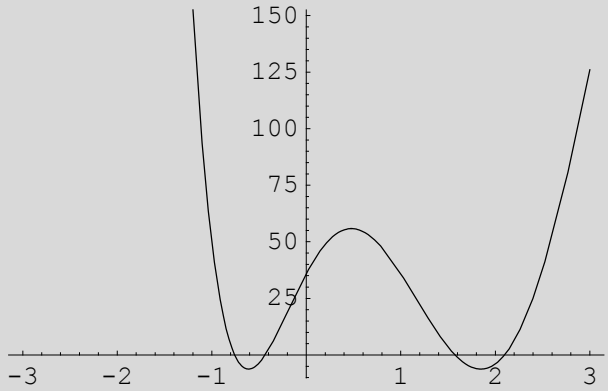
```
Show[GraphicsArray[{{aa, bb}, {cc, dd}}]]
```



- GraphicsArray -

También es muy sencillo componer funciones:

```
f[x_] := x^2 + 5 x
h[x_] := f[g[x]]
Plot[h[x], {x, −3, 3}]
```



- Graphics -

Gráficos en coordenadas polares

Para representar funciones en coordenadas polares utilizamos la orden **`PolarPlot[x[t],{t,tmin,tmax}]`** donde **`x[t]`** es el radio y **`t`** es el ángulo. Para ello debemos cargar un paquete de gráficos. Esta vez lo cargaremos mediante el comando Needs:

```
Needs["Graphics`Graphics`"]
```

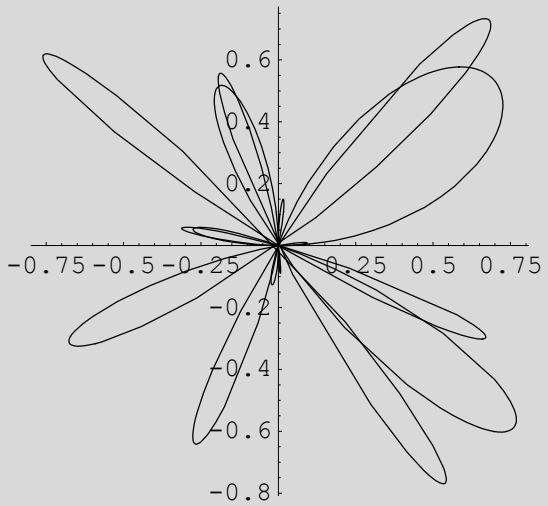
Recordemos que también podemos cargarlo de esta forma:

```
<< Graphics`Graphics`
```

Es importante tener en cuenta que es conveniente cargarlo antes de pedir un gráfico en polares, ya que si no se produce un error.

La diferencia entre las dos formas de cargarlo consiste en que si lo hacemos mediante el comando **`Needs[]`** no vuelve a cargarlo si el paquete ya ha sido cargado anteriormente, con el consiguiente ahorro de memoria en la máquina. Representemos la función g(t) = sen(2t) en coordenadas polares.

```
PolarPlot[Sin[2 t] * Cos[t^2], {t, 0, 2 Pi}]
```

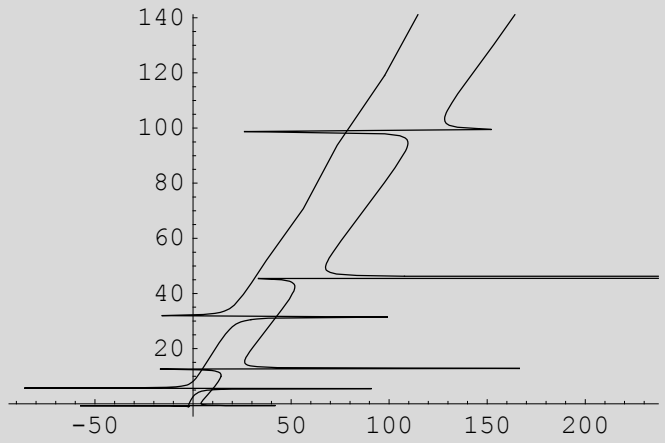



```
- Graphics -
```

Gráficos de funciones expresadas en forma paramétrica

En este caso no hay que cargar previamente paquete alguno. El comando es **`ParametricPlot[{x[t],y[t]},{t,tmin,tmax}]`** donde **`x[t]`** e **`y[t]`** son las expresiones de **`x`** e **`y`** en función del parámetro **`t`**. Por ejemplo, representemos la función **`(x(t),y(t))=(t^2-2,t^2+2t)`** donde el parámetro **`t`** varía en el intervalo **`(-8,8)`**:

```
ParametricPlot[{t^2 – 2 + Tan[t], t^2 + 2 t}, {t, –15, 15}]
```



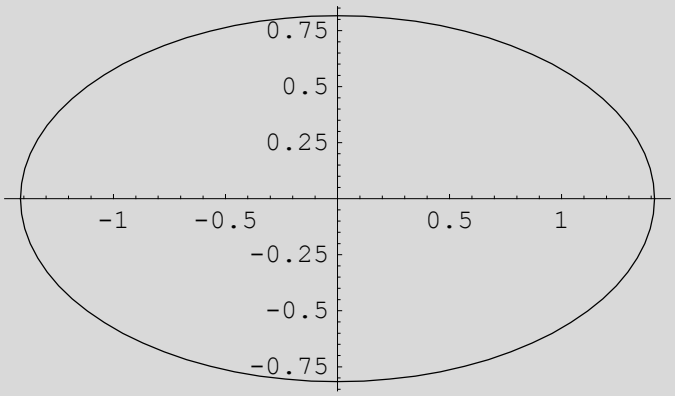

```
- Graphics -
```

Gráficos de funciones implícitas

En esta ocasión sí es necesario cargar previamente el paquete correspondiente:

```
Needs["Graphics`ImplicitPlot`"]
```

```
ImplicitPlot[x^2 + 3 y^2 – 2 == 0, {x, –2, 2}]
```



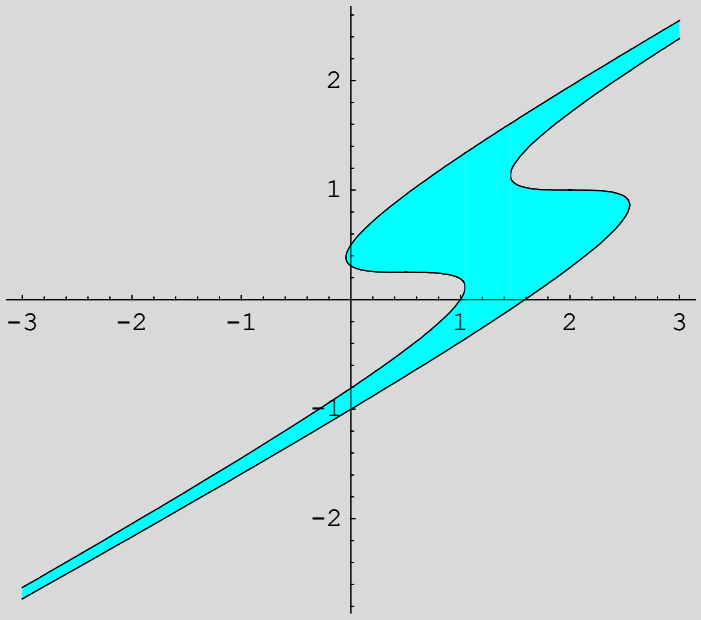

```
- Graphics -
```

Nota: obsérvese que el primer argumento de **`ImplicitPlot[]`** es una ecuación. Es por esto por lo que hemos debido de añadir el doble signo de igualdad en la ecuación.

Gráficas de inecuaciones

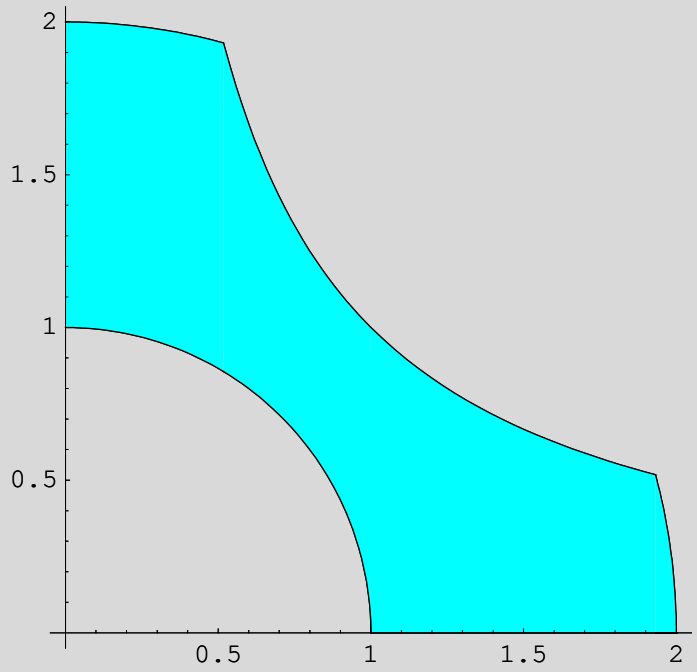
Para estudiar optimización con restricciones necesitaremos saber, por ejemplo, cómo se dibujan regiones en el plano. Estas regiones vienen definidas por inecuaciones y para su representación gráfica utilizaremos el comando `InequalityPlot[inecuaciones, {x,xmin,xmax},{y,ymin,ymax}]`. Esta función se encuentra en el paquete **`Graphics`InequalityGraphics`** que debemos cargar previamente:

```
<< Graphics`InequalityGraphics`
InequalityPlot[ 1 ≤ (x – 2 y)^3 + 4 y ≤ 4, {x, –3, 3}, {y, –3, 3}]
```




```
- Graphics -
```

```
InequalityPlot[ 1 < x^2 + y^2 < 4 ∧ x y < 1 ∧ x > 0 ∧ y > 0, {x, −2, 3}, {y, −2, 2} ];
```



Gráficas de Listas de Datos

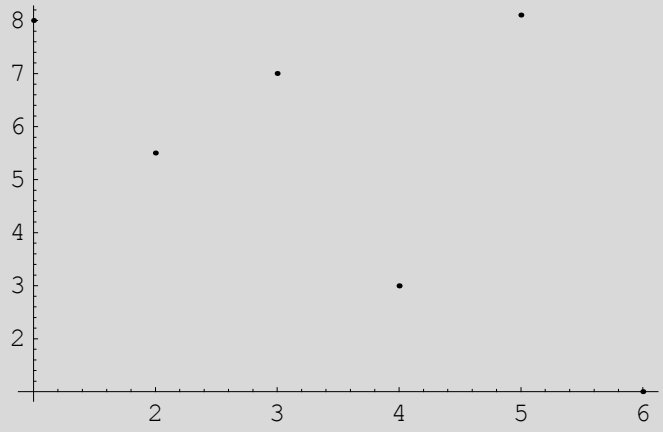
En muchas ocasiones es necesario mostrar gráficamente datos obtenidos, por ejemplo, de una experiencia o experimento. Las coordenadas obtenidas pueden no responder a una expresión analítica, pero puede interesarnos "ver" cómo se disponen esos puntos en una gráfica (también denominada *nube de puntos*). Para ello, veremos cómo generar gráficas a partir de listas de datos correspondientes a dos variables cuantitativas. Introducimos una lista de datos de la siguiente forma:

```
tabla = {{1, 8}, {2, 5.5}, {3, 7}, {4, 3}, {5, 8.1}, {6, 1}}
```

```
{{1, 8}, {2, 5.5}, {3, 7}, {4, 3}, {5, 8.1}, {6, 1}}
```

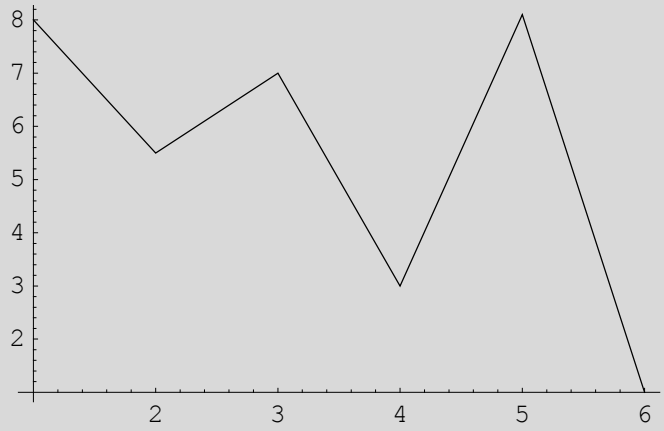
Para mostrar estos puntos gráficamente, utilizamos el comando **`ListPlot`**:

```
ListPlot[tabla];
```



Si queremos mostrar los puntos unidos por segmentos, agregamos la siguiente opción:

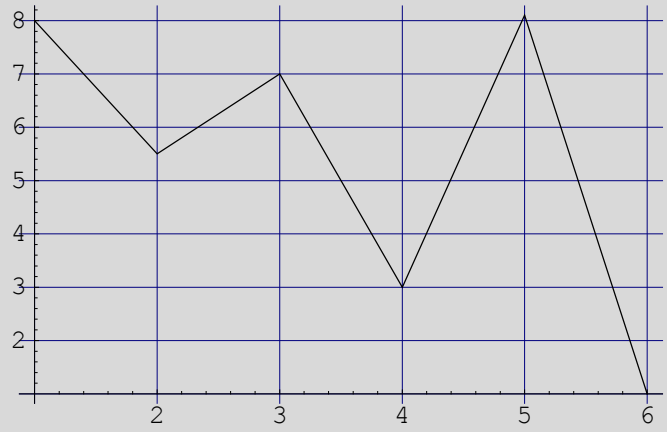
```
ListPlot[tabla, PlotJoined → True]
```




```
- Graphics -
```

Puede agregarse una malla:

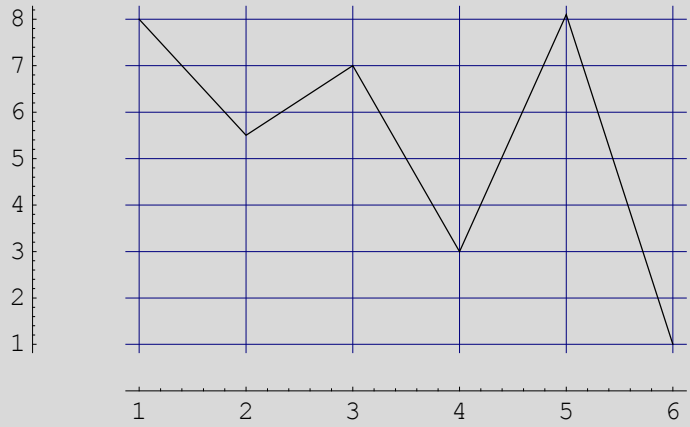
```
ListPlot[tabla, PlotJoined → True, GridLines → Automatic]
```




```
- Graphics -
```

Observación: el programa acomoda los ejes coordenados de forma que los datos sean fácilmente visibles, pero esto puede no responder a la posición real de esos puntos. Para obtener una visualización más fiel usamos el comando **`AxesOrigin->{0,0}`**, que ubica los ejes en su posición normal:

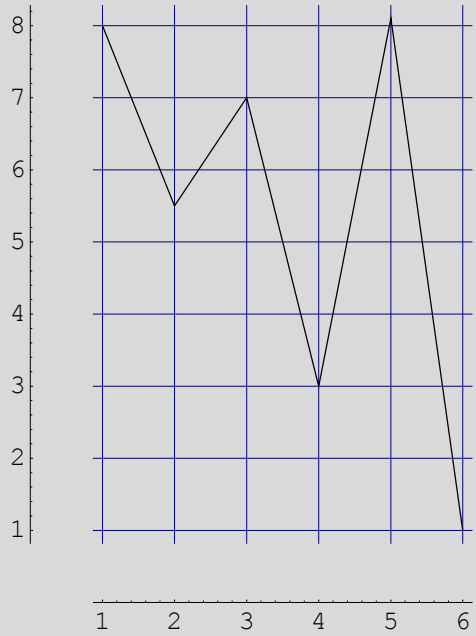
```
ListPlot[tabla, PlotJoined → True, GridLines → Automatic, AxesOrigin → {0, 0}]
```




```
- Graphics -
```

También puede pedirse la misma escala en ambos ejes mediante la opción **`AspectRatio->Automatic`**:

```
ListPlot[tabla, PlotJoined → True, GridLines → Automatic,
  AxesOrigin → {0, 0}, AspectRatio –> Automatic];
```



Esta gráfica puede servir de ayuda en el caso en que deseemos interpolar y queramos aproximar las coordenadas de un punto intermedio. Para ello, marquemos el gráfico (un click encima del gráfico) y, posicionando el cursor sobre él, presionemos la tecla *Ctrl*. Así *Mathematica* mostrará en la parte inferior izquierda de la pantalla las coordenadas aproximadas del punto que señalemos en la gráfica. Es importante tener en cuenta que es necesario, para que las coordenadas obtenidas sean correctas, que el origen de coordenadas esté en (0,0) (esta observación también es válida para el resto de representaciones gráficas en 2D).

Opciones gráficas

Cuando *Mathematica* realiza un gráfico, tiene que elegir cómo realizar la representación gráfica (escalas, ejes, origen, color, grosor de línea, etc). En muchas de las ocasiones, *Mathematica* probablemente hará una buena elección. No obstante, si se desea cambiar algunas de estas opciones se pueden utilizar las órdenes que veremos en esta sección.

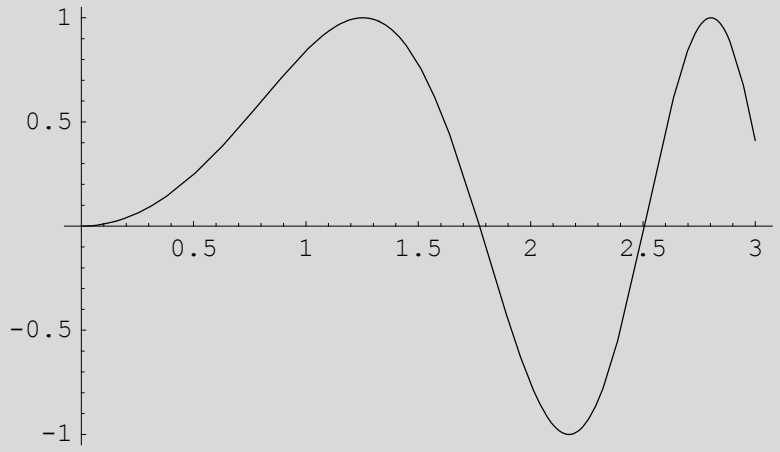
Como último argumento del comando, por ejemplo **`Plot[]`**, se puede incluir una secuencia de órdenes del tipo *nombre–>valor*, para especificar el valor de varias de las opciones gráficas. A cada opción que no asignemos un valor explícito, *Mathematica* le asignará un valor por defecto.

Una orden como **`Plot[]`** tiene muchas opciones que podemos modificar. Algunas de las que veremos son las siguientes:

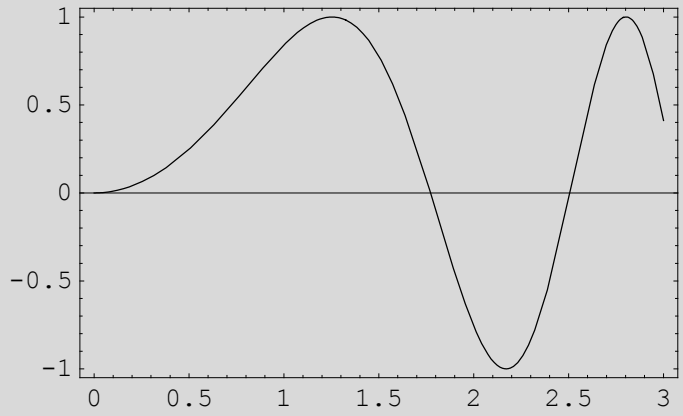
Opción	Valor por defecto	Descripción
AspectRatio	1/GoldenRatio	Proporción alto-ancho; el valor *Automatic* asigna la misma unidad de medida en ambos ejes.
Axes	Automatic	Para incluir los ejes coordenados.
AxesLabel	None	Incluye etiquetas en los ejes; el valor *ylabel* especifica una etiqueta para el eje *oy,* **y** {xlabel,ylabel} para ambos ejes.
AxesOrigin	Automatic	Punto de corte de los ejes (origen).
TextStyle	$TextStyle	Estilo para el texto de la gráfica.
FormatType	StandardForm	Formato usado por defecto para el texto de la gráfica.
Frame	False	Enmarca o no el gráfico.
FrameLabel	None	Etiquetas alrededor del marco; una lista de como máximo 4 elementos etiqueta en el sentido de las agujas del reloj empezando por el eje x.
FrameTicks	Automatic	Muestra una regla en cada lado del marco; el valor *None* no muestra las reglas.
GridLines	None	Muestra una malla para cada valor de la regla del marco; el valor *Automatic* incluye el las líneas para la mayoría de las marcas.
PlotLabel	None	Etiqueta para el gráfico.
PlotRange	Automatic	Especifica la parte del recorrido que se desea mostrar; el valor *All* incluye todos los puntos (si es posible).
Ticks	Automatic	Si existen los ejes, los muestra reglados; el valor *None* no muestra las marcas.
ImageSize	Automatic	Un valor especifica el ancho del gráfico; {n,m} especifica el ancho y el alto.
Background	Automatic	Opciones para el fondo del gráfico (GrayLevel, Hue, RGBColor...).

Un ejemplo con todos los valores por defecto es:

```
Plot[Sin[x^2], {x, 0, 3}];
```

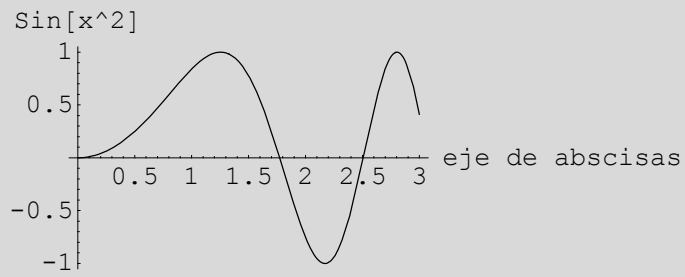



```
Plot[Sin[x^2], {x, 0, 3}, Frame → True];
```



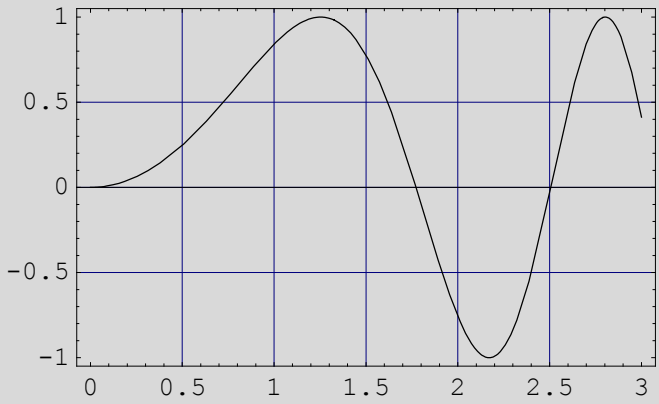
A continuación especificamos unas etiquetas para cada uno de los ejes. Las etiquetas aparecerán tal como aparecerían en una salida de *Mathematica.* Las etiquetas pueden incluir texto escribiéndolo entre comillas.

```
Plot[Sin[x^2], {x, 0, 3}, AxesLabel → {"eje de abscisas", "Sin[x^2]"}];
```



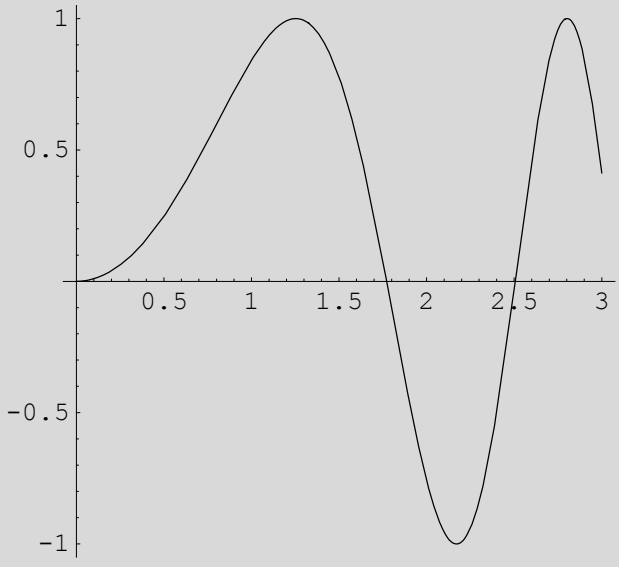
Para la malla:

```
Plot[Sin[x^2], {x, 0, 3}, Frame → True, GridLines → Automatic];
```



Con la opción **AspectRatio** se puede cambiar la forma completa de la gráfica. Como dijimos antes, **AspectRatio** mide la razón o proporción del ancho con respecto a la altura. El valor por defecto es el inverso del número de oro (Golden Ratio) que se supone la mejor forma para la estética de un rectángulo. El número de oro vale $(1+\sqrt{5})/2 \approx 1.61803$.

```
Plot[Sin[x^2], {x, 0, 3}, AspectRatio → 1];
```



Algunos de los valores más usuales para varias de las opciones vistas hasta ahora son

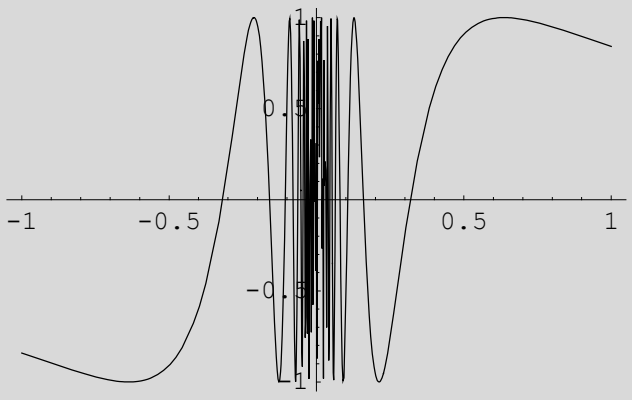
`Automatic`	utiliza los valores por defecto para los algoritmos internos.
`None`	no incluye la opción.
`All`	lo incluye todo.
`True`	realiza la opción.
`False`	no realiza la opción.

Cuando *Mathematica* realiza un gráfico, trata de fijar unas escalas para *x* e *y* con la idea de incluir solo las partes más interesantes de la gráfica. Si la función crece rápidamente, o tiene singularidades, las partes de la gráfica demasiado largas las corta. Con la orden `PlotRange` podemos controlar el rango de variación para la *x* y la *y*. Los posibles valores para esta opción son:

`Automatic`	muestra una gran parte de la gráfica incluyendo la parte " más interesante " que aparecería por defecto.
`All`	muestra todos los puntos (si es posible).
{*ymin, ymax*}	muestra un rango específico para los valores de *y*.
{*xrange, yrange*}	muestra un rango específico para x e y.

Mathematica siempre trata de representar curvas "suaves". Así, en lugares donde la función se "mueve" u "oscila" mucho, *Mathematica* usará más puntos para la representación gráfica. La función **`sen(1/x)`** se "retuerce" indefinidamente alrededor del punto **`x=0`**. *Mathematica* tomará más puntos en aquella región donde la función se comporta de esta manera, pero nunca podrá evaluar los infinitos valores necesarios para la representación exacta. Como resultado de esto puede haber pequeños saltos o incorrecciones en la gráfica.

```
Plot[Sin[1 / x], {x, −1, 1}];
```



Otras opciones válidas también para la orden **`Plot[]`** son las siguientes:

Opción	Valor por defecto	
`PlotStyle`	`Automatic`	Estilo de la línea o puntos de la gráfica.
`PlotPoints`	`25`	Número mínimo de puntos para ser evaluados y realizar la gráfica.
`MaxBend`	`10`	Ángulo máximo del pliegue entre sucesivos segmentos de la curva.
`PlotDivision`	`30`	Valor máximo en el que subdividir la representación de la gráfica.

Si incrementamos el número de puntos con **`PlotPoints`**, podemos hacer que *Mathematica* evalúe la función en un número mayor de puntos. Pero parece lógico pensar que, mientras mayor sea el número de puntos, mayor será el tiempo que necesite *Mathematica* para representarla, aunque a su vez la curva será más suave.

Podemos obtener información de todas las opciones (con sus correspondientes valores por defecto) de las que dispone una función, como por ejemplo **`Plot[]`**, mediante la orden **`Options[]`**:

```
Options[Plot]

{AspectRatio → 1/GoldenRatio, Axes → Automatic,
 AxesLabel → None, AxesOrigin → Automatic, AxesStyle → Automatic,
 Background → Automatic, ColorOutput → Automatic, Compiled → True,
 DefaultColor → Automatic, DefaultFont :→ $DefaultFont,
 DisplayFunction :→ $DisplayFunction, Epilog → {},
 FormatType :→ $FormatType, 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, TextStyle :→ $TextStyle, Ticks → Automatic}
```

Si queremos saber el valor por defecto de una sola de las opciones haremos algo parecido:

```
Options[Plot, AspectRatio]

{AspectRatio → 1/GoldenRatio}
```

Otras opciones gráficas: color, grosor, brillo, fondo...

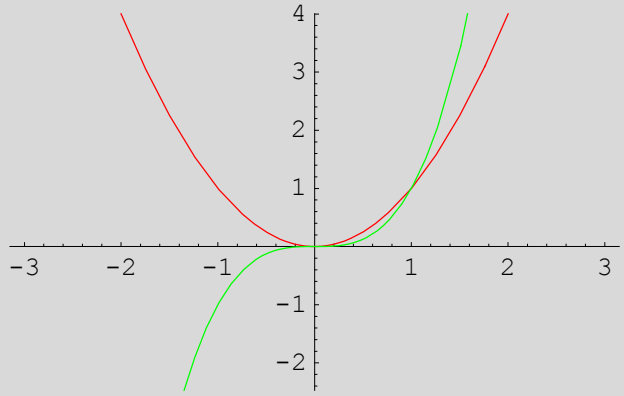
Cuando se crea un objeto gráfico en *Mathematica*, normalmente se añade una lista de opciones gráficas. En esta lista se puede añadir directivas gráficas que especifican cómo se deben reproducir los elementos del gráfico.

Mathematica dispone de varias opciones gráficas. Una de las más importantes es aquélla que especifica el color de los elementos del gráfico. Incluso si el gráfico es en blanco y negro, se puede especificar la intensidad de color dentro de la escala de los grises. También se puede utilizar en gráficos a color las intensidades del gris con la opción **`ColorOutput -> GrayLevel`**.

`GrayLevel[ i ]`	intensidad del gris entre 0 (negro) *y* 1 (blanco).
`RGBColor[ r, g, b ]`	color mezclando rojo (r), verde (g) y azul (b), todos entre 0 y 1.
`Hue[ h ]`	color con tono *h* entre 0 y 1.
`Hue[ h, s, b ]`	color especificando el tono, saturación y brillo, entre 0 y 1.

A continuación se muestran dos curvas a color:

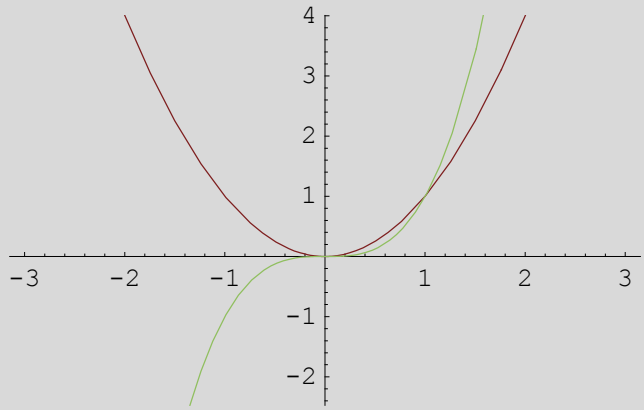
```
Plot[{x^2, x^3}, {x, −3, 3}, PlotStyle → {{RGBColor[1, 0, 0]}, {RGBColor[0, 1, 0]}}]
```




```
- Graphics -
```

La opción **`Hue[h]`** nos proporciona otra manera de especificar el color usando un único parámetro. Cuando **`h`** varía de 0 a 1, **`Hue[h]`** se mueve cíclicamente entre el rojo, amarillo, verde, cian, azul, magenta, negro y rojo de nuevo. **`Hue[h,s,b]`** nos permite especificar no solo el tono del color sino la "saturación" o contraste y el brillo. Tomando la saturación igual a 1 tenemos el color más profundo; haciendo decrecer este valor a 0 obtenemos progresivamente un color más "claro".

```
Plot[{x^2, x^3}, {x, −3, 3}, PlotStyle → {{Hue[1, 0.75, 0.5]}, {Hue[0.25, 0.5, 0.75]}}]
```




```
- Graphics -
```

Cuando especificamos una opción gráfica como **`RGBColor[]`**, ésta afecta a todos los elementos gráficos que aparecen en una lista. *Mathematica* también dispone de opciones gráficas que son específicas a algunos tipos de gráficos:

La opción **`PointSize[d]`** sirve para que todos los puntos del gráfico los represente como un círculo de diametro ***d***. Con **`PointSize`**, el diametro **`d`** está medido en proporción al ancho de todo el gráfico.

Mathematica también permite la opción **`AbsolutePointSize[d]`**, que especifica el diametro "absoluto" de los puntos, medidos en unidades fijas.

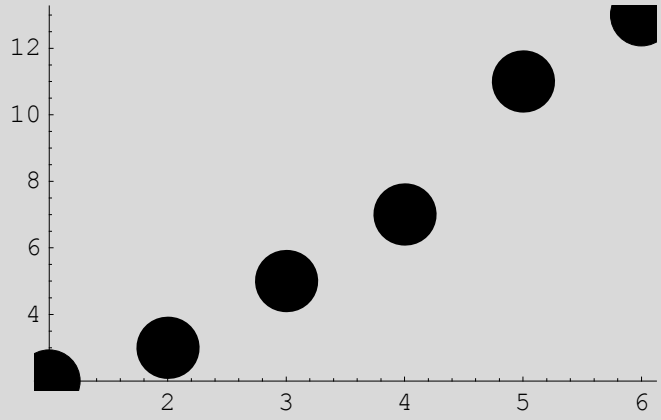
`PointSize[` *d* `]`	todos los puntos con diámetro *d* en proporción al ancho del gráfico completo.
`AbsolutePointSize[` *d* `]`	todos los puntos con diámetro *d* en unidades absolutas.

Aquí vemos cómo construir una lista con seis puntos de dos coordenadas:

```
Table[{n, Prime[n]}, {n, 6}]

{{1, 2}, {2, 3}, {3, 5}, {4, 7}, {5, 11}, {6, 13}}
```

Mostramos cada punto con diámetro igual a un 10% del ancho del gráfico.

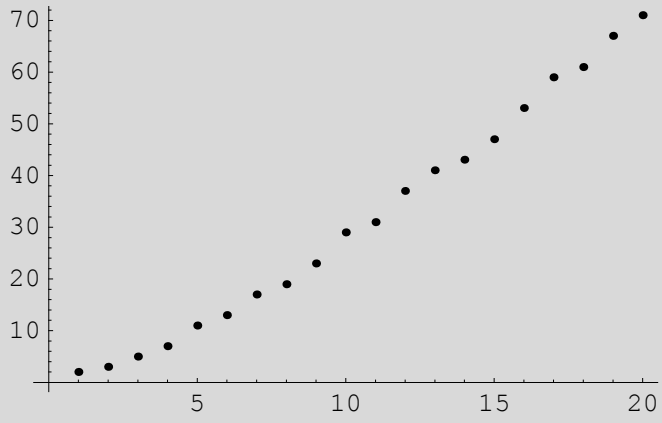
```
ListPlot[%, PlotStyle → PointSize[0.1]]
```




```
- Graphics -
```

En el siguiente gráfico, cada punto tiene un tamaño de 3 "unidades absolutas":

```
ListPlot[Table[Prime[n], {n, 20}], Prolog → AbsolutePointSize[3]]
```



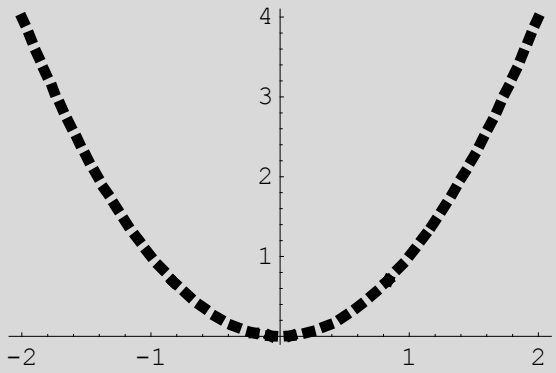

```
- Graphics -
```

Algunas de las opciones gráficas para la orden **`Line[]`** (**`Line[{punto1,punto2}]`** une con un segmento rectilíneo ambos puntos) son:

`Thickness[ w ]`	todas las líneas con grosor *w*, en proporción al ancho del gráfico.
`AbsoluteThickness[ w ]`	todas las líneas con grosor *w* en unidades absolutas
`Dashing[ { w1, w2, ... } ]`	muestra todas las líneas como una sucesión de segmentos discontinuos, con medidas *w1*, *w2*, ...
`AbsoluteDashing[ { w1, w2, ... } ]`	usa unidades absolutas para la longitud de los segmentos discontinuos.

Veamos un ejemplo:

```
Plot[x^2, {x, -2, 2}, PlotStyle → {Thickness[.02], Dashing[{.005, .03}]}]
```




```
- Graphics -
```

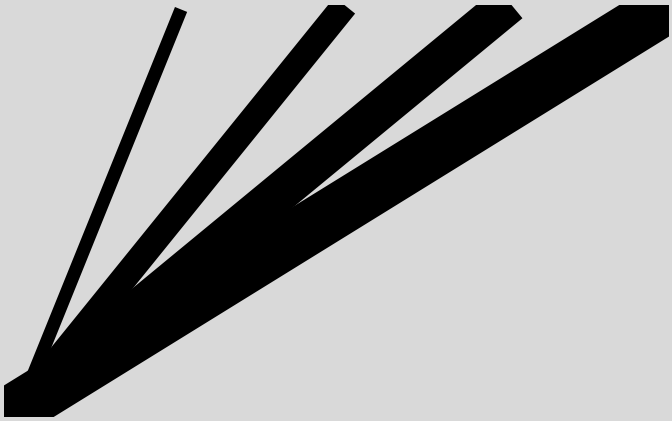
Lo siguiente genera (pero no representa) una lista de líneas de diferentes grosores:

```
Table[{Thickness[n / 50], Line[{{0, 0}, {n, 1}}]}, {n, 4}]
```

$$\left\{\left\{\text{Thickness}\left[\frac{1}{50}\right], \text{Line}[\{\{0, 0\}, \{1, 1\}\}]\right\},\right.$$
$$\left\{\text{Thickness}\left[\frac{1}{25}\right], \text{Line}[\{\{0, 0\}, \{2, 1\}\}]\right\},$$
$$\left\{\text{Thickness}\left[\frac{3}{50}\right], \text{Line}[\{\{0, 0\}, \{3, 1\}\}]\right\},$$
$$\left.\left\{\text{Thickness}\left[\frac{2}{25}\right], \text{Line}[\{\{0, 0\}, \{4, 1\}\}]\right\}\right\}$$

Y ahora las mostramos:

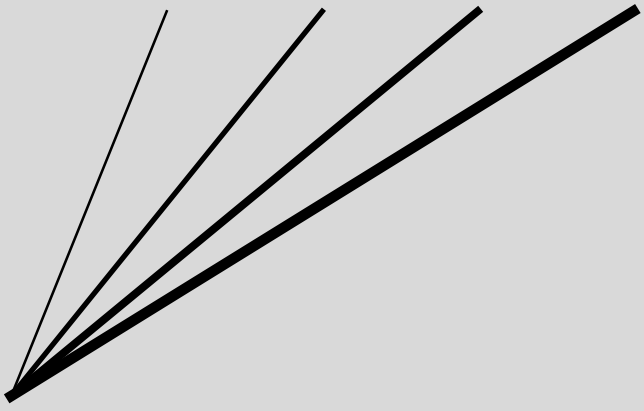
```
Show[Graphics[%]]
```



```
- Graphics -
```

```
Table[{AbsoluteThickness[n], Line[{{0, 0}, {n, 1}}]}, {n, 4}]

{{AbsoluteThickness[1], Line[{{0, 0}, {1, 1}}]},
 {AbsoluteThickness[2], Line[{{0, 0}, {2, 1}}]},
 {AbsoluteThickness[3], Line[{{0, 0}, {3, 1}}]},
 {AbsoluteThickness[4], Line[{{0, 0}, {4, 1}}]}}
```

```
Show[Graphics[%]]
```

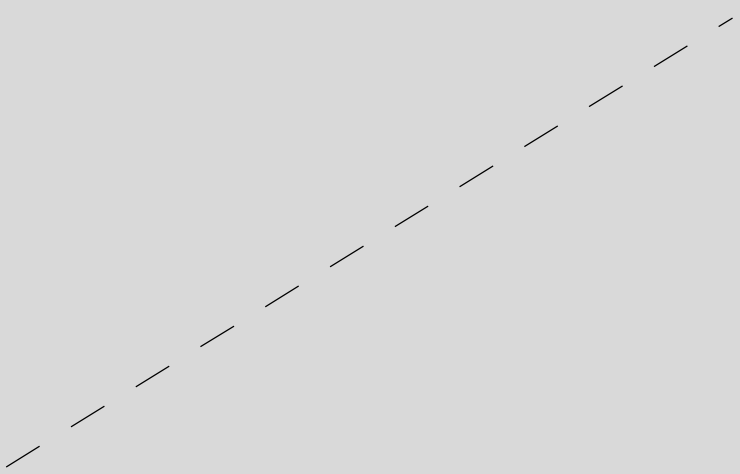


```
- Graphics -
```

La orden `Dashing` nos permite crear líneas con distintos tipos de segmentos. La idea es romper la línea en segmentos que son dibujados y omitidos alternativamente. Cambiando la medida de los segmentos se obtienen líneas de diferentes estilos. `Dashing` nos permite tam-

bién especificar una sucesión de medidas para los elementos. Esta sucesión se repite tantas veces como sea necesaria hasta representar la línea completa.

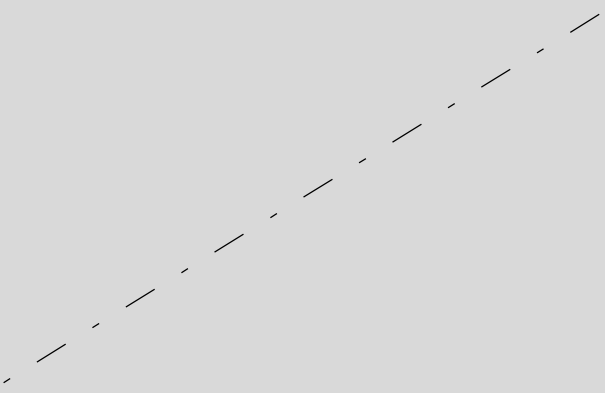
```
Show[Graphics[{Dashing[{0.05, 0.05}], Line[{{-1, -1}, {1, 1}}]}]]
```



```
- Graphics -
```

Ahora con líneas y puntos:

```
Show[Graphics[{Dashing[{0.01, 0.05, 0.05, 0.05}], Line[{{-1, -1}, {1, 1}}]}]]
```



```
- Graphics -
```

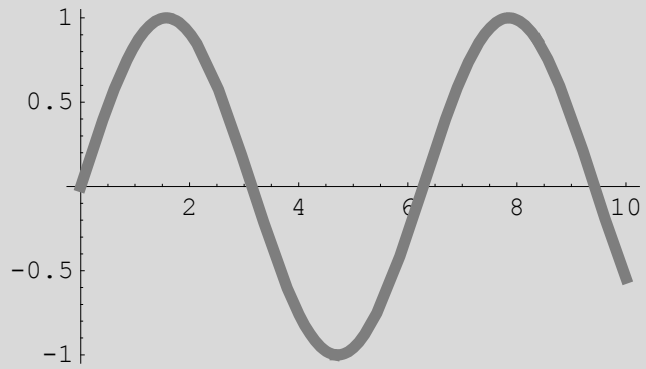
Otra manera de usar las opciones gráficas es insertarlas directamente en una lista. No obstante, algunas veces queremos que las opciones se apliquen de manera más global y, por ejemplo, determinar el estilo global en toda la sesión de trabajo independientemente de cómo sean algunas opciones gráficas particulares.

Algunas opciones para especificar el estilo de forma general son:

`PlotStyle -> style`	especifica un estilo para ser usado en todas las curvas dibujadas con Plot.
`PlotStyle -> {{style1}, {style2}, ...}`	especifica estilos para ser usados cíclicamente para una sucesión de curvas en Plot.
`MeshStyle -> style`	especifica un estilo para ser usado en una malla en gráficos de superficie y densidad.
`BoxStyle -> style`	especifica un estilo para ser usado por la caja en gráficos 3 D, según veremos.

Veamos un ejemplo en el cual la curva está representada en un estilo específico:

```
Plot[Sin[x], {x, 0, 10}, PlotStyle → {{Thickness[0.02], GrayLevel[0.5]}}]
```



- Graphics -

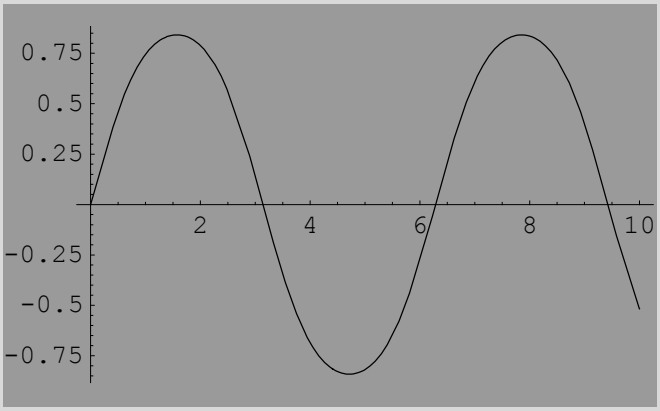
Y algunos estilos son:

`GrayLevel[0.5]`	gris
`RGBColor[1, 0, 0]`, etc.	rojo, etc.
`Thickness[0.05]`	grueso
`Dashing[{0.05, 0.05}]`	discontinuo
`Dashing[{0.01, 0.05, 0.05, 0.05}]`	punto−segmento

Mathematica también permite modificar otros aspectos del gráfico:

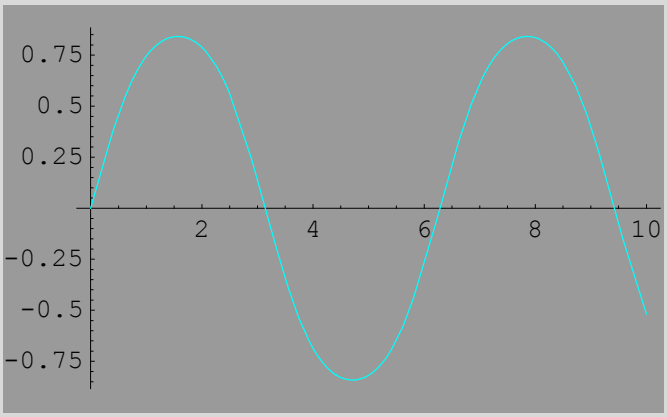
`Background -> color`	especifica el color del fondo
`DefaultColor -> color`	especifica el color por defecto del gráfico
`Prolog -> g`	proporciona un gráfico antes de que Plot haya finalizado
`Epilog -> g`	proporciona gráficos después de que Plot haya finalizado

```
Plot[Sin[Sin[x]], {x, 0, 10}, Background → GrayLevel[0.6]]
```



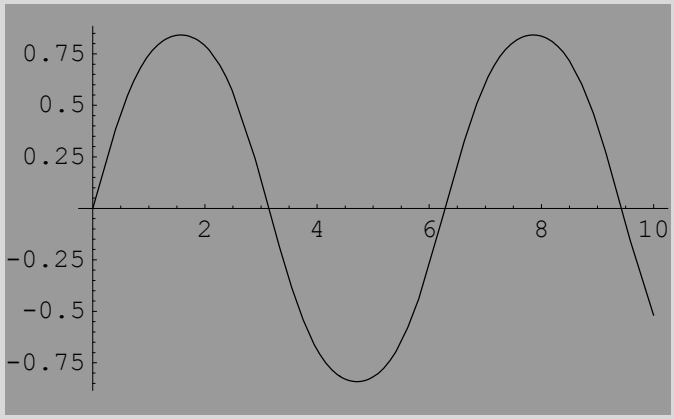

```
- Graphics -
```

```
Plot[Sin[Sin[x]], {x, 0, 10}, Background → GrayLevel[0.6],  Prolog → Hue[.5]]
```




```
- Graphics -
```

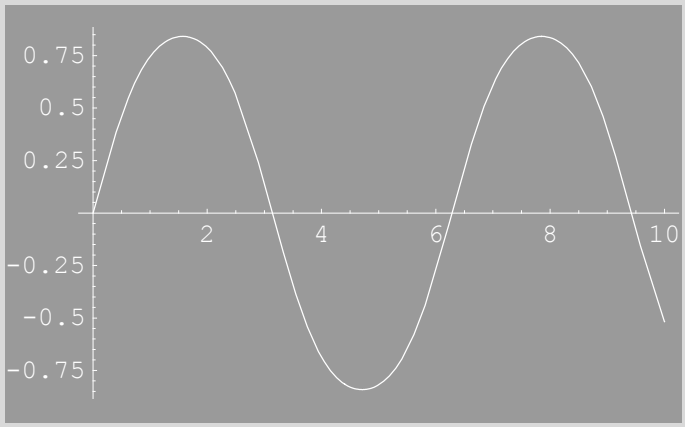
```
Plot[Sin[Sin[x]], {x, 0, 10}, Background → GrayLevel[0.6],  Epilog → Dashing[{0.05, 0.05}]]
```




```
- Graphics -
```

Cambiamos el color por defecto a blanco:

```
Show[%, DefaultColor → GrayLevel[1]]
```

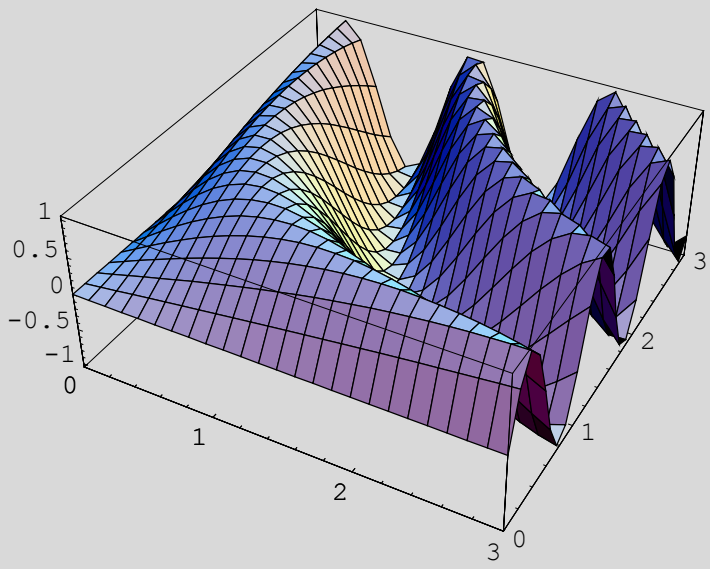



```
- Graphics -
```

Representación gráfica en 3D

La orden **`Plot3D[f,{x,xmin,xmax},{y,ymin,ymax}]`** realiza representaciones gráficas en tres dimensiones.

```
Plot3D[Sin[2 x y], {x, 0, 3}, {y, 0, 3}]
```




```
- SurfaceGraphics -
```

Mathematica dispone de una gran cantidad de opciones gráficas para gráficos en 3D. Al igual que para **`Plot[]`**, podemos obtener información de todas las opciones disponibles con **`Options[]`**:

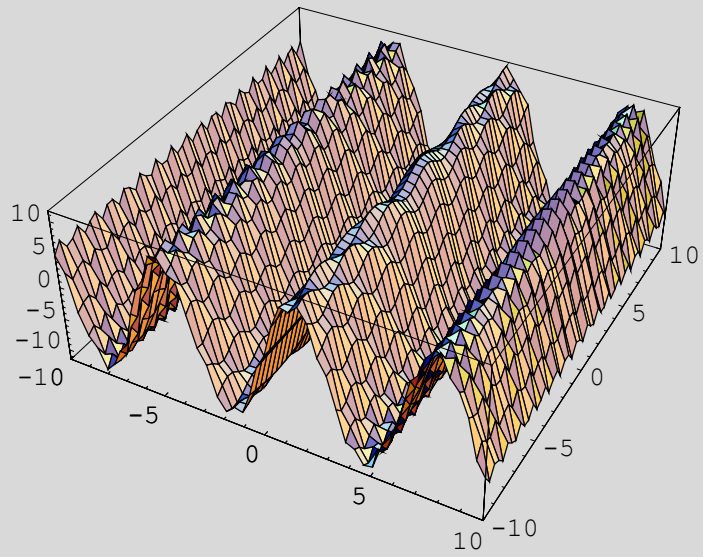
```
Options[Plot3D]

{AmbientLight → GrayLevel[0], AspectRatio → Automatic,
 Axes → True, AxesEdge → Automatic, AxesLabel → None,
 AxesStyle → Automatic, Background → Automatic, Boxed → True,
 BoxRatios → {1, 1, 0.4}, BoxStyle → Automatic, ClipFill → Automatic,
 ColorFunction → Automatic, ColorFunctionScaling → True,
 ColorOutput → Automatic, Compiled → True,
 DefaultColor → Automatic, DefaultFont :→ $DefaultFont,
 DisplayFunction :→ $DisplayFunction, Epilog → {},
 FaceGrids → None, FormatType :→ $FormatType, HiddenSurface → True,
 ImageSize → Automatic, Lighting → True, LightSources →
  {{{1., 0., 1.}, RGBColor[1, 0, 0]}, {{1., 1., 1.}, RGBColor[0, 1, 0]},
   {{0., 1., 1.}, RGBColor[0, 0, 1]}}, Mesh → True,
 MeshStyle → Automatic, Plot3Matrix → Automatic, PlotLabel → None,
 PlotPoints → 25, PlotRange → Automatic, PlotRegion → Automatic,
 Prolog → {}, Shading → True, SphericalRegion → False,
 TextStyle :→ $TextStyle, Ticks → Automatic, ViewCenter → Automatic,
 ViewPoint → {1.3, -2.4, 2.}, ViewVertical → {0., 0., 1.}}
```

En la siguiente tabla mostramos algunas de las anteriores. Muchas de las opciones que aparecen en la siguiente tabla son análogas a otras ya vistas para gráficos en 2D.

Opción	**Valor por defecto**	**Descripción**
Axes	True	incluye los ejes.
AxesLabel	None	etiqueta los ejes; *zlabel* etiqueta el eje *z* y *{xlabel, ylabel, zlabel}* etiqueta todos los ejes.
Boxed	True	encierra el gráfico 3D en una caja.
ColorFunction	Automatic	colores para ser usados; **`Hue[]`** usa una sucesión de tonos.
TextStyle	$TextStyle	estilo por defecto para el texto.
FaceGrids	None	malla sobre las caras de la caja; **`All`** dibuja una malla en cada cara.
HiddenSurface	True	representa la superficie como un sólido.
Lighting	True	simula focos de luz sobre la superficie.
Mesh	True	malla sobre la superficie.
PlotRange	Automatic	especifica un rango en las coordenadas; puede ser *All*, *{zmin, zmax}* o *{{xmin, xmax}, {ymin, ymax}, {zmin, zmax}}*.
Shading	True	colorea la superficie.
ViewPoint	{1.3, -2.4, 2}	punto en el espacio desde el que se ve la superficie.
PlotPoints	25	número de puntos en cada dirección a evaluar; *{nx,ny}* especifica números diferentes en cada eje.

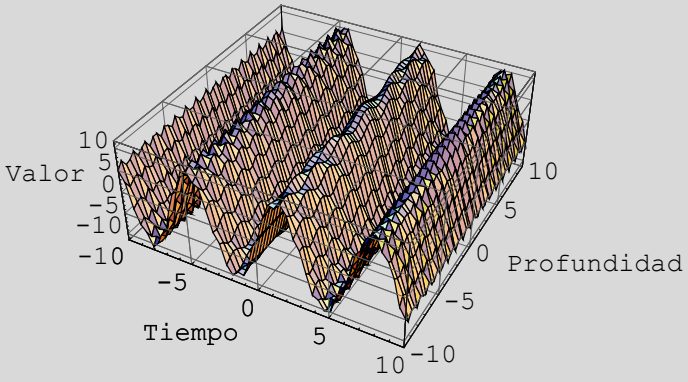
Veamos algunos ejemplos más:

```
Plot3D[10 Sin[x] + Sin[x y], {x, −10, 10}, {y, −10, 10}, PlotPoints → 50]
```



- SurfaceGraphics -

Show[%, AxesLabel → {"Tiempo", "Profundidad", "Valor"}, FaceGrids → All]

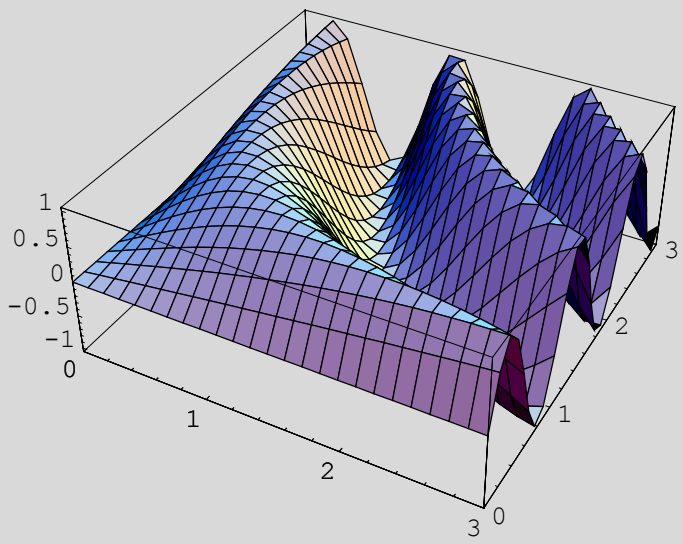


- SurfaceGraphics -

La opción **ViewPoint** para **Plot3D** y **Show** nos permite cambiar el punto de vista del observador. En muchas versiones de *Mathematica* existen otras maneras de elegir el punto de vista interactivamente.

El punto de vista de la siguiente superficie es el tomado por defecto (**{1.3, -2.4, 2}**).

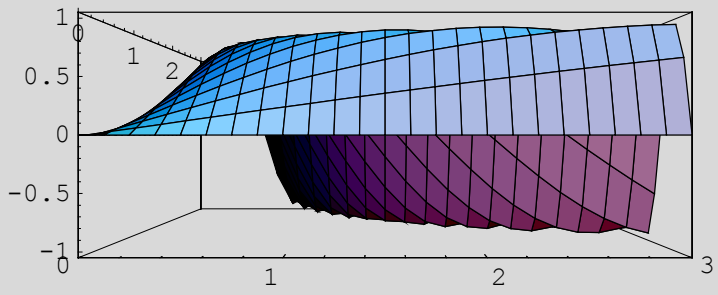
```
Plot3D[Sin[2 x y], {x, 0, 3}, {y, 0, 3}]
```




```
- SurfaceGraphics -
```

Tomemos un punto de vista justo en frente del eje OY. Nótese que se ha usado un efecto en la perspectiva que hace la parte del fondo más pequeña que la delantera:

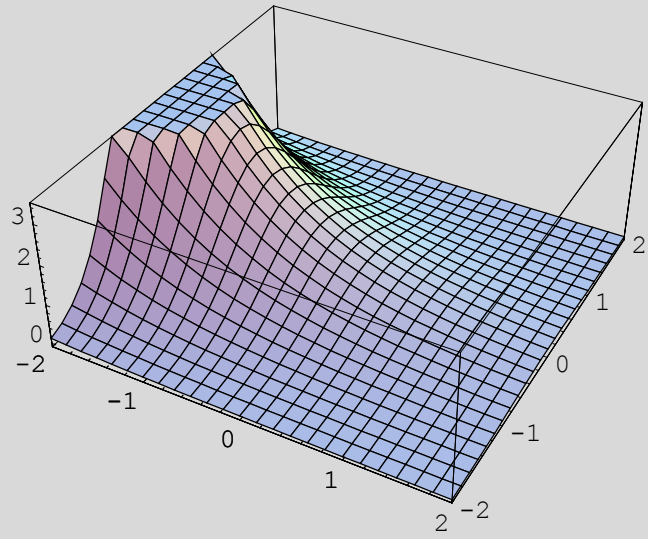
```
Show[%, ViewPoint → {0, −2, 0}]
```




```
- SurfaceGraphics -
```

Los puntos de vista ligeramente por encima de la superficie suelen funcionar bien. Generalmente es una buena idea tomar un punto de vista suficientemente cerca de la superficie. Añadir la caja alrededor de la superficie nos ayuda a reconocer la orientación de la superficie.

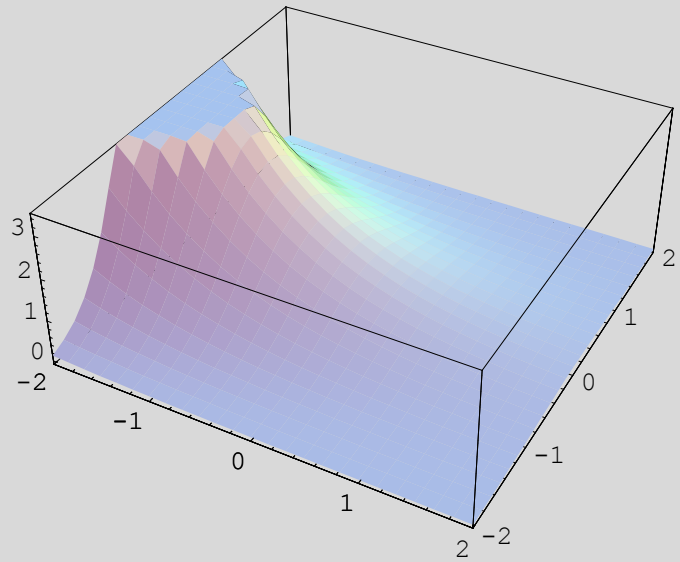
```
g = Plot3D[Exp[−(x + y^2)], {x, −2, 2}, {y, −2, 2}]
```




```
- SurfaceGraphics -
```

Veamos una superficie sin malla. Usualmente es más difícil apreciar la forma de la superficie sin la malla usual.

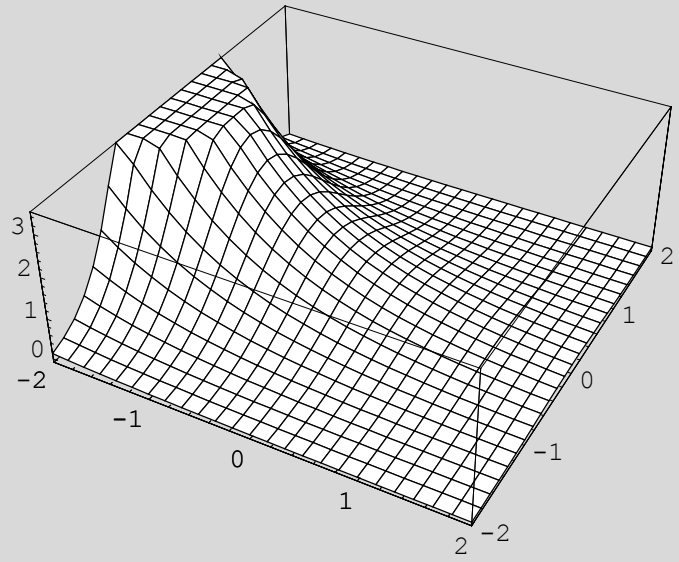
```
Show[g, Mesh → False]
```




```
- SurfaceGraphics -
```

Veamos ahora un superficie sin el efeto de color:

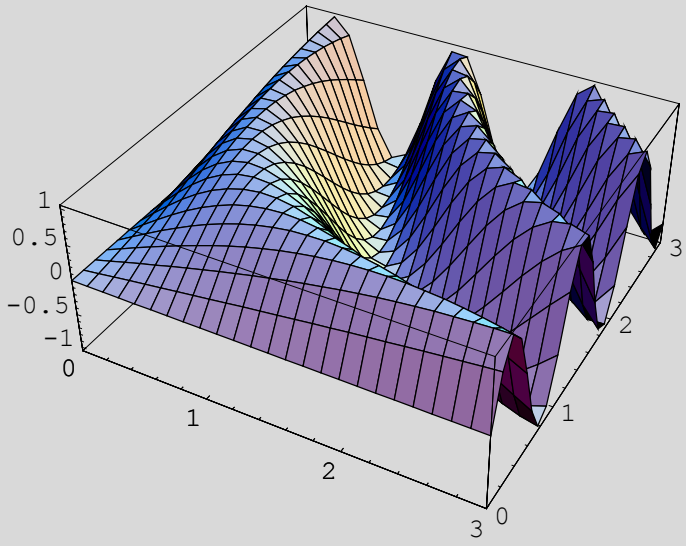
```
Show[g, Shading → False]
```




```
- SurfaceGraphics -
```

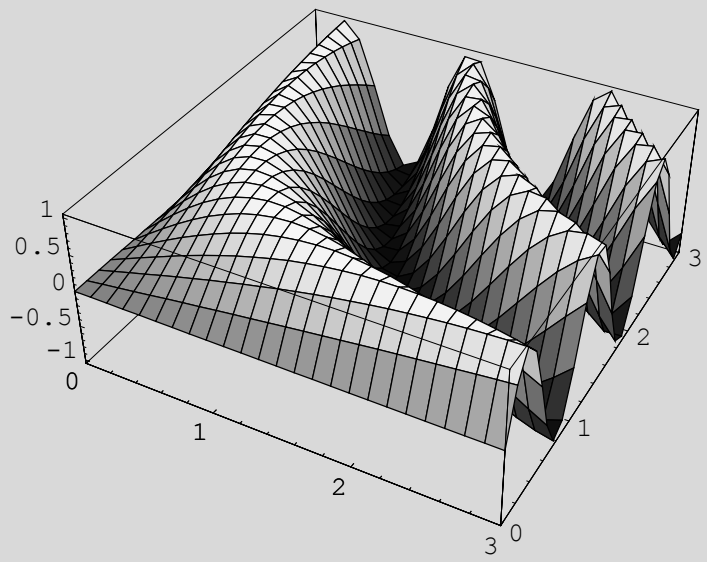
Para añadir un elemento extra de realismo, *Mathematica* simula una superficie iluminada. Por defecto, *Mathematica* asume que hay tres fuentes de luz sobre el objeto en la parte superior derecha. No obstante, este efecto luminoso puede confundirnos en algunos casos. Si seleccionamos la opción **`Lighting -> False`**, entonces *Mathematica* no usará tal efecto sino una escala de grises para sombrear la superficie en función de la altura del punto.

```
Plot3D[Sin[2 x y], {x, 0, 3}, {y, 0, 3}]
```




```
- SurfaceGraphics -
```

Show[%, Lighting → False]



- SurfaceGraphics -

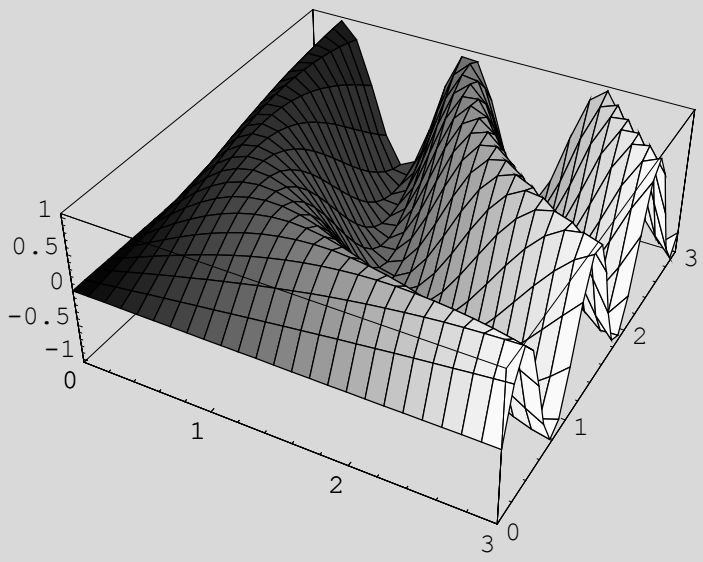
Como ya hemos comentado, con **Lighting -> False**, *Mathematica* sombrea la superficie en función de la altura. También podemos especificarle a *Mathematica* cómo sombrear cada parte de la superficie; usaremos para ello:

Plot3D[{f,GrayLevel[s]},{x,xmin,xmax}] representa **f** sombreando en gris de acuerdo a la función s.

Plot3D[{f,Hue[s]},{x,xmin,xmax}] sombrea variando el tono de color en lugar de usar la escala de grises.

La siguiente superficie corresponde a **Sin[2x y]**, pero el sombreado está determinado por **GrayLevel[x/3]**.

Plot3D[{Sin[2 x y], GrayLevel[x / 3]}, {x, 0, 3}, {y, 0, 3}]



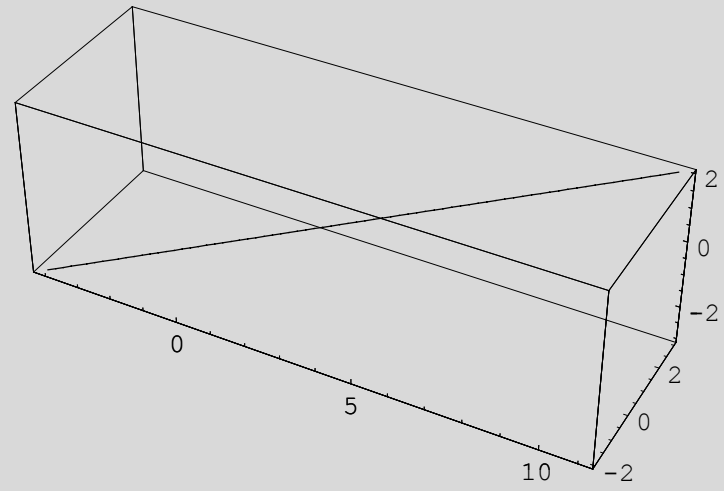
- SurfaceGraphics -

Para la representación gráfica en 3D de funciones definidas mediantes sus ecuaciones paramétricas podemos utilizar, de manera similar a como ya vimos para 2D, la orden ParametricPlot3D[{fx,fy,fz},{t,tmin,tmax},{u,umin,umax},...].

Por ejemplo, la recta que pasa por el punto **p=(2,0,-1)** en la dirección de **v=(3,1,1)**:

```
p = {2, 0, -1}
v = {3, 1, 1}
ParametricPlot3D[p + λ v, {λ, -2, 3}]

{2, 0, -1}
```

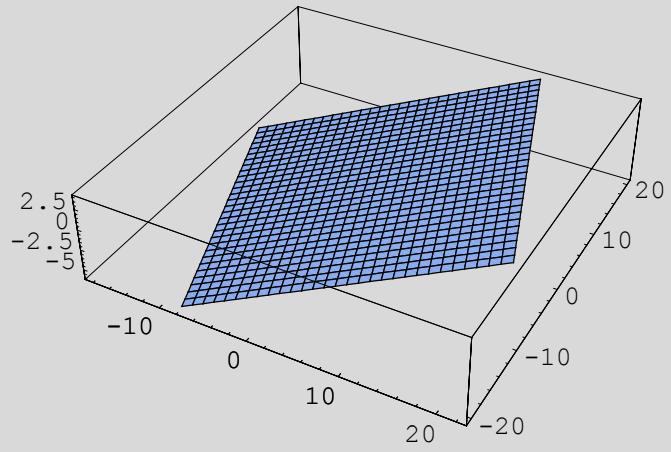
```
{3, 1, 1}
```




```
- Graphics3D -
```

Y ahora el plano que pasa por **p=(2,0,-1)** en la dirección de **v=(3,1,1)** y de **w=(1,-3,0)**:

```
w = {1, -3, 0}
ParametricPlot3D[p + λ v + μ w, {λ, -5, 5}, {μ, -5, 5}]

{1, -3, 0}
```

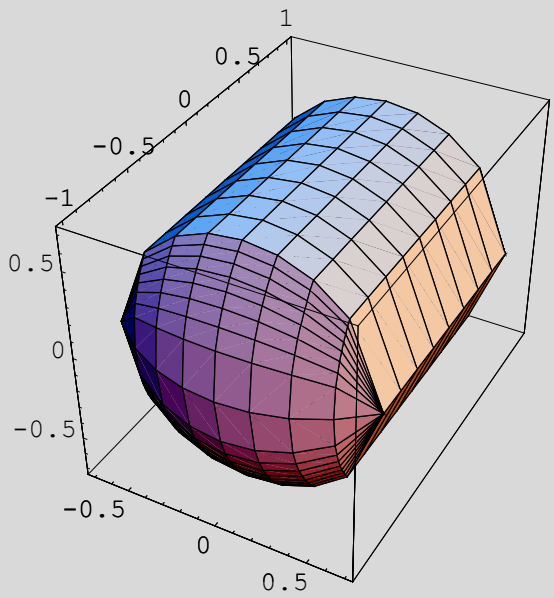


- Graphics3D -

También se pueden hacer representaciones gráficas en 3D de sólidos definidos con inecuaciones. Recuérdese que se debe de cargar antes el paquete correspondiente.

```
<< Graphics`InequalityGraphics`
```

```
InequalityPlot3D[ x^2 + y^2 + z^2 ≤ 1 ⋀ x^2 + z^2 ≤ 0.5, {x}, {y}, {z}];
```



Estadística

Dedicamos este último capítulo a mostrar brevemente algunas de las aplicaciones que posee *Mathematica* a la rama de la Estadística. Únicamente indicaremos algunos de los paquetes más importantes que el alumno podrá utilizar para resolver muchos de los problemas que se encontrará en una asignatura con contenido estadístico.

Estadística Descriptiva. Regresión y correlación lineal

Para no extendernos demasiado, enunciaremos las órdenes estadísticas más usuales y veremos cómo se utilizan con un ejemplo. Las funciones que enunciamos a continuación se encuentran en el paquete Statistics`DescriptiveStatistics`, que debemos cargar previamente.

```
<< Statistics`DescriptiveStatistics`
datos = {1, 2, 3, 3, 4, 6.5, 7, 8.9, 9, 17}

{1, 2, 3, 3, 4, 6.5, 7, 8.9, 9, 17}
```

Las funciones más usuales son:

Nombre	**Función**
Media	Mean[]
Moda	Mode[]
Mediana	Median[]

Media geométrica	GeometricMean[]
Media armónica	HarmonicMean[]
Cuantiles n (entre 0 y 1)	Quantile[datos,n]
Cuartiles	Quartiles[]
Rango	SampleRange[]
Varianza	VarianceMLE[]
Cuasivarianza	Variance[]
Desviación estándar	StandardDeviationMLE[]
Cuasidesviación estándar	StandardDeviation[]
Coeficiente de variación	CoefficientOfVariation[]
Momento central de orden n	CentralMoment[datos,n]
Coeficiente de asimetría	Skewness[]
Coeficiente de Curtosis	KurtosisExcess[]

Así, para nuestro ejemplo, la media, moda, mediana, el percentil 34, los cuartiles, la desviación estándar y el momento central de orden 3 son:

```
Mean[datos]
Mode[datos]
Median[datos]
Quantile[datos, 0.34]
Quartiles[datos]
StandardDeviationMLE[datos]
CentralMoment[datos, 3]

6.14
```

```
3
```

```
5.25
```

```
3.
```

```
{3, 5.25, 8.9}
```

```
4.4996
```

```
104.745
```

En el caso de que nuestra variable sea bidimensional y queramos realizar una regresión (lineal, cuadrática o de cualquier otro tipo) y una correlación, deberemos cargar previamente el paquete Statistics`LinearRegression`. A continuación, en primer lugar, introducimos los datos y, por último, utilizamos una de las dos siguientes funciones: Regress[datos,{1,f1(x),f2(x),...},x] o bien Fit[datos,{f1(x),f2(x),...},x].

A continuación introducimos los datos de la variable bidimensional (X,Y) como una lista de puntos en el plano:

```
<< Statistics`LinearRegression`
datos2 = {{1, 1}, {1.2, 1.5}, {2, 1.8}, {2.1, 2}, {2.5, 3}, {3, 6}}

{{1, 1}, {1.2, 1.5}, {2, 1.8}, {2.1, 2}, {2.5, 3}, {3, 6}}
```

En el caso de que queramos realizar una regresión lineal haremos:

```
Regress[datos2, {1, x}, x]

                        Estimate      SE           TStat         PValue
{ParameterTable → 1     -1.53514      1.22479      -1.25339      0.278327 ,
                  x     2.07719       0.587242     3.5372        0.0240741
 RSquared → 0.757748, AdjustedRSquared → 0.697186,
 EstimatedVariance → 0.997774, ANOVATable →
             DF      SumOfSq      MeanSq        FRatio       PValue
  Model      1       12.4839      12.4839       12.5118      0.0240741}
  Error      4       3.99109      0.997774
  Total      5       16.475
```

O bien:

```
Fit[datos2, {1, x}, x]

-1.53514 + 2.07719 x
```

La diferencia entre Regress[] y Fit[] es evidente. Mientra que Fit[] nos proporciona la recta de regresión de Y en función de X, la orden Regress[] realiza un estudio más completo y nos proporciona además el coeficiente de determinación (RSquared->0.757748) y, entre otras cosas, la tabla del análisis de la varianza (ANOVA).

Si, por ejemplo, nos interesa más una regresión cuadrática, bastará con añadir a la lista de funciones el término x^2:

```
Regress[datos2, {1, x, x^2}, x]

{ParameterTable →
          Estimate      SE            TStat         PValue
  1       4.63568       1.25777       3.68564       0.0346212
  x       -5.17166      1.39881       -3.69718      0.034345 ’
  x^2     1.85868       0.354499      5.24312       0.0135071
 RSquared → 0.976164, AdjustedRSquared → 0.960274,
 EstimatedVariance → 0.130897, ANOVATable →
             DF      SumOfSq       MeanSq        FRatio      PValue
  Model      2       16.0823       8.04115       61.431      0.00367993}
  Error      3       0.392692      0.130897                            }
  Total      5       16.475
```

de donde obtenemos que la parábola que mejor se ajusta es Y=4.63568-5.17166X+1.85868X^2 con un coeficiente de determinación R^2=0.976164.

En el caso de que el ajuste que deseemos sea, por ejemplo, exponencial, bastará con:

```
Regress[datos2, {1, Exp[x]}, x]

{ParameterTable →
          Estimate       SE             TStat         PValue
  1       0.0883523      0.339534       0.260217      0.807543    ,
  e^x     0.274219       0.0316264      8.67056       0.000973643
 RSquared → 0.949481, AdjustedRSquared → 0.936852,
 EstimatedVariance → 0.208074, ANOVATable →
             DF      SumOfSq       MeanSq        FRatio       PValue
  Model      1       15.6427       15.6427       75.1786      0.000973643}
  Error      4       0.832296      0.208074                              }
  Total      5       16.475
```

O un ajuste hiperbólico:

```
Regress[datos2, {1, 1/x}, x]

{ParameterTable →
          Estimate      SE           TStat         PValue
  1       5.46902       1.51611      3.60728       0.0226113,
  1/x     -4.9435       2.37632      -2.08032      0.105985
 RSquared → 0.519676, AdjustedRSquared → 0.399595,
 EstimatedVariance → 1.97833, ANOVATable →
             DF      SumOfSq      MeanSq       FRatio       PValue
  Model      1       8.56166      8.56166      4.32771      0.105985}
  Error      4       7.91334      1.97833
  Total      5       16.475
```

Cálculo de Probabilidades

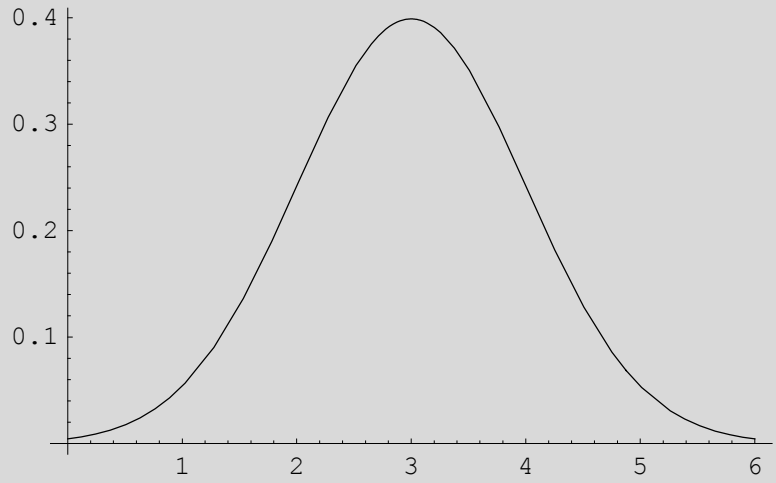
Para los casos en los que queramos manejar algún modelo de distribución de probabilidad, debemos cargar previamente el paquete Statistics`ContinuousDistributions` o bien Statistics`DiscreteDistributions` dependiendo de si nuestro modelo corresponde a una variable continua o discreta, respectivamente.

Así, si queremos definir una variable Normal de media 3 y desviación típica 1 haremos:

```
<< Statistics`ContinuousDistributions`
dist = NormalDistribution[3, 1]

NormalDistribution[3, 1]
```

La función de densidad de probabilidad de esta distribución en un punto x cualquiera (de su dominio) es PDF[dist,x] (Probability Density Function), cuya gráfica podemos representar:

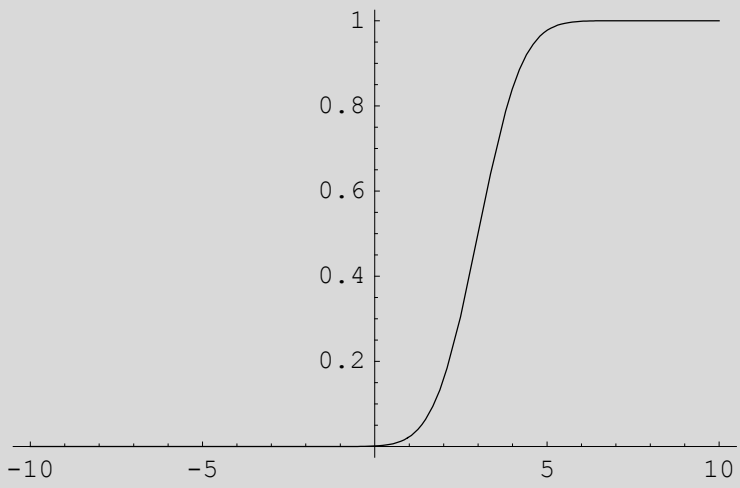
```
Plot[PDF[dist, x], {x, 0, 6}]
```




```
- Graphics -
```

El valor de la función de distribución en el punto x de su dominio es CDF[dist,x]. Esta función también podemos representarla:

```
Plot[CDF[dist, x], {x, -10, 10}]
```




```
- Graphics -
```

Una vez definida la distribución, podemos analizarla con muchas de las funciones introducidas en la sección anterior:

```
Domain[dist]
Mean[dist]
Variance[dist]
Skewness[dist]
Random[dist]

Interval[{-∞, ∞}]
```

```
3
```

```
1
```

```
0
```

```
2.77339
```

Obsérvese que hemos utilizado dos órdenes nuevas que no habíamos visto hasta ahora. Éstas son Domain[] y Random[]. La primera nos calcula el dominio de la distribución dist y la segunda genera un número aleatorio según la distribución dist (luego cada vez que la ejecutemos nos proporcionará una número distinto).

Para el caso de las distribuciones discretas, ya hemos comentado que se debe cargar el paquete Statistics`DiscreteDistributions` donde se encuentran cargadas las distribuciones discretas más importantes, como:

BinomialDistribution[n,p]	Distribución binomial.
GeometricDistribution[p]	Distribución geométrica.
HypergeometricDistribution[n,n1,n2]	Distribución hipergeométrica.
NegativeBinomialDistribution[n,p]	Distribución binomial negativa.
PoissonDistribution[m]	Distribución de Poisson.

Por ejemplo:

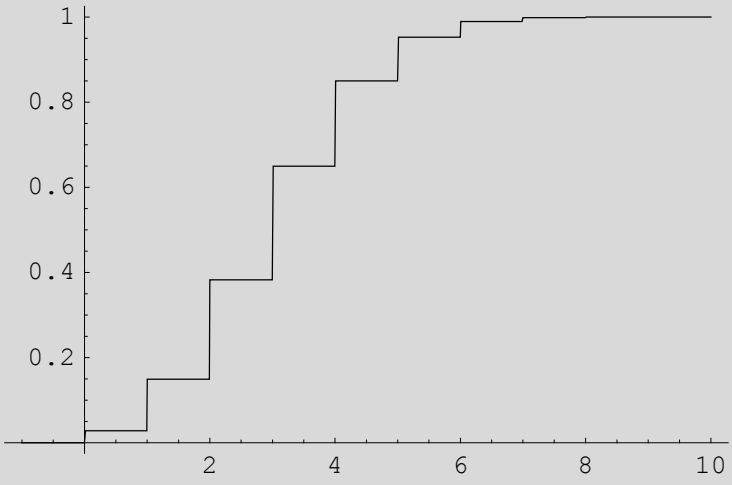
```
<< Statistics`DiscreteDistributions`
dist2 = BinomialDistribution[10, 0.3]
Domain[dist2]
Mean[dist2]
Variance[dist2]
Plot[CDF[dist2, x], {x, −1, 10}]
```

```
BinomialDistribution[10, 0.3]
```

```
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
```

```
3.
```

```
2.1
```




```
- Graphics -
```

Inferencia Estadística

Finalizamos el capítulo con otro breve comenteario sobre la Inferencia Estadística. Destacamos dos de los paquetes más importantes que posee *Mathematica* dentro de esta rama. En primer lugar, para calcular los intervalos de confianza usaremos el paquete Statistics`ConfidenceIntervals` y para los contrastes de hipótesis cargaremos previamente Statistics`HypothesisTests`.

Podemos calcular un intervalo de confianza para la media (con un nivel de confianza, por defecto, del 95%) de unos datos que suponemos aleatorios y distribuidos normalmente (también se supone la varianza desconocida) con la siguiente secuencia de órdenes:

```
<< Statistics`ConfidenceIntervals`
datos3 =
  {76, 73, 75, 73, 74, 74, 74, 74, 74, 77, 74, 72, 75, 76, 73, 71, 73, 80, 75, 75, 68, 72, 78, 74, 75}
```

```
{76, 73, 75, 73, 74, 74, 74, 74, 74, 77, 74,
 72, 75, 76, 73, 71, 73, 80, 75, 75, 68, 72, 78, 74, 75}
```

```
MeanCI[datos3]
```

```
{73.2393, 75.1607}
```

Para obtener un intervalo de confianza para la varianza se ejecuta la siguiente orden:

```
VarianceCI[datos3]
```

```
{3.3025, 10.4829}
```

En el caso de que queramos calcular un intervalo de confianza para la diferencia de medias de dos distribuciones que suponemos normales, debemos comprobar si se verifica la igualdad de varianzas:

```
datos4 = {78, 80, 80, 81, 82, 83, 87, 88, 88, 89}
datos5 = {76, 76, 78, 78, 78, 81, 81, 81, 82, 83}
VarianceRatioCI[datos4, datos5]
```

```
{78, 80, 80, 81, 82, 83, 87, 88, 88, 89}
```

```
{76, 76, 78, 78, 78, 81, 81, 81, 82, 83}
```

```
{0.644746, 10.4505}
```

Como el 1 está dentro del intervalo, no podemos rechazar la igualdad de varianzas. Luego el intervalo de confianza para la diferencia de las medias de datos4 y datos5 es

```
MeanDifferenceCI[datos4, datos5]

{1.00117, 7.39883}
```

Con respecto a los contrastes de hipótesis, comentaremos dos: el contraste para la media y el contraste para la diferencia de medias.

Si queremos contrastar si la media de la distribución normal que siguen los datos3 es, por ejemplo, 74, obtenemos el p-valor

```
<< Statistics`HypothesisTests`
MeanTest[datos3, 74]

OneSidedPValue → 0.335635
```

Como el p-valor es mayor que el nivel de significación tomado por defecto, 0.05, llegamos a la conclusión de que no podemos rechazar que la media de la distribución de datos3 sea 74.

Si queremos comprobar si la media de datos4 es mayor o igual que la media de datos5, podemos realizar un contraste para ver si la diferencia de ambas medias es cero. Obtenemos:

```
MeanDifferenceTest[datos4, datos5, 0]

OneSidedPValue → 0.00674533
```

Como el p-valor obtenido es menor que el nivel de significación, no podemos suponer que ambas medias sean iguales. Esta conclusión ya se podía haber obtenido cuando se calculó el intervalo de confianza para la diferencias de medias y vimos que el valor 0 no estaba incluido en éste.

Como en el resto de temas explicados, todo aquél que quiera profundizar más en esta parte que se ha tratado más superficialmente, le recomendamos que utilice la Ayuda tan ámplia y extensa que posee *Mathematica*.