Prácticas de ordenador_

10 octubre 2008

Cálculo

Ingeniería de Telecomunicaciones

Índice

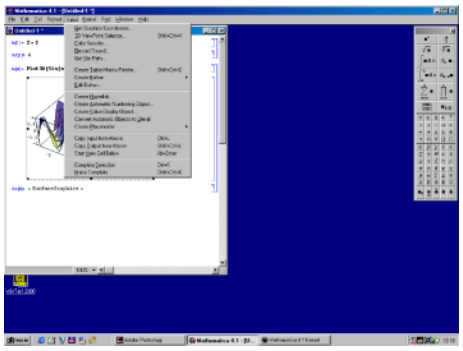
Primeros pasos

1

1.1 Operaciones elementales

Comenzaremos a familiarizarnos con *Mathematica*[1] haciendo, al principio, cosas muy simples que nos ayudarán a conocer los principales comandos y cómo se deben usar.

Para efectuar cálculos numéricos, el programa *Mathematica* funciona como una simple calculadora, con la salvedad de que con el *Mathematica* puedes obtener una precisión que no tienen las calculadoras normales.



Una vez que hemos empezado el programa nos encontramos delante de una pantalla más o menos como la de la figura. Arriba tienes la barra de Menú, debajo tienes una ventana vacía que es donde trabajaremos, y a la derecha tenemos una "paleta" con operaciones que ya iremos comentando con más detalle.

Una vez iniciada la ejecución del programa, pulsa con el botón izquierdo del ratón en la ventana principal y escribe: 3+2, y luego pulsa Shift+Return o Return en el teclado numérico. Observarás que en la pantalla aparece lo siguiente:

```
In[1]:=
       3+2
Out[1]=
       5
```

Observación 1.1. No intentes escribir los símbolos "`In[1]:=`" y "`Out[1]=`", ya que éstos los escribe el programa para llevar un control sobre las operaciones que va efectuando. No importa si dicho dígito de control no coincide con el que aparezca en este texto.

A continuación, con el ratón puedes pulsar sobre los números que habías escrito, y cambiarlos, o añadir nuevos sumandos, etc... y recuerda que siempre que quieras obtener el resultado, deberás pulsar Shift+Return. También podrás escribir los siguientes comandos debajo del resultado anterior, con lo que quedará constancia escrita de las operaciones que vas haciendo. Observarás que en las sucesivas operaciones el ordenador responde más rápido que en la primera, ya que al efectuar la primera operación, el programa debió leer del disco las "reglas de cálculo" (esto es, el *kernel* del programa), y dichas reglas permanecen en su memoria a partir de ese momento.

Para multiplicar números no es necesario escribir el símbolo de la multiplicación (opcionalmente "*"), y basta con poner un espacio entre los factores.

[1] *Mathematica* es una marca registrada de Wolfram Research Inc. En estas prácticas, salvo error, hemos utilizado la versión 3.0 de dicho programa.

```
In[2]:=
      3 2
Out[2]=
      6
```

⚠ **Observación 1.2.** Aunque no es necesario, es recomendable usar el asterisco (*) para escribir productos por una cuestión de legibilidad: en una fórmula es difícil distinguir entre 24 y 2 por 4.

```
In[3]:=
      5+(2 4+6)/7
Out[3]=
      7
```

Para efectuar potencias, puedes escribir

```
In[4]:=
      3^85
Out[4]=
      359175455476860593658082200801511413170 43
```

Ya que lo sabemos hacer directamente, comentemos que la paleta se puede usar, entre otras cosas, para escribir potencias. Si pulsas en el primer botón de la paleta te aparecerá en la ventana de comandos lo siguiente: $\blacksquare^{\square}$.

Puedes teclear directamente la base y, cuando termines, el tabulador te lleva al exponente. Una vez escrito pulsa como siempre (Shift)+(Return) y obtendrás el resultado, algo así:

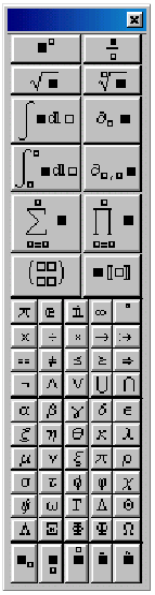


Figura 1.1 Paleta

```
In[5]:=
      2^3
Out[5]=
      8
```

Puedes hacer operaciones con fracciones, y obtener la fracción resultante...

```
In[6]:=
      2+3/13
Out[6]=
      29
      --
      13
```

(¿Por qué el resultado no ha sido $\frac{5}{13}$?) Si lo prefieres, puedes aproximar el resultado mediante unos cuantos dígitos de su expresión decimal

```
In[7]:=
      N[2+3/13]
Out[7]=
      2.23077
```

... y si quieres que el *Mathematica* calcule 41 dígitos (incluyendo la parte entera)...

```
In[8]:=
      N[2+3/13,41]
Out[8]=
      2.23076923076923076923076923076923076923076
```

¿Por qué el último dígito del comando anterior fue un 7 y no un 6? y ¿por qué ahora el último dígito ha sido un 8 y no un 7?

Si quieres, puedes poner dos mil decimales. Haz la prueba.

Sigamos probando... para obtener la raíz de un número se usa el comando `Sqrt` o utilizar la paleta (observa que la "S" es mayúscula, y el número debe ir entre corchetes):

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

... pues vale... y encima es hasta verdad... pero si quieres la expresión decimal con quince dígitos,

```
In[10]:=
      N[Sqrt[5],15]
Out[10]=
      2.23606797749979
```

También puedes hacer la raíz cuadrada de un número, elevando dicho número al exponente $\frac{1}{2}$

```
In[11]:=
      5^(1/2)
Out[11]=
      √5
```

¿Podrías ahora obtener una aproximación decimal de $\sqrt[7]{86}$ con doscientas cifras decimales?. Inténtalo.

Además de saber calcular las raíces, *Mathematica* también conoce las reglas de cálculo para operar con ellas:

```
In[12]:=
      Sqrt[2] Sqrt[3]
Out[12]=
      √6
```

...o bien...

```
In[13]:=
      Sqrt[5^(1/3)]
Out[13]=
      5^1/6
```

Como ya habrás notado, en la paleta tienes botones que permiten escribir fracciones, raíces de cualquier orden y algunas de las constantes más usuales como el número **e** o π.

Se supone que conocemos aproximadamente el valor del número π, y también *Mathematica* lo conoce...

```
In[14]:=
      Pi
Out[14]=
      π
```

...ya... parece más interesante así:

```
In[15]:=
      N[Pi]
Out[15]=
      3.14159
```

Es probable que alguno de nosotros conozca más dígitos, pero *Mathematica* es capaz de decirnos los primeros cinco mil dígitos de π. Prueba.

1.2 Resultados exactos y aproximación decimal

Hay una diferencia básica entre el concepto abstracto de número real y cómo trabajamos con ellos mediante un ordenador. Tenemos que tener en cuenta que la memoria y la capacidad de

proceso de un ordenador son finitos. La precisión de un ordenador es el número de dígitos con los que trabaja a la hora de hace los cálculos. En un hipotético ordenador que únicamente tuviera capacidad para almacenar el primer decimal, el número π sería representado como 3.1. Esto puede dar lugar a errores si, por ejemplo, restamos números similares. En ese hipotético ordenador que sólo puede almacenar el primer decimal la operación $\frac{2}{\pi-3.12}$ daría lugar a un error al dividir por cero.

Mathematica puede realizar los cálculos de forma simbólica o numérica. En principio, la primera forma es mejor pero hay ocasiones en las que no es posible. Para ello, *Mathematica* tiene dos tipos de "números": exactos y aproximados. La diferencia entre ambos es la esperable. $\frac{1}{3}$ es un número exacto y 0.333 es una aproximación del anterior. En una calculadora normal todos los números son aproximados y la precisión (el número de dígitos con el que trabaja la calculadora) es limitada, usualmente 10 o 12 dígitos.

Para *Mathematica* son números exactos los enteros, las fracciones o las constantes como π o e. Esto se traduce en que *Mathematica* es capaz de calcular tantos dígitos de dicho número como se requiera.

```
In[16]:=
        1/2 + 1/3
Out[16]=
        5/6
```

Esto ocurre porque *Mathematica* considera como números exactos los enteros 1, 2,... pero en cuanto aparece un número con un decimal escrito explícitamente como 1.3 *Mathematica* lo considera como número aproximado. Por ejemplo

```
In[17]:=
        1.2 + 1/2
Out[17]=
        1.7
```

Observación 1.3. *Mathematica* no cambia entre ambos tipos de números salvo que se lo indiquemos. Ahora bien, en cuento aparezcan números aproximados la salida vendrá dada en términos de números aproximados. Es importante que te des cuenta de que no es lo mismo escribir $\frac{1}{2}$ que 0.5. Observa la diferencia:

```
In[18]:=
        1/2 * E
Out[18]=
        E/2
```

En cambio,

```
In[19]:=
      0.5*E
Out[19]=
      1.35914
```

La orden N nos permite pasar de números reales a su aproximación númerica.

N[*expresión*, número] aproximación decimal de *expresión*

nos da la expresión decimal de la *expresión* con la cantidad de dígitos dada por *número.* También existe la posibilidad, en muchas ocasiones más cómoda, de escribir

expresión // N aproximación decimal de *expresión*

para obtener la aproximación de la *expresión* de *Mathematica* . Por ejemplo,

```
In[20]:=
      1/3 // N
Out[20]=
      0.33333
```

Precision[*expresión*] precisión de *expresión*

La orden Precision[*expresión*] nos da el número de dígitos de la *expresión* con los que *Mathematica* está operando. Por ejemplo, si el número es exacto la precisión es infinito:

```
In[21]:=
      Precision[π]
Out[21]=
      ∞
```

Si el número es aproximado, la precisión por defecto es de 16 dígitos:

```
In[22]:=
      Precision[√3.]
Out[22]=
      16
```

salvo que nosotros hayamos fijado una precisión mayor:

```
In[23]:=
      Precision[N[π,50]]
Out[23]=
      50
```

Recordemos que *no* podemos aumentar la precisión de un número: $\sqrt{3.}$ tiene una precisión de 16 dígitos por defecto aunque no lo muestre en pantalla.

```
In[24]:=
      √3.
Out[24]=
      1.73205
```

Si pedimos 50 dígitos, ¿cuál es la respuesta?. Prueba a calcular `N[√3.,20]` y `N[√3.,50]`.

Observación 1.4. Para saber más, puedes consultar en la ayuda de *Mathematica* `MachinePrecision`.

Ejercicio 1.1

Explica la respuesta de Mathematica al comando `N[N[Pi, 15], 30]`.

1.2.1 Números complejos

Después de calcular la raíz cuadrada de 2, lo siguiente que a uno se le ocurre es: vamos a "liar" al ordenador. Pongamos un número negativo:

```
In[25]:=
      Sqrt[-1]
Out[25]=
      I
```

No le hemos conseguido. *Mathematica* opera perfectamente con números complejos. De hecho, trabaja *siempre* con números complejos. Tendremos más noticias de esto cuando queramos encontrar las raíces de un polinomio. Mientras tanto nos contentamos con saber que `I` representa la unidad imaginaria. Puedes consultar el Capítulo ?? si quieres más detalles.

```
In[26]:=
      (3+2I)(1-I)
Out[26]=
      5-I
```

1.3 Algunas funciones

Además de las operaciones elementales que hemos visto, *Mathematica* tiene definidas la mayor parte de las funciones elementales. Los nombres de estas funciones suelen ser su abreviatura en inglés, que algunas veces difiere bastante de su nombre castellano. En general, cualquier comando de *Mathematica* se escribe con la primera letra en mayúscula. Por ejemplo

```
In[27]:=
      Sqrt[4]
Out[27]=
      2
```

`Sqrt[x]`	raíz cuadrada de x ($\sqrt{x}$)
`Exp[x]`	exponencial de x (e^x)
`Log[x]`	logaritmo neperiano de x, ($\log_e(x)$)
`Log[b,x]`	logaritmo en base b de x ($\log b(x)$)
`Sin[x], Cos[x], Tan[x]`	seno, coseno y tangente *en radianes*
`ArcSin[x], ArcCos[x], ArcTan[x]`	arcoseno, arcocoseno y arcotangente
`n!`	factorial de n
`Abs[x]`	valor absoluto de x ($\|x\|$)
`Random[]`	número aleatorio entre 0 y 1

Veamos con más detalle algunas de estas funciones.

- *Función exponencial:* `Exp[x]`

```
In[28]:=
      Exp[2]
Out[28]=
      E^2
```

(para *Mathematica* el número e se escribe `E`) y si queremos su expresión decimal

```
In[29]:=
      N[Exp[2]]
Out[29]=
      7.38906
```

Otra forma de calcular la función exponencial aplicada a x es elevando E a x. ¿Podrías calcular así e^5? ¿y su primera cifra decimal? Haz lo mismo usando la función Exp y comprueba que da el mismo resultado.

- *Función logaritmo neperiano:* `Log[x]`

```
In[30]:=
      Log[20]
Out[30]=
      Log[20]
```

...ya empezamos... Bueno, si lo que nos interesa es su expresión decimal

```
In[31]:=
      N[Log[20]]
Out[31]=
      2.99573
```

En *Mathematica* , si no decimos lo contrario, los logaritmos serán neperianos. Si queremos calcular el logaritmo en base b de x usamos `Log[b,x]`. Por ejemplo, el logaritmo decimal de 100

```
In[32]:=
      Log[10,100]
Out[32]=
      2
```

¿Cuánto valdrá el logaritmo en base 2 de 64? Compruébalo. ¿Cuánto debe valer Log[Exp[x]]? ¿Y Exp[Log[x]]? Pide a *Mathematica* que los calcule.

- *Funciones trigonométricas:* `Sin[x]`, `Cos[x]`, `Tan[x]` (seno, coseno y tangente)

```
In[33]:=
      Sin[Pi/4]
Out[33]=
      1/√2
```

Por defecto, las funciones trigonométricas están expresadas en radianes. Si queremos utilizar grados, podemos aprovechar la constante `Degree` o el símbolo (°) que vienen incorporados a *Mathematica* . Su valor es $\pi/180$, la razón de cambio de radianes a grados, lo que que nos permite escribir

```
In[34]:=
      Cos[45 Degree]
Out[34]=
       1
      √2
In[35]:=
      Sin[45°]
Out[35]=
       1
      √2
```

También podemos usar las funciones trigonométricas inversas, esto es, el arcoseno, arcocoseno y arcotangente, que se escriben respectivamente `ArcSin[x]`, `ArcCos[x]` y `ArcTan[x]`. Observa que la cuarta letra de cada comando está en mayúscula; escríbelo así, si no, *Mathematica* no lo entenderá.

```
In[36]:=
      ArcTan[1]
Out[36]=
      π
      4
```

Ejercicio 1.2

Calcula

a) los primeros cien decimales del número e,

b) el logaritmo en base 3 de

$$515377520732011331036461129765621272702107522001, \text{ y}$$

c) la tangente de $\frac{\pi}{2}$.

Ejercicio 1.3

Prueba a componer dos o más de estas funciones y a hacer cálculos con ellas.

⚠ **Observación 1.5.** El argumento de una función *siempre* se escribe entre corchetes y *todas* las funciones conocidas por *Mathematica* empiezan con mayúscula. Esta es una de las maneras más típicas de que *Mathematica* nos de un error. Estamos acostumbrados a usar $f(x)$ y eso no lo entiende *Mathematica* como la definición de una función.

1.4 Operadores lógicos y relacionales

x==y	igual
x!=y	distinto
x>y	mayor
x<y	menor
x>=y	mayor o igual
x<=y	menor o igual

Mathematica puede comprobar si se da una igualdad (o desigualdad). Sólo tenemos que escribirla y nos dirá qué le parece:

```
In[37]:=
      3<5
Out[37]=
      True
```

También podemos encadenar varias condiciones y en ese caso nos preguntamos si todas son ciertas.

```
In[38]:=
      3<5<=1
Out[38]=
      False
```

En este último ejemplo, queríamos saber si era cierto que $3 < 5$ **y** $5 \leq 1$. En *Mathematica* los operadores lógicos "y" y "o" se escriben && y ||. También podemos negar que se cumpla algo: lo hemos visto antes cuando hemos puesto "!" delante de un igual para obtener distinto.

&&	y
||	o
! *expresión*	negación de *expresión*

```
In[39]:=
      2<3 && 7!=1
Out[39]=
      True
```

Ejercicio 1.4

¿Qué número es mayor e^{π} o π^{e}? Responde a la misma pregunta con los números 1000000^{999999} y $999999^{1000000}$. ¿Sabrías hacerlo sin ordenador?

1.5 ¿Cuál era el resultado anterior?

% último resultado
%% penúltimo resultado
%número resultado *número*

Con *Mathematica* podemos usar el resultado de una operación anterior sin necesidad de teclearlo. Esto se consigue con la orden %. Si queremos el resultado de la salida *n* (lo que *Mathematica* ha escrito como Out[n]), podemos obtenerlo con %n. Por ejemplo, si queremos el resultado de la operación número 15 (unas cuantas páginas atrás),

```
In[40]:=
      %15
Out[40]=
      3.14159
```

además podemos usar esa información como cualquier otro dato

```
In[41]:=
      %15^2
Out[41]=
      25
```

Por ejemplo,

```
In[1]:=
      9*6
Out[1]=
      54
In[2]:=
      % + 10
Out[2]=
      64
In[3]:=
      Sqrt[%]+%%
Out[3]=
      62
```

Observación 1.6. % siempre se refiere a la última respuesta que haya dado *Mathematica* que no tiene que ser la que tienes justo encima del cursor. Es posible incluso que la hayas borrado o esté en una ventana diferente.

1.6 Variables

Cuando algún paso intermedio es importante, es cómodo ponerle nombre para poder volver a utilizarlo sin tener que escribirlo de nuevo. Ésta es una de las ideas que están detrás del uso de variables. Ahora vamos a trabajar con variables.

```
In[4]:=
      a=2
Out[4]=
      2
In[5]:=
      a^2
Out[5]=
      4
```

Podemos cambiar el valor de una variable, asignándole un nuevo valor:

```
In[6]:=
      a=0
Out[6]=
      0
```

Cuando una variable tenga asignado un valor concreto, a veces diremos que es una constante, para distinguir del caso en que no tenga ningún valor asignado. Veremos el interés de trabajar con variables y constantes en la siguiente sección cuando hablemos de cálculo simbólico.

Observación 1.7. El nombre de una variable puede ser cualquier cosa que no empiece por un número. Puede ser una palabra, una letra o una mezcla de ambas cosas. Como ya habrás observado, todas las órdenes, nombre de funciones, constantes, en fin, cualquier cosa que esté predefinida en *Mathematica* comienza con mayúscula. Por esto es mejor usar nombres de variable que empiecen con minúscula. De esta forma podemos distinguir las cosas que corresponden a *Mathematica* y las que hemos definido nosotros.

```
In[7]:=
      largo=10
Out[7]=
      10
In[8]:=
      ancho=7
Out[8]=
      7
In[9]:=
      largo*ancho
Out[9]=
      70
```

¿Cuál es la respuesta que da *Mathematica* a la siguiente entrada?

```
In[10]:=
      x=3
      y=4
      x+y
      x*y
      xy
```

Ésta es otra fuente frecuente de errores: `x*y` es el producto de las variables `x` e `y`, pero `xy` es un nombre de variable *nuevo* sin relación alguna con `x` e `y`. Moraleja: es más fácil encontrar * que un espacio en blanco a la hora de buscar un error.

Los valores que asignamos a una variable no se borran por si solos. Siguen en activo mientras no los cambiemos o comencemos una nueva sesión de *Mathematica* . Quizá por costumbre, todos tendemos a usar como nombre de variables x, y, z, t igual que los primeros nombres que se nos vienen a la cabeza de funciones son f o g. Después de trabajar un rato con *Mathematica* es fácil que usemos una variable que ya hemos definido antes. Esto suele ser una de las causas más frecuentes de error. ¿Cómo podemos evitarlo? Borrando las variables cuando no vayamos a usarlas más.

`a=1`	la variable a vale 1
`Clear[a,b]`	borrar los valores de las variables a y b
`a=.`	borra el valor de la variable a

Por ejemplo, veamos el uso de la orden `Clear[`*`nombre variable`*`]`.

```
In[11]:=
      a=7
Out[11]=
      7
In[12]:=
      a^2
Out[12]=
      49
```

es decir, le damos a la letra a el valor 7 y *Mathematica* la sustituye siempre por ese valor. Si usamos `Clear`

```
In[13]:=
      Clear[a]
```

(observa que esta entrada no produce ninguna salida)

```
In[14]:=
      a^2
Out[14]=
      a^2
```

ahora, a es una variable, esto es, no tiene un valor concreto y *Mathematica* debe tratarla simbólicamente. Si queremos que todas las constantes que hayamos definido pierdan su valor concreto (que pasen a ser variables) usaremos `Clear["Global`*"]` (el acento que hay que usar es el que está a la derecha de la p).

1.6.1 Asignación diferida

Cuando realizamos una asignación del tipo `a=1`, a partir del mismo instante en que ejecutamos dicha orden, la variable `a` vale 1. Sin embargo, hay ocasiones en las que es conveniente que *Mathematica* evalúe en cada ocasión el valor de dicha variable. Para poner un ejemplo aprovechemos la orden `Random[]` que devuelve un valor aleatorio comprendido entre 0 y 1. Si definimos

```
In[15]:=
      a=Random[]
```

hemos fijado el valor de `a` para siempre. Si utilizamos la asignación diferida

```
In[16]:=
      a:=Random[];a
```

Mathematica no asigna un valor inmediatamente a `a` sino que cada vez que la utilicemos, vuelve y calcula su valor: compruébalo calculando su valor un par de veces.

1.7 Valores y reglas

Ya hemos hablado de variables (y constantes). Sabemos que *Mathematica* puede operar con expresiones del tipo $1+x+x^2$ y que podemos dar un valor concreto a la variable `x` utilizando una asignación de la forma `x=2`.

expresión`/.`*regla* aplica la *regla* a la *expresión*

Prueba lo siguiente

```
In[17]:=
      1+x+x^2/. x->2
Out[17]=
      7
```

Mathematica entiende una expresión del tipo `x->2` como una regla que tiene que aplicar (por llevar delante `/.`). Esto quiere decir que donde haya una `x`, *Mathematica* la sustituye por un `2`. La regla no tiene por qué ser tan simple. Podemos hacer cosas como

```
In[18]:=
      1+x+x^2/.x->1+t
Out[18]=
      2+t+(1+t)^2
```

También se pueden utilizar varias reglas al mismo tiempo. Ya veremos que para escribir "lista" de reglas (o de cualquier cosa) escribimos entre llaves y separadas por comas todas las reglas que necesitemos.

```
In[19]:=
      x+2y-x^2/.{x->3,y->1-z}
Out[19]=
      -6+2(1-z)
```

¿Cuál es la diferencia entre aplicar reglas y realizar una asignación? ¿Cuáles son las ventajas o los incovenientes entre unos y otros? La principal diferencia entre ambos es el ámbito al que se aplican. Me explico: cuando realizamos una asignación del tipo `x=3`, a partir de ese momento, para toda la sesión y mientras no se le cambie o se le borre el valor, `x` *es* `3`. Por ejemplo:

```
In[20]:=
      x-x^2
Out[20]=
      x-x^2
In[21]:=
      x=2
Out[21]=
      2
In[22]:=
      %%
Out[22]=
      -2
```

La expresión `x-x^2` vale -2 y no podemos factorizarla ni utilizarla de manera simbólica. En cambio, utilizar reglas tiene un efecto "local": sólo afecta a la expresión a la que se aplica. Por ejemplo

```
In[23]:=
      y-y^2/.y->2
Out[23]=
      -2
In[24]:=
      y-y^2/.y->0
Out[24]=
      0
```

Ya habíamos comentado que una de las causas de error más comunes es asignar un valor a una variable (convertirla en constante) y utilizar más tarde esa variable sin darnos cuenta de que tiene un valor concreto. Si sólo vamos a utilizar una variable en un momento concreto entonces la solución correcta es utilizar reglas. Compáralo tu mismo: si queremos saber el valor de $(1+a)^2$ para $a=2$ podemos utilizar reglas

```
In[25]:=
      (1+a)^2/.a->2
Out[25]=
      9
```

o podemos utilizar una asignación

```
In[26]:=
      a=2
Out[26]=
      2
In[27]:=
      (1+a)^2
Out[27]=
      9
In[28]:=
      Clear[a]
```

1.7.1 Varias operaciones a la vez

Consideremos el siguiente ejemplo:

```
In[29]:=
      a=1
Out[29]=
      1
In[30]:=
      b=2
Out[30]=
      2
In[31]:=
      c=3
Out[31]=
      3
In[32]:=
      a*b*c
Out[32]=
      6
```

¡Tiene que ser posible escribirlo de una manera más corta! Además, ya sabemos que a vale 1...no hace falta que *Mathematica* vuelva a escribirlo. La solución es fácil: el punto y coma (;) permite encadenar varias operaciones y además indica a *Mathematica* que no escriba el resultado de la operación. Por ejemplo, si escribimos

```
In[33]:=
      a=0;
```

la variable "a" tendrá el valor cero pero *Mathematica* no da ninguna respuesta. Si quieres saber qué hubiese respondido sólo hay que preguntarlo:

```
In[34]:=
      %
Out[34]=
      0
```

Usando puntos y comas podemos escribir los cálculos anteriores de una manera más corta y elegante:

```
In[35]:=
      a=1;b=2;c=3;a*b*c
Out[35]=
      6
```

Como ves, sólo da el resultado de la última operación que es la única que no está terminada con un punto y coma.

Ejercicio 1.5

Razona la respuesta que da el *Mathematica* al siguiente comando:

```
In[36]:=
      a=5;
      b=12;
      N[Sqrt[a+b],b]
Out[36]=
      4.12310562562
```

1.8 Cálculo simbólico con *Mathematica*

Hasta ahora sólo hemos usado el *Mathematica* como una calculadora muy potente, pero prácticamente todo lo que hemos aprendido puede hacerse sin dificultad con una calculadora convencional. Entonces, ¿qué puede hacer *Mathematica* que sea imposible con una calculadora? Bueno, entre otras muchas cosas que veremos posteriormente, la principal utilidad de *Mathematica* es el cálculo simbólico, es decir, el trabajar con expresiones algebraicas (expresiones donde intervienen variables, constantes... y no tienen por qué tener un valor numérico concreto) en vez de con números. Por ejemplo, el programa sabe que la función Log es inversa de Exp, con lo que si ponemos

```
In[37]:=
      Exp[Log[x]]
Out[37]=
      x
```

es decir, sin saber el valor de la variable x el programa es capaz de trabajar simbólicamente con ella. Más ejemplos

```
In[38]:=
      Exp[x]Exp[y]
Out[38]=
      E^(x+y)
In[39]:=
      a+2a+5b
Out[39]=
      3 a + 5 b
```

Como comentamos, para *Mathematica* cualquier letra o combinación de letras puede ser una variable o una constante. Hay excepciones como la letra E, que siempre indica el número **e** y no puede usarse como variable ni constante. Sin embargo, las letras *x, y, z... a, b, c, d...* pueden usarse sin problemas como nombres de constantes y variables.

Vamos a practicar con comandos de *Mathematica* para manejar expresiones algebraicas: polinomios, funciones racionales, expresiones trigonométricas, ecuaciones...

1.8.1 Cálculos algebraicos

Si introducimos el siguiente polinomio

```
In[40]:=
      x^2 + 2 x + 1
Out[40]=
      1 + 2 x + x^2
```

lo único qu hace *Mathematica* es reescribirlo en orden ascendente. Si escribimos

```
In[41]:=
      (x+1)^2
Out[41]=
      (1+x)^2
```

tampoco *Mathematica* desarrolla el cuadrado. Probemos ahora a restar las dos expresiones:

```
In[42]:=
      %% - %
Out[42]=
      1 + 2 x + x^2 - (1 + x)^2
```

Mathematica no se da cuenta de que la expresión vale cero. Esto es porque sólo realiza automáticamente algunas operaciones sencillas como cambiar $x^2 + 2x^2$ por $3x^2$. No factoriza ni desarrolla automáticamente: debemos decirle que lo haga. ¿Cómo lo hacemos? Con las siguientes órdenes

`Expand[`*`expresion`*`]`	realiza productos y potencias
`ExpandAll[`*`expresion`*`]`	aplica `Expand` en todas partes
`Factor[`*`expresion`*`]`	escribe la expresión como producto de factores más sencillos
`Together[`*`expresion`*`]`	aplica mínimo común denominador
`Apart[`*`expresion`*`]`	separar en fracciones simples

La orden `Expand` desarrollo productos y potencias ya sea en una variable

```
In[43]:=
      Expand[(x-2)(x-1)x(x+1)(x+2)]
Out[43]=
      4 x - 5 x^3 + x^5
```

o con varias variables

```
In[44]:=
      Expand[(2y-x)(-3x+y)(x+y)]
Out[44]=
      3 x^3 - 4 x^2 y - 5 x y^2 + 2 y^3
```

La orden `Factor` nos devuelve la expresión anterior. Se podría decir que `Expand` y `Factor` son operaciones inversas.

```
In[45]:=
      Factor[%]
Out[45]=
      (x - 2 y) (3 x - y) (x + y)
In[46]:=
      Expand[1+2x+x^2 - (1+x)^2]
Out[46]=
      0
In[47]:=
      Factor[1+2x+x^2 - (1+x)^2]
Out[47]=
      0
```

En el caso de que haya fracciones en la expresión, `Expand` no tiene ningún efecto sobre los denominadores. `ExpandAll` es la solución. Esta orden aplica la orden `Expand` en todas partes de la expresión. Por ejemplo,

```
In[48]:=
      Expand[(1+x)/(x-1)^2 + (x-3)^2/(x+7)^3]
Out[48]=
      1/(-1+x)^2 + x/(-1+x)^2 + 9/(x+7)^3 + 6x/(x+7)^3 + x^2/(x+7)^3
```

En cambio,

```
In[49]:=
      ExpandAll[(1+x)/(x-1)^2 + (x-3)^2/(x+7)^3]
Out[49]=
      1/(1-2x+x^2) + x/(1-2x+x^2) + 9/(343+147x+21x^2+x^3) + 6x/(343+147x+21x^2+x^3)
      + x^2/(343+147x+21x^2+x^3)
```

La orden `Apart` nos permite descomponer en fracciones simples y `Together` vuelve a unirlas en una sóla fracción.

```
In[50]:=
      Apart[1/(x^4 - 1)]
Out[50]=
       1          1          1
      ------ - ------- - ---------
      4(-1+x)   4(1+x)   2(1+x^2)
In[51]:=
      Together[%]
Out[51]=
              1
      ---------------
      (-1+x)(1+x)(1+x^2)
```

1.8.2 Simplificar expresiones

`Simplify[`*expresion*`]`	buscar una expresión más simple usando las reglas usuales
`FullSimplify[`*expresion*`]`	similar a la anterior pero con un conjunto más amplio de reglas

La orden `Simplify` busca la forma más sencilla en la puede escribir una expresión. Para ello usa la mayoría de las reglas usuales. Pero, ¿qué es la forma más sencilla? Para *Mathematica* , la forma más sencilla es la más corta, la que tiene menos partes.

```
In[52]:=
      Simplify[x^2+2x+1+(1+x)^2]
Out[52]=
      2(1 + x)^2
In[53]:=
      Simplify[ 1/4(1/(x-1) - 1/(x+1) - 2/(1+x^2))]
Out[53]=
       1
      -------
      -1+x^4
```

Hay veces que la expresión es demasiado complicada y `Simplify` no da ningún resultado práctico. En este caso se puede intentar usar la orden `FullSimplify`, pero hay que tener en cuenta que el tiempo para realizar los cálculos puede aumentar mucho. `FullSimplify` intenta combinar de todas las formas posibles las diferentes partes de la expresión y, para expresiones largas, esto puede tardar mucho.

```
In[54]:=
      Simplify[∛(5√13 − 18)]
Out[54]=
      (-18 + 5 √13)^(1/3)
In[55]:=
      FullSimplify[∛(5√13 − 18)]
Out[55]=
      1/2 (−3 + √13)
```

1.8.3 Expresiones trigonométricas

Mathematica conoce las identidades trigonométricas y puede usarlas para simplificar expresiones en las que aparezcan dichas funciones. En lugar de `Expand` y `Factor`, utilizaremos las órdenes `TrigExpand` y `TrigFactor`.

`TrigExpand[`*`expresion`*`]`	escribe la expresión como suma de términos
`TrigFactor[`*`expresion`*`]`	escribe la expresión como producto de términos
`TrigReduce[`*`expresion`*`]`	simplifica la expresión usando ángulos diferentes

Por ejemplo,

```
In[56]:=
      TrigExpand[Cos[α + β]]
Out[56]=
      Cos[α] Cos[β]-Sin[α] Sin[β]
In[57]:=
      TrigExpand[Sin[2ArcTan[t]]]
Out[57]=
      2t/(1+t^2)
In[58]:=
      TrigExpand[Sin[x+3*y]+Cos[2* z] Sin[x-y]]
Out[58]=
      Cos[y]^3 Sin[x] + Cos[y] Cos[z]^2 Sin[x] +
      3 Cos[x] Cos[y]^2 Sin[y] - Cos[x] Cos[z]^2 Sin[y] -
      3 Cos[y] Sin[x] Sin[y]^2 - Cos[x] Sin[y]^3 -
      Cos[y] Sin[x] Sin[z]^2 + Cos[x] Sin[y] Sin[z]^2
In[59]:=
      TrigExpand[8 Sin[2*x]^2 Cos[x]^3]
Out[59]=
      4Cos[x]^3 - 4 Cos[x]^7 + 24 Cos[x]^5 Sin[x]^2 - 4Cos[x]^3 Sin[x]^4
```

El siguiente ejemplo muestra la diferencia entre `TrigFactor` y `TrigReduce`:

```
In[60]:=
      TrigReduce[Sin[a+b]*Cos[a+b]]
Out[60]=
      1/2 Sin[2a+2b]
In[61]:=
      TrigExpand[Sin[a+b]*Cos[a+b]]
Out[61]=
      Cos[a] Cos[b]^2 Sin[a] + Cos[a]^2 Cos[b] Sin[b] -
      Cos[b] Sin[a]^2 Sin[b] - Cos[a] Sin[a] Sin[b]^2
In[62]:=
      TrigFactor[Sin[a+b]*Cos[a+b]]
Out[62]=
      Sin[a+b] Cos[a+b]
```

`TrigExpand` también trabaja con funciones hiperbólicas:

```
In[63]:=
      TrigExpand[Sinh[2 x]^3]
Out[63]=
      -3/2 Cosh[x] Sinh[x] + 3/2 Cosh[x]^5 Sinh[x] + 5 Cosh[x]^3 Sinh[x]^3 +
        3/2 Cosh[x] Sinh[x]^5
```

Ejercicio 1.6

Mediante el comando `TrigExpand`, comprueba las siguientes igualdades trigonométricas

a) $\operatorname{sen}(x+y) = \operatorname{sen}(x)\cos(y) + \cos(x)\operatorname{sen}(y)$,

b) $\cos(x+y) = \cos(x)\cos(y) - \operatorname{sen}(x)\operatorname{sen}(y)$,

c) $\operatorname{sen}(2x) = 2\operatorname{sen}(x)\cos(x)$,

d) $\cos(2x) = \cos^2(x) - \operatorname{sen}^2(x) = 2\cos^2(x) - 1 = 1 - 2\operatorname{sen}^2(x)$,

e) $\operatorname{sen}(a) + \operatorname{sen}(b) = 2\operatorname{sen}\left(\frac{a+b}{2}\right)\cos\left(\frac{a-b}{2}\right)$,

f) $\cos(a) + \cos(b) = 2\cos\left(\frac{a+b}{2}\right)\cos\left(\frac{a-b}{2}\right)$

Ejercicio 1.7

Comprueba si las funciones trigonométricas y las correspondientes "arco"-versiones son inversas.

1.9 Listas, vectores y matrices

Mathematica tiene una manera fácil de agrupar objetos con las mismas propiedades, ya sean números, funciones, etc. y poder operar con ellos. Una lista se escribe agrupando entre llaves los objetos que queramos separados por comas. Por ejemplo,

```
In[1]:=
      {0,1,-3}
Out[1]=
      {0,1,-3}
```

es una lista de números (o sea, un vector). También podemos escribir listas de funciones

```
In[2]:=
      {x,x^2,x^3}
In[3]:=
      {x,x^2,x^3}
```

Los elementos que forman la lista pueden ser, a su vez, listas (piensa en matrices como "listas de vectores"):

```
In[4]:=
      {{1,2,1},{0,1,-1},{7,1.5,1}}
Out[4]=
      {{1,2,1},{0,1,-1},{7,1.5,1}}
```

Si queremos extraer un elemento o varios de una lista, sólo tenemos que dar su posición. Por ejemplo,

```
In[5]:=
      lista={{1,2,1},{0,1,-1},{7,1.5,1}}
Out[5]=
      {{1,2,1},{0,1,-1},{7,1.5,1}}
In[6]:=
      lista[[1]]
Out[6]=
      {1,2,1}
```

nos da el primer elemento de la lista. También podemos extraer varios elementos de la lista enumerando sus posiciones. Por ejemplo, el primer y el último elemento de la lista anterior se obtendrían escribiendo

```
In[7]:=
      lista[[{1,3}]]
Out[7]=
      {{1,2,1},{7,1.5,1}}
```

En este caso los elementos de la lista vuelven a serlo y, por ejemplo, podemos obtener el segundo elemento de la tercera lista:

```
In[8]:=
      lista[[3]][[2]]
Out[8]=
      1.5
```

También conseguimos el mismo efecto escribiendo

```
In[9]:=
      lista[[3,2]]
Out[9]=
      1.5
```

1.9.1 Vectores y matrices

Vectores y matrices son simplemente listas y listas de listas. El vector $(1, 2-1)$ es, simplemente, la lista $\{1, 2, -1\}$. Las operaciones usuales de vectores o matrices se escriben como puedes imaginar. La suma se realiza coordenada a coordenada, ya sean vectores o matrices:

```
In[10]:=
      A={{1,2,3},{4,5,6},{1,3,2}}
      B={{4,3,2},{7,3,1},{1,0,0}}
      a={1,5,8}
      b={1,3,4}
Out[10]=
      {{1,2,3},{4,5,6},{1,3,2}}
Out[11]=
      {{4,3,2},{7,3,1},{1,0,0}}
Out[12]=
      {1,5,8}
Out[13]=
      {1,3,4}
In[14]:=
      a+b
Out[14]=
      {2,8,12}
In[15]:=
      A+B
Out[15]=
      {{5,5,5},{11,8,7},{2,3,2}}
```

Para el producto escalar de dos vectores o el producto de una matriz por un vector se utiliza el punto, `.`, en lugar del asterisco.

```
In[16]:=
      a.b
Out[16]=
      48
In[17]:=
      A.a
Out[17]=
      {35,77,32}
```

Eso sí, como has visto *Mathematica* no muestra el resultado en la forma de matriz que estamos acostumbrados a ver. Conseguiremos esto mediante el uso de la orden `MatrixForm`.

```
In[18]:=
      MatrixForm[A]
Out[18]=  // MatrixForm=
```

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 1 & 3 & 2 \end{pmatrix}$$

Table

Con *Mathematica* podemos definir matrices de forma automática mediante la orden `Table`. La sintaxis es la siguiente

`Table[`*función(i,j)*`,{i,`*límite de i*`},{j,`*límite de j*`}]` matriz de entradas *función(i,j)*

Puedes comprenderla mejor con el siguiente ejemplo: la entrada (i, j) de la matriz es $ix + jy$. Además i y j varían de 1 a 3.

```
In[19]:=
      Table[ i*x+ j*y, {i, 3},{j,3}]
Out[19]=
      {{x+y,x+2y,x+3y},{2x+y,2x+2y,2x+3y},
      {3x+y,3x+2y,3x+3y}}
In[20]:=
      MatrixForm[%]
Out[20]=   // MatrixForm=
```

$$\begin{pmatrix} \text{x+y} & \text{x+2y} & \text{x+3y} \\ \text{2x+y} & \text{2x+2y} & \text{2x+3y} \\ \text{3x+y} & \text{3x+2y} & \text{3x+3y} \end{pmatrix}$$

Cómo "dibujar" con Mathematica

2

2.1 Funciones

Una de las grandes virtudes de *Mathematica* es lo fácil y completo que es su tratamiento de los gráficos para funciones de una o varias variables. Es posible dibujar a la vez varias funciones y personalizar el resultado en cuanto a escalas, color, etc. También se pueden representar funciones en coordenadas paramétricas o realizar animaciones.

Un primer paso antes de empezar a representar gráficamente una función es tener una manera cómoda de definirla. Siguiendo la costumbre usaremos las letras f, g,... para nombrarlas (aunque puedes usar cualquier nombre que sea válido como variable). Para definir la función $\operatorname{sen}(x)$ haremos lo siguiente

```
In[1]:=
      f[x_]=Sin[x]
Out[1]=
      Sin[x]
```

Observa que se usan los corchetes para todo y que después de la variable x aparece _. No hay que olvidar este guión después de cada variable *cuando estemos definiendo una función*.

`f[x_]=x^2`	define la función $f(x) = x^2$
`?f`	muestra la definición de f
`Clear[f]`	borra la definición de f

Una vez definida la función podemos evaluarla en un punto, derivarla, representarla gráficamente, etc. Por ejemplo,

```
In[2]:=
      f[1]
Out[2]=
      Sin[1]
```

o si queremos su valor numérico

```
In[3]:=
      N[f[1]]
Out[3]=
      0.841471
```

También podemos asignar un valor a la variable x y luego evaluar:

```
In[4]:=
      x=0;
      f[x]
Out[4]=
      0
```

Aunque hubiera sido más fácil (y no necesitaríamos borrar el valor de x) hacerlo de la siguiente forma

```
In[5]:=
      Clear[x]
In[6]:=
      f[x]/.x->0
Out[6]=
      0
```

Fíjate que a la hora de evaluar una función no se usa el "_". Sólo se utiliza en el momento de definir la función.

Pueden surgir problemas al definir una función si la variable que usamos tiene asignado un valor concreto, con lo que la función sería constante; en este caso, resolvemos el problema borrando valores con la sentencia `Clear`. También podemos comprobar si ya hemos usado f como función. Por ejemplo, si queremos definir $f(x) = \ln(x)$, podemos comprobar en primer lugar si la función f ya está definida.

```
In[7]:=
      ?f
Out[7]=
      Global‘f
      Sin[x]
In[8]:=
      Clear[x,f];
      f[x_]=Log[x]
```

y la función queda bien definida aunque antes hubiésemos asignado un valor a x.

Funciones a trozos

También se pueden definir funciones "a trozos", pero en ese caso *no* se puede utilizar el símbolo (=) sino (:=). Por ejemplo, para definir la función

$$f(t) = \begin{cases} t, & \text{si } 0 < t < 2 \\ -t+4, & \text{si } t \notin]0,2[\end{cases}$$

escribimos

```
f[t_]:= t /; t>0 && t<2
f[t_]:= -t+4 /; t<=0 || t>=2
```

Observa que *Mathematica* no produce ninguna salida. Recuerda que esto ocurre con las asignaciones diferidas.

El símbolo `/;` hace las veces de **si** (condicional); los símbolos || y && son respectivamente **o** e **y** (suma y producto lógicos).

Ejercicio 2.1

a) Define la función

$$f(x) = \begin{cases} e^{3x+1}, & \text{si } x \in [0,10[\\ \ln\left(x^2+1\right), & \text{en otro caso.} \end{cases}$$

b) Calcula el valor de f en 1, 2 y -7.

c) Usa la orden `Table` para calcular el valor de f en los primeros cien naturales.

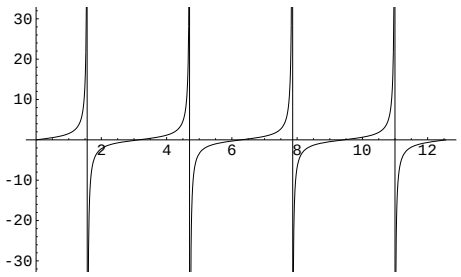
2.2 El comando Plot

Para representar gráficamente una función de una variable, el comando que se utiliza es

`Plot[`*función*`,{`x`,`x_{min}`,`x_{max}`},`*opciones*`]`	dibuja la gráfica de la *función* en el intervalo $[x_{min}, x_{max}]$

Mathematica decide automáticamente cuál es la escala más apropiada para que la gráfica se vea lo mejor posible. Si la función que pretendemos representar tiene alguna discontinuidad o no está definida en algún punto, *Mathematica* intentará dibujarla poniendo de manifiesto este hecho. Por ejemplo, intenta dibujar la función tangente entre 0 y 4π:

```
In[9]:=
      Plot[Tan[x],{x,0, 4Pi}]
```




```
Out[9]=  - Graphics -
```

Un comentario sobre lo que acabamos de hacer: como se puede ver, podemos escribir directamente la función a representar "dentro" del comando `Plot`. El inconveniente que tiene hacerlo así es que cada vez que queramos trabajar con esa función tenemos que volver a escribirla, siendo más conveniente ponerle nombre (p.e. `f[x_]=Tan[x]`, y usar después `Plot[f[x],{x,0,4Pi}]`.

⚠ **Observación 2.1.** Para dibujar la gráfica de una función *Mathematica* utiliza el método que hemos dicho en clase que *no* hay que utilizar: *Mathematica* evalúa la función en varios puntos del intervalo donde queremos representarla y luego une dichos puntos mediante segmentos. Este método da una representación bastante fiel si el número de puntos que tomamos es alto pero si la función tiene grandes variaciones puede pasar cualquier cosa.

2.2.1 Opciones de Plot

Cuando presentamos el comando `Plot` dijimos que podíamos poner "opciones"; veamos cuáles son algunas de ellas:

a) `AspectRatio-->`*número*

Puedes modificar el tamaño o la relación entre anchura y altura de cualquier gráfica utilizando el ratón. Por defecto, la relación entre la altura y la anchura viene dada por la llamada razón aurea, $\frac{1+\sqrt{5}}{2}$, independientemente de la función que estemos representado. La opción `AspectRatio` permite modificar este valor.

Si queremos que *Mathematica* tenga en cuenta la gráfica que vamos a representar a la hora de elegir la escala podemos utilizar `AspectRatio->Automatic`:

```
In[10]:=
      Plot[Sin[2πx],{x,0,1},AspecRatio->Automatic]
```

También podemos elegir nosotros la proporción de la gráfica. Si especificamos el valor 1, `AspectRatio->1`, los dos ejes tendrán el mismo tamaño. En el siguiente ejemplo el eje OY será dos veces más grande que el eje OX.

```
In[11]:=
      Plot[Tan[x],{x,0,4Pi},AspectRatio->2]
```

Esta opción nos será útil para hacer "redondas" algunas gráficas. Prueba a representar $f(x) = \sqrt{1-x^2}$ en $[-1,1]$. El resultado debería ser una semicircunferencia pero aparece aplastada. Para arreglarlo, utiliza la opción `AspecRatio->1`. Muchas veces, incluido el ejemplo anterior, es suficiente con utilizar `AspecRatio->Automatic`. Comprueba el resultado de

```
In[12]:=
      Plot[√1-x^2,{x,-1,1},AspectRatio->Automatic]
```

b) `PlotRange-->{`*número1,número2*`}`

Por defecto, *Mathematica* selecciona los valores de la función que considera más importantes. Eso hace que puede cortar la gráfica de una función que crezca muy rápidamente. Por ejemplo,

```
In[13]:=
      Plot[Exp[x^3],{x,-1.5,1.5}]
```

Podemos decirle a *Mathematica* que muestre el intervalo completo donde toma valores la función mediante la opción `PlotRange->All`:

```
In[14]:=
      Plot[Exp[x^3],{x,-1.5,1.5},PlotRange->All]
```

aunque *Mathematica* no dibuja la gráfica completa.

Con la opción `PlotRange->{`*número1,número2*`}` sólo se presentan los valores de $f(x)$ comprendidos entre *número1* y *número2*.

```
In[15]:=
      Plot[Tan[x],{x,0,4Pi},PlotRange->{2,20}]
```

c) `PlotPoints->{`*número*`}`

Especifica cuántos puntos utiliza *Mathematica* para dibujar la función. En la mayoría de las ocasiones la gráfica que nos da el ordenador con los valores por defecto es adecuada. Hay que tener claro lo que *Mathematica* hace para dibujar la función: simplificando, calcula el valor de la función en veintitantos puntos y luego los une suavizando la gráfica. Cuando la función varíe mucho entre puntos cercanos puede ser conveniente aumentar el número de puntos que *Mathematica* utiliza para dibujarla. Normalmente, una cantidad entre 50 y 100 puntos suele ser más que suficiente. Prueba con lo siguiente

```
In[16]:=
      Plot[Sin[x],{x,0,3*Pi}],PlotPoints->75]
```

d) `PlotStyle`

Permite escoger, entre otras cosas, el color y el grosor del trazo con el que se dibuja la gráfica de la función. Hay más opciones pero nosotros sólo nos vamos a centrar en el color.

```
PlotStyle->RGBColor[número1,número2,número3]
```

El color se elige en función de la cantidad de rojo (*número1*), verde (*número2*) y azul (*número3*), donde los números pueden tomar cualquier valor entre 0 y 1. Por ejemplo, escribe lo siguiente si quieres dibujar la gráfica de la función tangente entre 0 y 4π en color rojo:

```
In[17]:=
      Plot[Tan[x],{x,0,4Pi},PlotStyle->RGBColor[1,0,0]]
```

Veamos algunos ejemplos de cómo utilizar estas opciones. Prueba las siguientes órdenes, compara los resultados e intenta variar algunos de los parámetros "a ver qué pasa".

```
Clear[x,f];
f[x_]=Exp[x] ArcTan[x];
Plot[f[x],{x,0,5},PlotRange->{-1,30}];
```

También se pueden dibujar varias gráficas de funciones a la vez. Una de las formas de distinguirlas es dibujando cada una de un color diferente:

```
Plot[{Sin[x],Cos[x]}, {x,-2 Pi,2 Pi},
  PlotStyle->{RGBColor[1,0,0],RGBColor[0,0,1]}];
```

Observación 2.2. Si quieres volver a usar los mismos nombres para funciones distintas recuerda "limpiar" los nombres de las variables y las funciones con el comando `Clear`; si quieres borrar los valores de *todas* las variables, usa `Clear["Global`*"]`. Por ejemplo, si quieres volver a usar las letras f y g para otras dos funciones haz lo siguiente:

```
Clear[f,g,x];
f[x_]=Sin[2 x];
g[x_]=Cos[4 x];
Plot[{f[x],g[x]},{x,0,10 Pi},
  PlotStyle->{RGBColor[1,0,1],RGBColor[0,1,0.5]}];
```

2.3 Curvas en el plano

Como ya recordarás, al menos para rectas, podemos representar una curva del plano en coordenadas paramétricas, o sea, a partir de las coordenadas $(x(t), y(t))$ variando t a lo largo de un intervalo.

Para esto tenemos la siguiente orden `ParametricPlot`

`ParametricPlot[{x(t),y(t)},{t,a,b},`*`opciones`*`]`	dibuja la gráfica de la curva $(x(t), y(t))$ en $[a, b]$

junto con las opciones ya conocidas de la orden `Plot`. Prueba con alguno de los siguientes ejemplos:

```
ParametricPlot[{t,t^2},{t,-2,2}]
ParametricPlot[{Cos[t],Sin[t]},{t,0,2 Pi}]
ParametricPlot[{{4 Cos[t],3 Sin[t]},{8 Cos[t],7 Sin[t]}},{t,0,2 Pi}]
ParametricPlot[{Cos[2 t],Sin[2 t]},{t,0,2 Pi}]
```

¿Hay alguna diferencia entre el segundo y el último de los ejemplos?. ¿Cuál es?

2.3.1 Algunas curvas planas

- *Astroide*. El camino recorrido por un punto de un círculo de radio r rodando dentro de otro círculo de radio $4r$ parametrizado en coordenadas rectangulares es el siguiente:

```
ParametricPlot[{3*Cos[t]/4+ Cos[3*t]/4, 3*Sin[t]/4
  - Sin[3*t]/4},{t, 0, 2 Pi}]
```

- *Cardioide*. Es la curva trazada por un punto fijo de un círculo de radio r que rueda sin deslizarse alrededor de otro círculo fijo del mismo radio.

```
ParametricPlot[{2 Cos[t] + Cos[2 t], 2 Sin[t]
  + Sin[2 t]},{t,0,2 Pi}]
```

- *Catenaria*. Es la forma que adopta un cable ideal con densidad uniforme colgando de dos puntos.

```
ParametricPlot[{t,Cosh[t]},{t,-2,2}]
```

- *Lemniscata de Bernoulli*. Es una parametrización de $(x^2 + y^2)^2 = (x^2 - y^2)$, también llamada hipérbola lemniscata.

```
ParametricPlot[{Cos[t]/(1+Sin[t]^2), Sin[t] Cos[t]/(1+Sin[t]^2)},{t,0,Pi}]
```

- *Espiral equiangular*. Es aquella espiral en la que el radio corta a la curva en un ángulo constante α.

```
ParametricPlot[{ E^(t*Cot[α])*Cos[t], E^(t*Cot[α])*Sin[t]},{t,0,10 Pi}]
```

- *Serpentina*. Es una parametrización de $x^2y + a^2y - b^2x = 0$.

```
ParametricPlot[{t,(b^2 t)/(a^2 + t^2)},{t,-10,10}]
```

- *Hipérbola*. La representación en coordenadas cartesianas de una hipérbola con vértice $(1,0)$ y foco $(a,0)$ (excentricidad a) es $x^2 - y^2/(a^2 - 1) = 1$. Una hipérbola rectangular tiene excentricidad $\sqrt{2}$ y su ecuación es $xy = 1$.

```
ParametricPlot[{-Sec[t],Sqrt[a^2-1] Tan[t]},{t,0,2 Pi}]
```

2.4 El comando Show

Como recordarás, el símbolo % era utilizado para referirse al resultado anterior. Cuando el resultado es un gráfico que queremos volver a visualizar usaremos el comando Show en la forma siguiente

```
Show[%1]
```

Si queremos volver a representar los gráficos 1 y 2:

```
Show[%1,%2]
```

La orden Show nos permite volver a mostrar gráficos sin necesidad de recalcularlos. Ten en cuenta que pueden ser necesarios muchos cálculos para representar gráficos (sobre todo en tres dimensiones).

⚠ **Observación 2.3.** El punto y coma, (;), al final de una orden hace que *Mathematica* ejecute dicha orden pero que no muestre el resultado en pantalla. Si hacemos lo mismo con un gráfico lo único que conseguimos es que *Mathematica* no escriba - Graphics - después de la gráfica. Para que una gráfica no se muestre en pantalla tenemos la opción DisplayFunction->Identity. Si queremos que las vuelva a mostrar debemos utilizar DisplayFunction->$DisplayFunction. Veamos un ejemplo:

```
In[18]:=
      dib1=Plot[Sin[x],{x,0,2Pi},
        PlotStyle->RGBColor[1,0,0],DisplayFunction->Identity];
In[19]:=
      dib2=Plot[Cos[x],{x,0,2Pi},
        PlotStyle->RGBColor[0,0,1],DisplayFunction->Identity];
In[20]:=
      Show[dib1,dib2,DisplayFunction->$DisplayFunction]
```

2.5 Animaciones

Las gráficas de funciones que dependen de un parámetro se pueden conseguir utilizando listas con la orden Table o con el comando Animate.

Animate[*gráfico*,*parámetro*]	serie de gráficos en función de un parámetro

Por ejemplo,

```
In[21]:=
      Animate[Plot[Sin[x+n],{x,-5,5}],{n,1,10}]
```

dibuja las funciones $\text{sen}(x+n)$ con $n = 1, ..., 10$. Cuenta cuántos gráficos has hecho. ¿Son diez?

Observación 2.4. La orden Animate *no* funciona directamente en versiones posteriores de *Mathematica* . Es necesario cargar el paquete *Animation* mediante

```
In[22]:=
      <<Graphics`Animation`
```

Puedes conseguir el mismo resultado utilizando la orden `Table`. Compruébalo con

```
In[23]:=
      Table[Plot[Sin[x+n],{x,-5,5}],{n,1,10}]
```

2.6 Ejercicios

Ejercicio 2.2

a) Representa en una misma gráfica las funciones seno y coseno en el intervalo $[-2\pi, 2\pi]$. Utiliza las opciones de la orden `Plot` para representarlas en diferente color.
b) Dibuja las funciones $\text{sen}(x)$ y $\text{sen}(-x)$ o con las funciones $\cos(x)$ y $\cos(-x)$. ¿Qué función es par y cuál es impar?.
c) Comprueba la paridad de las funciones: $f(x) = 2^{-x}\,\text{sen}(6x)$, $g(x) = 2^{-x}$ y $h(x) = -2^{-x}$.

Ejercicio 2.3

Dibuja la función logaritmo neperiano, exponencial y $f(x) = x^2$ con colores diferentes. Compara el crecimiento de estas funciones cerca de cero y lejos de cero. ¿Qué pasa si la base de la exponencial y el logaritmo son menores que uno?

Ejercicio 2.4

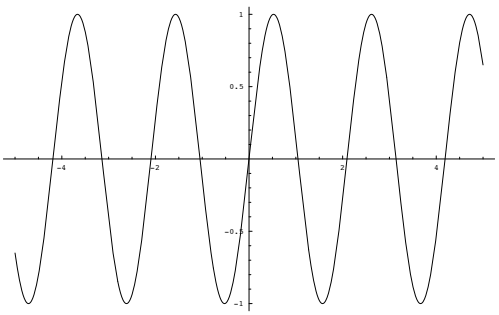
Dibuja las funciones seno y coseno hiperbólico. ¿Alguna de ellas es par o impar?

Ejercicio 2.5

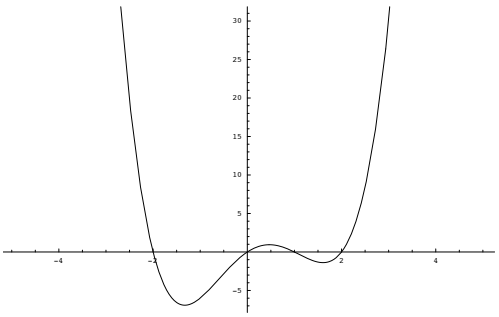
¿Cómo cambia la gráfica de una función $f(x)$ cuando la cambiamos por $f(a*x)$, $a*f(x)$, $f(x+a)$, $f(x)+a$?. Compruébalo dibujando
a) $\text{sen}(n*x)$ para $n = 1, 2, ..., 15$,
b) $n*\text{sen}(x)$, para $n = 1, 2, ..., 15$,
c) $\text{sen}(x)+n$, para $n = 1, 2, ..., 15$, y
d) $\text{sen}(x+n)$, para $n = 1, 2, ..., 15$.

Ejercicio 2.6

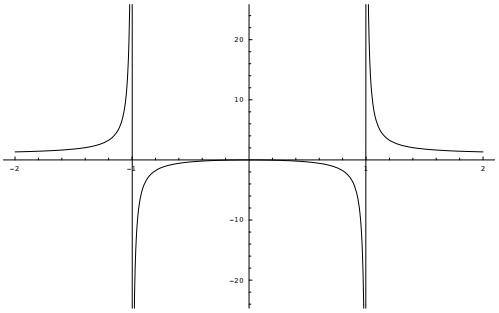
A continuación puedes encontrar la representación de varias funciones. Encuentra de qué funciones provienen.



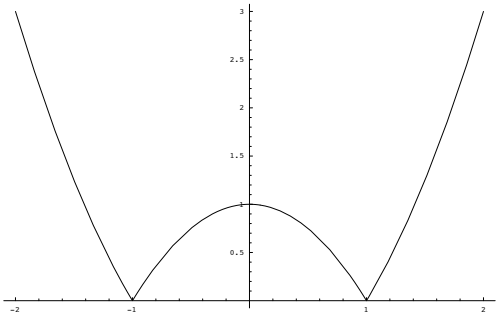
(a)



(b)



(c)



(d)

Derivación

3

3.1 Continuidad y límites

Básicamente todas las funciones que han aparecido hasta ahora eran continuas. En el caso de que no lo fueran *Mathematica* tampoco parece haber tenido demasiados problemas. Cuando hemos dibujado la gráfica de la función tangente, *Mathematica* ha dibujado una asíntota vertical en los puntos donde el coseno valía cero.

En el caso de que queramos estudiar la continuidad de una función en un punto a, tenemos que calcular $\lim_{x\to a} f(x)$. *Mathematica* puede estudiar límites (también laterales). La orden es la siguiente:

`Limit[`*expresión*`,variable->a]`	Límite en a
`Limit[`*expresión*`,variable->a,Direction->1]`	Límite en a por la izquierda
`Limit[`*expresión*`,variable->a,Direction->1]`	Límite en a por la derecha

La primera calcula el límite de la expresión cuando la variable tiende a a. Las siguientes calculan el límite por la izquierda y la derecha respectivamente. Por ejemplo,

```
In[1]:=
      Limit[n/(n+1),n->+∞]
Out[1]=
      1
In[2]:=
      Limit[Tan[x],x->Pi/2,Direction->1]
Out[2]=
      ∞
```

Cuando la función oscila entre dos valores *Mathematica* intenta dar el intervalo aunque no exista el límite.

```
In[3]:=
      Limit[Sin[1/x],x->0]
Out[3]=
      Interval[{-1,1}]
```

Hasta ahora hemos visto la utilidad de la orden `Limit`. Ahora vamos a ver su principal inconveniente. La función valor absoluto no es derivable en el origen: los límites laterales son distintos. Vamos a comprobarlo:

```
In[4]:=
      Limit[Abs[x]/x, x->1,Direction->1]
Out[4]=
      -1
In[5]:=
      Limit[Abs[x]/x, x->1,Direction->-1]
Out[5]=
      1
```

Perfecto. Acabamos de comprobar que los límites laterales son distintos pero, ¿qué hubiera pasado si no supiéramos que teníamos que estudiar los límites laterales por separado? La situación normal es justamente la contraria. Imaginemos por un momento que queremos estudiar la derivabilidad en el origen del valor absoluto:

```
In[6]:=
      Limit[Abs[x]/x, x->1]
Out[6]=
      1
```

¿Qué ocurre aquí? *Mathematica* está calculando un límite lateral sin avisarnos. A la vista de esto hay que tener mucho cuidado porque es posible que un límite no exista y *Mathematica* responda que sí.

3.2 Derivadas

La derivada de una función previamente definida se puede calcular de una forma muy similar a como lo escribes normalmente.

```
In[7]:=
      f[x_]=x^3+Sin[x]
Out[7]=
      x^3+Sin[x]
In[8]:=
      f'[x]
Out[8]=
      3x^2+Cos[x]
```

El único detalle importante es no confundir (') con cualquier otro acento. La tecla (') la tienes junto con el signo (?).

La derivada vuelve a ser una función que podemos evaluar en un punto o derivar de nuevo.

```
In[9]:=
      f'[1]
Out[9]=
      3+Cos[1]
In[10]:=
      f''[x]
Out[10]=
      6x-Sin[x]
```

De acuerdo. ¿Cuál es la cuarta derivada de f?

```
In[11]:=
      f''''[x]
Out[11]=
      Sin[x]
```

¿Cuánto vale la derivada de orden 8? No parece muy práctico escribir $f''''''''(x)$. es más cómodo utilizar el comando `D` que nos da la derivada de una función.

`D[`*expresión*`,{`*variable*`,`*orden*`}]` derivada de orden dado

De esta manera, la derivada de orden 8 de f es

```
In[12]:=
      D[f[x],{x,8}]
Out[12]=
      Sin[x]
```

No es necesario definir previamente la función. Podemos derivar una expresión directamente:

```
In[13]:=
      D[Cos[x*y],{x,2}]
Out[13]=
      -y^2Cos[x y]
```

Aunque volveremos a recordarlo cuando veamos funciones de varias variables: sí, lo que acabas de hacer es una derivada parcial de segundo orden.

3.3 Rectas tangentes y secantes

Recta tangente

El valor de la derivada de una función f en un punto a es la pendiente de la recta tangente en dicho punto. Dicha recta es $y = f(a) + f'(a)(x - a)$. La siguiente función nos permite calcularla:

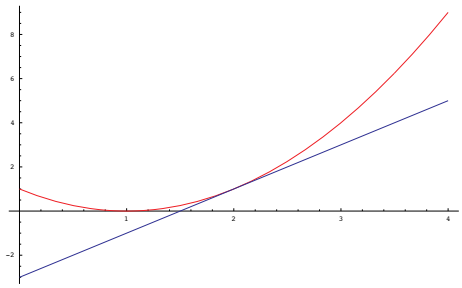
```
In[14]:=
      tangente[f_,a_][x_] := f[a]+f'[a]*(x-a)
```

Si queremos calcular, por ejemplo, la recta tangente de la función $f(x) = 1 - 2x + x^2$ en el punto $a = 2$:

```
In[15]:=
      f[x_]=1-2x+x^2
Out[15]=
      1-2x+x^2
In[16]:=
      tangente[f,2][x]
Out[16]=
      1+2(-2+x)
```

Hemos utlizado `:=` para definir la función `tangente` y, por tanto, no da ninguna respuesta pero podemos utilizarlo para dibujar la función y su derivada. Compruébalo. ¿Cuál es la respuesta de la siguiente orden?

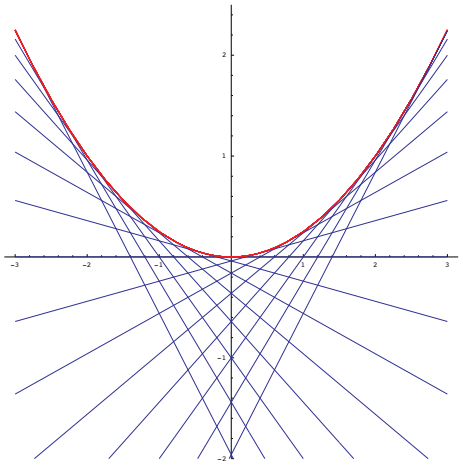
```
In[17]:=
      Plot[{f[x],tangente[f,2][x]},{x,0,4},
        PlotStyle->{RGBColor[1,0,0],RGBColor[0,0,1]}]
```



```
Out[17]=  - Graphics -
```

También podemos utilizar la orden `Table` para dibujar la familia de las rectas tangentes a, en este caso, la paráboloa $y = \frac{x^2}{4}$.

```
In[18]:=
      Clear[f];f[x_]=x^2/4
Out[18]=
       x^2
       ---
        4
In[19]:=
      Show[Table[Plot[{f[x],tangente[f,-3+6k/30][x]},{x,-3,3},
        PlotRange->{-2,2.5}, PlotStyle->{RGBColor[1,0,0],RGBColor[0,0,1]},
        AspectRatio->1, DisplayFunction->Identity],
        {k,1,29,2}],DisplayFunction->$DisplayFunction];
```



Rectas secantes

La recta tangente se obtiene como límite de las rectas secantes a la gráfica de f que pasan por los puntos $(a, f(a))$ y $(x, f(x))$ cuando x tiende a a. La fórmula de la recta que pasa por los puntos (a, c) y (b, d) es

$$y = \frac{(c-d)x}{a-b} + \frac{ad-bc}{a-b}$$

siempre que a sea distinto de b. Si $a = b$, la recta es vertical.

Igual que con la derivada, podemos automatizar su cálculo:

```
In[20]:=
      recta[{a_,c_},{b_,d_}][x_]:= (c-d)/(a-b)*x+(a*d-b*c)/(a-b)
```

Para ver cómo se van acercando las rectas secantes a la tangente, vamos a dibujar en un mismo gráfico la recta tangente y la recta secante que pasa por los puntos $(a, f(a))$ y $(a+h, f(a+h))$.

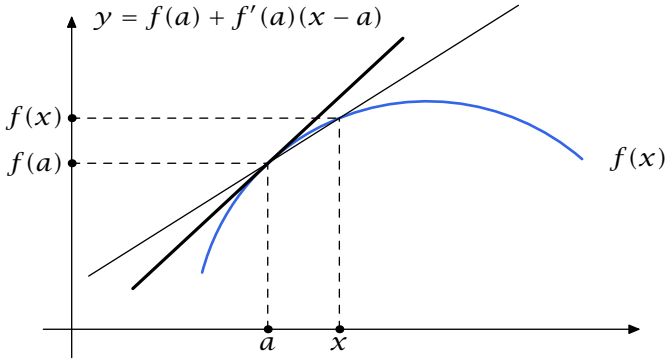


Figura 3.1 Recta tangente

Dando valores a h tendentes a cero podemos hacernos una idea bastante aproximada de qué está pasando. Por ejemplo,

```
In[21]:=
      Clear[f]; f[x_]:=-(x-3)^2+6;
In[22]:=
      Animate[Plot[{f[x],tangente[f,1][x],recta[{1,f[1]},{1-h,f[1-h]}][x]},
        {x,-2,5}, PlotRange->{-2,8},
        PlotStyle->{RGBColor[0,0,1],RGBColor[0,0,0],RGBColor[1,0,0]}],
        {h,-3,-.2,.2}];
```

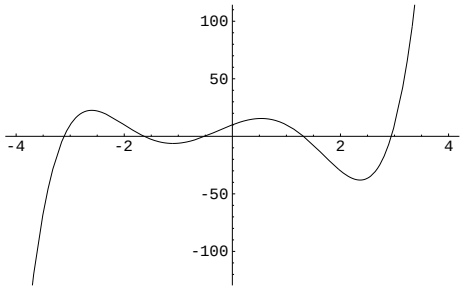
dibuja la función $f(x) = 6 - (x-3)^2$, su recta tangente en $a = 1$ y las rectas secantes.

3.4 Máximos y mínimos relativos

Esencialmente, lo que haremos para localizar los extremos relativos de una función f será encontrar soluciones de la ecuación $f'(x) = 0$. Para solucionar dicha ecuación, podemos usar (cuando sea posible por la simplicidad de la ecuación) comandos directos como `Solve` ó `NSolve`, o bien usaremos el comando `FindRoot` cuando las soluciones no se puedan obtener directamente.

Ejemplo 3.1. Comencemos buscando los extremos relativos de la función polinómica $f(x) = x^5 + x^4 - 11x^3 - 9x^2 + 18x + 10$ en el intervalo $[-4, 4]$. Para ello, definiremos convenientemente la función f, y dibujaremos su gráfica entre -4 y 4.

```
In[23]:=
      Clear["Global`*"]
      f[x_]= x^5+x^4-11x^3-9x^2+18x+10;
      Plot[f[x],{x,-4,4}]
```




```
Out[23]=  - Graphics -
```

A simple vista observaremos que hay:
a) un máximo relativo entre -3 y -2,
b) un mínimo relativo entre -2 y -1,
c) un máximo relativo entre 0 y 1,
d) un mínimo relativo entre 2 y 3.
En cualquier caso, dada la simplicidad de esta función (polinómica de grado bajo), es posible calcular directamente los ceros de la derivada con el comando

```
In[24]:=
      NSolve[f'[x]==0,x]
Out[24]=
      {{x->-2.60082},{x->-1.09686},{x->0.533861},{x->2.36382}}
```

Como vemos, aparecen cuatro soluciones que se corresponden con los extremos que habíamos localizado a simple vista. Para comprobarlo, podemos evaluar la segunda derivada en dichos puntos. Para ello podemos cortar y pegar o, mucho mejor, podríamos guardar el resultado de la orden `NSolve` en una variable y luego evaluar la segunda derivada:

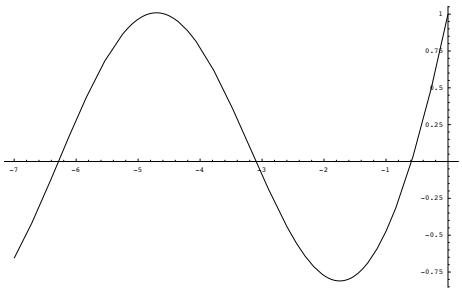
```
In[25]:=
      puntoscriticos=NSolve[f'[x]==0,x]
Out[25]=
      {{x->-2.60082},{x->-1.09686},{x->0.533861},{x->2.36382}}
In[26]:=
      f''[x]/.puntoscriticos
Out[26]=
      {-117.028,42.4372,-46.7717,157.202}
```

Como puedes ver, la segunda derivada nos confirma que tenemos alternativamente máximo relativo, mínimo relativo, máximo y mínimo relativo.

No todas las funciones permiten calcular de un modo simple las soluciones de la ecuación $f'(x) = 0$. En ese caso sólo nos queda recurrir a buscar uno a uno los posibles puntos críticos.

Ejemplo 3.2. Consideremos ahora la función $f(x) = \text{sen}(x) + e^x$ en el intervalo $[-7, 0]$. Pintémosla igual que antes, sustituyendo la antigua función por la nueva.

```
In[27]:=
      Clear[f,x];f[x_]=Exp[x]+Sin[x]
Out[27]=
      Exp[x]+Sin[x]
In[28]:=
      Plot[f[x],{x,-7,0}
```



A simple vista se observa que hay:
a) un máximo relativo cerca de -5,
b) un mínimo relativo cerca de -2
Sin embargo, ahora no funciona satisfactoriamente el comando `NSolve` (inténtalo). Optaremos por aplicar el comando `FindRoot` para encontrar las raíces de la derivada de f. Recordemos que en el comando `FindRoot` es necesario introducir una "primera aproximación" de la solución, que servirá para iniciar el proceso iterativo. En este caso concreto, dada la situación aproximada de los extremos, probaremos a iniciar el método desde -5 y también desde -2. Concretamente, escribiremos el comando

```
In[29]:=
      FindRoot[f'[x],{x,-5}]
Out[29]=
      {x->-4.70332}
In[30]:=
      FindRoot[f'[x],{x,-2}]
Out[30]=
      {x->-1.74614}
```

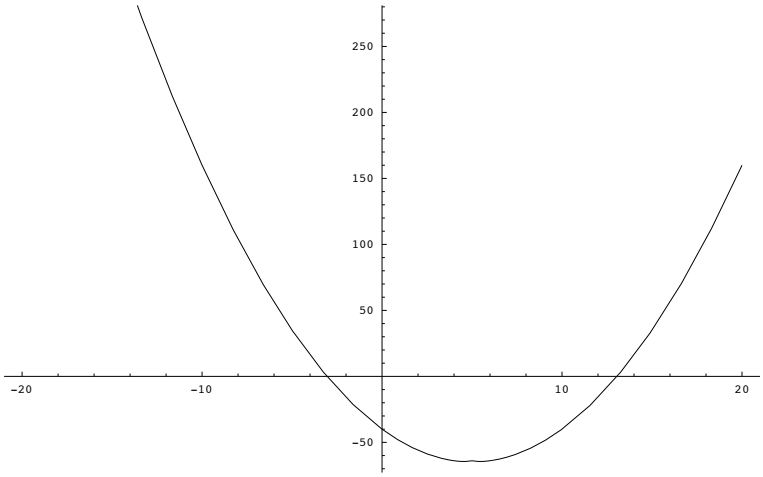
El programa nos da entonces una buena aproximación de los puntos donde la función parece que alcanza los extremos observados. Podemos asegurarnos mirando el signo de la segunda derivada

```
In[31]:=
      f''[-4.70332]
Out[31]=
      -0.990894
```

y

```
In[32]:=
      f''[-1.74614]
Out[32]=
      1.15911
```

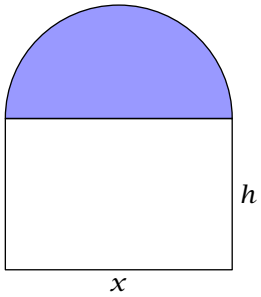
Ejemplo 3.3. En este ejemplo veremos un caso en el que la gráfica diseñada por el programa nos puede llevar a engaño. Descubriremos el error gracias al "test de la segunda derivada". Sea $f(x) = x^2 - 10x - 40 + \frac{1}{10x^2-100x+251}$, y pintémosla en el intervalo $[-20, 20]$. Su aspecto es algo así:



Aparentemente, hay un mínimo cerca de 5. Si buscamos raíces de la derivada ejecutando el comando `FindRoot` con dato inicial 5, obtenemos que precisamente para $x = 5$ la derivada se anula (se podía haber comprobado antes pidiendo al *Mathematica* el valor de $f'(5)$). A la vista de la gráfica, en $x = 5$ parece haber un mínimo de la función f. Sin embargo, pidámosle al *Mathematica* que calcule el valor de la segunda derivada de f en 5. Obtendremos que $f''(5) = -18 < 0$. La segunda derivada es negativa y por tanto en $x = 5$ tiene que haber un máximo.
Sería conveniente ver "más de cerca" lo que ocurre cerca de 5. Prueba a pintar la función en el intervalo $[4, 6]$. Ahora parece haber un mínimo cerca de 4.5 y otro cerca de 5.5, y efectivamente, un máximo en 5... ¿cómo podemos asegurar que cerca de los extremos que ahora vemos no vuelven a ocurrir nuevas "oscilaciones ocultas" como la que antes no veíamos?...

La capacidad de *Mathematica* de trabajar con variables de manera simbólica es útil para plantear y resolver problemas de optimización. El siguiente ejemplo es una muestra.

Ejemplo 3.4.



En la relación de problemas de clase tienes el siguiente ejercicio: se desea construir una ventana con forma de rectángulo coronado de un semicírculo de diámetro igual a la base del rectángulo. Pondremos cristal blanco en la parte rectangular y cristal de color en el semicírculo. Sabiendo que el cristal coloreado deja pasar la mitad de luz (por unidad de superficie) que el blanco, calcular las dimensiones para conseguir la máxima luminosidad si se ha de mantener el perímetro constante dado A.
Si llamamos x a la longitud de la base y h a su altura, el perímetro viene dado por

$$x + 2h + \frac{\pi x}{2} = A.$$

Utilizamos esta ecuación para relacionar las variables x y h. Por ejemplo, podemos despejar h en función de x:

```
In[33]:=
      altura=Solve[x+2h+Pi*x/2==A,h]
Out[33]=
      {h->1/2(A-x-π x/2)}
```

La función que queremos maximizar es la luminosidad de la ventana que viene dada por

$$f(x) = 2xh + \frac{\pi x^2}{8} = x\left(A - x - \pi\frac{x}{2}\right) + \pi\frac{x^2}{8} = Ax - \frac{1}{8}(8+3\pi)x^2.$$

Para terminar, calculamos puntos críticos y evaluamos la segunda derivada de la forma usual:

```
In[34]:=
      Clear[f]
      f[x_]=A*x-1/8(8+3Pi)x^2
      puntos=Solve[f'[x]==0,x]
      f''[x]/.puntos
```

para obtener que la máxima luminosidad se alcanza cuando $x = \frac{4A}{8+3\pi}$.

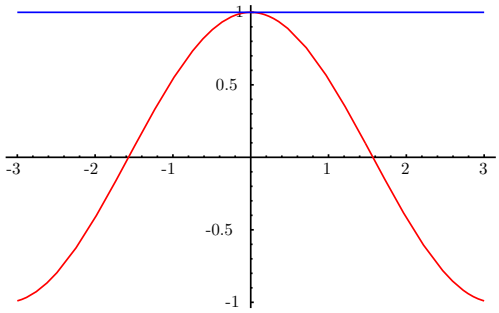
3.5 Polinomio de Taylor

Seguro que alguna vez te has preguntado cómo pueden una calculadora o un ordenador usar las funciones seno, coseno, exponencial, logaritmo... que no tienen una definición fácil, y que se basan en números "esencialmente" irracionales como π o e.

En esta práctica trataremos de entender cómo se pueden construir funciones conociendo únicamente las funciones suma (resta), producto (división) y potenciación. Necesitaremos además comprender el concepto de *límite* y la derivación de funciones. Se trata de "sustituir" funciones

complicadas por otras mas sencillas de calcular (que en este caso serán polinomios), controlando la "pérdida" que se produce.

El primer ejemplo puede ser "sustituir" una función por su recta tangente (si recuerdas, esto nos sirvió para encontrar ceros de funciones mediante el método de Newton-Raphson): dada una función f derivable en un punto a, la recta $y = f(a) + f'(a)(x-a)$ pasa por el punto $(a, f(a))$ y tiene la misma pendiente que la función f. En cierto sentido, lo que hemos hecho es encontrar una función más sencilla (una recta, o sea, un polinomio de grado uno) que se parece a la función de partida (coinciden el valor de la función y el de la derivada). ¿Hasta que punto puedo "cambiar" una función por su recta tangente? Si la función oscila demasiado, la aproximación no será buena, como por ejemplo en la función coseno.



En el gráfico podemos ver que en cuanto nos alejamos un poco del punto de tangencia (en este caso el 0), la función coseno y su tangente no se parecen nada. La forma de mejorar la aproximación será aumentar el grado del polinomio que usamos, y el problema es, fijado un grado, qué polinomio de grado menor o igual al fijado es el que más se parece a la función. El criterio con el que elegiremos el polinomio será hacer coincidir las sucesivas derivadas. Por ejemplo, si buscamos un polinomio P de grado dos que aproxime a una función dos veces derivable, f, vamos a exigir que $P(a) = f(a)$, $P'(a) = f'(a)$ y $P''(a) = f''(a)$. Con estas tres condiciones estamos en disposición de calcular los coeficientes de $A(x-a)^2 + B(x-a) + C$. Si lo hacemos, veremos que dicho polinomio "aproximante" de grado 2 es el siguiente (deriva el polinomio P y verás como sus derivadas hasta orden 2 coinciden con las de la función f)

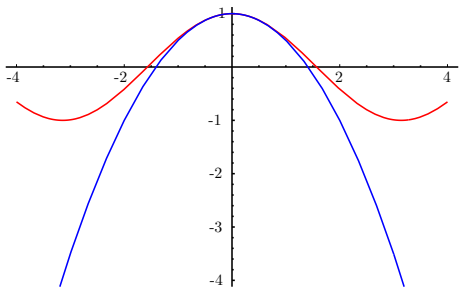
$$P(x) = f(a) + f'(a)(x-a) + \frac{f''(a)}{2!}\,(x-a)^2.$$

Veamos qué pasa con una función concreta. Consideremos la función $f(x) = \cos(x)$ y vamos a calcular el polinomio "aproximante" en 0. En este caso $f(0) = 1,\ f'(0) = 0,\ f''(0) = -1$. Por tanto el polinomio "de aproximación" de grado 2 es el siguiente:

$$P_2(x) = f(0) + f'(0)\,x + \frac{f''(0)}{2!}\,x^2 = 1 - \frac{1}{2}\,x^2.$$

Si dibujamos ahora las funciones f, P_1 y P_2 podremos comprobar qué es lo que ocurre:

```
In[35]:=
      Clear["Global`*"]
      f[x_]=Cos[x];
      Plot[{f[x],1-(x^2)/2},{x,-4,4},
        PlotStyle->{RGBColor[1,0,0],RGBColor[0,0,1]}]
```




```
Out[35]=  - Graphics -
```

El polinomio de orden 2 se parece un poco más a la función coseno que la recta tangente, pero de todas formas, si aumentamos el dominio (prueba por ejemplo $x \in [-10, 10]$) se puede ver que el "parecido" se pierde al alejarnos del origen. El problema es que estamos calculando polinomios de grado bajo.

Todo lo que hemos hecho hasta ahora se basa en un resultado teórico ya conocido: si I es un intervalo, $a, x \in I$ y $f : I \to \mathbb{R}$ es una función n-veces derivable, se define el *polinomio de Taylor* de orden n en el punto a como

$$P_n(x) = f(a) + f'(a)\ (x-a) + \frac{f''(a)}{2!}\ (x-a)^2 + \cdots + \frac{f^{(n)}(a)}{n!}\ (x-a)^n$$

Fórmula de Taylor

Teorema 3.5. *Si $f : I \to \mathbb{R}$ es una función $(n+1)$-veces derivable, entonces existe c entre a y x tal que*

$$f(x) - P_n(x) = \frac{f^{(n+1)}(c)}{(n+1)!}\ (x-a)^{n+1}.$$

El teorema anterior nos permite incluso acotar el error que estamos cometiendo: basta acotar el llamado *resto de Taylor*, $\frac{f^{(n+1)}(c)}{(n+1)!}$, y para ello, acotaremos la derivada de orden $n+1$ en el intervalo considerado. Volvamos al ejemplo anterior: Si trabajamos en el intervalo $[-\pi, \pi]$ y queremos usar P_2 en lugar de la función coseno, sabemos que el punto c que nos da el error también está en ese intervalo. Vamos a calcular el valor máximo que puede alcanzar:

$$|\cos(x) - P_2(x)| = \left|\frac{f'''(c)}{3!}\ x^3\right| = \left|\frac{\operatorname{sen}(c)}{3!}\ x^3\right| \leq \frac{1}{6}\pi^3.$$

La última acotación se basa en que el valor máximo que alcanza la tercera derivada, $\operatorname{sen}(x)$, es 1, y que x en valor absoluto vale a lo sumo π.

Mathematica tiene una orden que permite calcular directamente el polinomio de Taylor centrado en un punto a. Se trata del comando `Series`, cuya sintaxis es

`Series[`*`funcion`*`,{x,a,n}]`	polinomio de Taylor de la función centrado en a de grado n

Volviendo al ejemplo anterior, si pruebas

```
In[36]:=
      Series[Cos[x],{x,0,2}]
Out[36]=
      1 - x^2/2 + O[x]^3
```

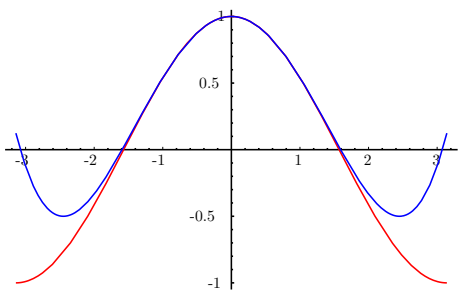
verás como, además del desarrollo del polinomio, aparece al final el orden de error ($O[x]^3$); representa que los términos que faltan son todos de grado mayor o igual que tres. Si quieres usar directamente el resultado (sin que aparezca el orden del error) puedes utilizar el comando

`Normal[`*`polinomio de Taylor con resto`*`]`	polinomio de Taylor *sin* resto

Este comando simplemente elimina el orden de error de la expresión que obtenemos con el comando `Series`, quedando sólo el polinomio de Taylor.

En nuestro ejemplo, para comparar la función coseno y el correspondiente polinomio de Taylor de grado 4 en cero haremos lo siguiente:

```
In[37]:=
      Clear[``Global`*''];
      f[x_]=Cos[x];
      g[x_]=Normal[Series[f[x],{x,0,4}]];
      Plot[{f[x],g[x]},{x,-Pi,Pi},
        PlotStyle->{RGBColor[1,0,0],RGBColor[0,0,1]}]
```




```
Out[37]=  - Graphics -
```

Recordemos que con `Animate` o `Table` somos capaces de representar funciones que dependen de un parámetro. En el ejemplo siguiente, aprovechamos esto para dibujar una función y polinomios de Taylor de varios grados.

```
In[38]:=
      Clear[``Global`*''];
      f[x_]=Cos[x]+Sin[x];
      a=0;
      b=-2Pi;
      c=2Pi;
      orden=30;
      Animate[Plot[Evaluate[
         {f[x],Normal[Series[f[x],{x,a,n}]]},
         {x,b,c}, PlotStyle->{RGBColor[1,0,0],
          RGBColor[0,0,1]},PlotRange->{-3,3}]],
       {n,1,orden,1}]
```

Cuando se ejecuta la orden anterior, *Mathematica* dibuja la función y los polinomios de Taylor de orden 1 hasta 30.

Después de esto el primer ejercicio es obligado.

Ejercicio 3.1

Compara la gráfica de las funciones elementales y sus polinomios de Taylor. ¿Cuáles de ellas coinciden con su polinomio? ¿Qué ocurre si cambias el punto a donde centramos el polinomio de Taylor?

Ejercicio 3.2

Haz una animación gráfica en la que se representen en rojo la función seno y en azul sus polinomios de Taylor de orden n centrados en 0, donde n toma valores desde 1 a 21 con incrementos de 2. Haz la representación en el intervalo $[-2\pi, 2\pi]$ y en el $[-8\pi, 8\pi]$. Debes fijar, con "PlotRange[ ]", un intervalo de representación común para todas las gráficas de la animación.Comenta lo que observas. Repite la misma operación pero centrando el desarrollo en $a = 2\pi$. ¿Cuáles son las diferencias?

Ejercicio 3.3

Haz una animación gráfica en la que se representen en rojo la función arco tangente y en azul sus polinomios de Taylor de orden n centrados en 0, donde n toma valores desde 1 a 21 con incrementos de 2. Haz la representación en el intervalo $[-1, 1]$ y en $[-\pi/2, \pi/2]$. Debes fijar, con "PlotRange[ ]", un intervalo de representación común para todas las gráficas de la animación. Comenta lo que observas. Repite la misma operación pero centrando el desarrollo en $a = 2$. ¿Cuáles son las diferencias?

Ejercicio 3.4

Aproxima la función exponencial ($f(x) = e^x$) mediante su polinomio de Taylor de grado 4 centrado en 0 en el intervalo $[-20, 20]$. Prueba a aumentar el grado y a usar intervalos cada vez más grandes.

Ejercicio 3.5

a) Define una función cuya sintaxis debe ser de la forma "rectanormal[f,a][x]", cuyo resultado sea la recta normal a la gráfica de la función f en el punto $(a, f(a))$. Debes definir una función condicionada según que $f'(a) \neq 0$ o que $f'(a) = 0$.

b) Representa gráficamente (en una sola gráfica y en azul) la familia de rectas normales a la parábola $y = x^2/4$. En la misma gráfica representa también la parábola en rojo. Haz la representación para $x \in [-6, 6]$. Representa las rectas normales en los puntos $(x_k, f(x_k))$ donde x_k son 101 puntos igualmente espaciados en el intervalo [-6,6].

c) Sobre la gráfica anterior representa, en rojo la gráfica de la curva $y = 2 + \sqrt[3]{\frac{27x^2}{4}}$ y comprueba que las rectas normales a la parábola son tangentes a dicha curva que se llama la *evoluta* de la parábola.

Integración

4

4.1 Integrales definidas e indefinidas

En esta práctica aprenderemos a calcular integrales con el programa *Mathematica* tanto definidas como indefinidas. El objetivo es poder calcular áreas entre curvas, áreas y volúmenes de sólidos de revolución, longitudes de curvas,etc. sin que el cálculo de primitivas sea un escollo.

Hay dos comandos básicos para calcular integrales: `Integrate` calcula la integral de manera exacta e `NIntegrate` la aproxima numéricamente. También se puede usar la paleta para escribir las integrales.

Figura 4.1
Paleta

`Integrate[`*`función`*`,`*`variable`*`]`	Primitiva de la función
`Integrate[`*`función`*`,{`*`variable`*`,a,b}]`	Integral de la función en $[a,b]$
`NIntegrate[`*`función`*`,{`*`variable`*`,a,b}]`	Aproximación númérica de la integral de la función en $[a,b]$

La primera de las órdenes mencionadas calcula primitivas o integrales definidas en un intervalo $[a,b]$ dado. Por ejemplo,

```
In[1]:=
      Integrate[Cos[x],x]
Out[1]=
      Sin[x]
In[2]:=
      ∫Cos[x]dx
Out[2]=
      Sin[x]
```

Parámetros

Para calcular una integral definida que depende de un parámetro puedes informar a *Mathematica* de las carcaterísticas de dicho parámetro. Si no lo haces *Mathematica* intentará dar una respuesta válida imponiendo, si es necesario, algunas restricciones al parámetro. Cuando se calculan primitivas Mathematica no se preocupa de valores particulares de los parámetros. Veamos un ejemplo

```
In[3]:=
      Integrate[Exp[-a*x],x]
Out[3]=
      -E^-ax/a
```

Si la integral es definida, es posible que *Mathematica* imponga algunas restricciones sobre el parámetro para dar una respuesta

```
In[4]:=
      Integrate[Exp[-a*x],{x,0,Infinity}]
Out[4]=
      If[Re[a]>0, 1/a, ∫_0^∞ E^-ax dx]
```

o podemos añadir nosotros las restricciones que deseemos

```
In[5]:=
      Integrate[Exp[-a*x],{x,0,Infinity},Assumptions->{a>0}]
Out[5]=
      1/a
```

En la primera integral, *Mathematica* se ha limitado a calcular una primitiva. En la segunda, calcula la integral si el parámetro a tiene parte real positiva (conviene recordar que para *Mathematica* todas las variables son por defecto variables complejas). En la tercera nos da el valor de la integral cuando $a > 0$, lo que incluye implícitamente que la parte real es positiva.

Nuevas funciones

Mathematica puede obtener prácticamente cualquier primitiva calculable por métodos elementales (integración por partes, integración de funciones racionales, trigonométricas, cambio de variable...). Por otro lado, sabemos que toda función continua admite primitiva, pero esta primitiva no siempre puede calcularse (en el sentido de tener una expresión de la primitiva en términos de funciones "elementales"). Por eso, al integrar algunas funciones en el resultado que obtenemos aparecen funciones que no conocemos:

```
In[6]:=
      ∫ Sin[x]/x dx
Out[6]=
      SinIntegral[x]
```

⚠ Busca en la ayuda cuál es la definición de la función `SinIntegral` (seno integral).

Aunque no conozcamos la expresión de la función que nos aparece, podemos utilizarla como cualquier otra función y calcular su valor en un punto

```
In[7]:=
      N[SinIntegral[2],7]
Out[7]=
      1.605413
```

o dibujarla, como cualquier otra función.

4.1.1 Aproximación númerica

La definición de la integral de una función se presta muy bien a su cálculo con un ordenador. Es por ello que, en el caso *Mathematica* no sepa calcular la primitiva de una función

```
In[8]:=
      ∫Exp[x^3 + x]dx
Out[8]=
      ∫E^(x^3+x) dx
```

siempre podemos intentar aproximar el valor de la integral. Esto es precisamente lo que hace el comando `NIntegrate`.

```
In[9]:=
      NIntegrate[Exp[x^3+x],{x,0,2}]
Out[9]=
      1846.36
```

Observa que no es lo mismo usar `NIntegrate[...]` que `N[Integrate[...]]`. En el segundo caso *Mathematica* tendría que saber calcular una primitiva para después calcular cuanto vale, mientras que el primer comando intenta aproximar el valor de la integral sin calcular una primitiva. Recordemos que el comando `N[`*`expresión`*`]` nos daba una aproximación numérica de la expresión. En el caso de integrales, va un poco más allá. `N[Integrate[`*`función`*`,x,a,b]]` nos da el valor aproximado de la integral calculada de forma exacta y, si no sabe calcularla como en este caso, aproxima la integral.

Curiosamente, con funciones a las que podemos calcular fácilmente una primitiva, el comando `NIntegrate` no siempre funciona bien; tecleando

```
In[10]:=
      NIntegrate[Cos[x],{x,0,Pi}]
```

te darás cuenta que *Mathematica* "protesta". El programa intenta calcular una aproximación con métodos numéricos; estos métodos pueden tener problemas cuando la función oscila mucho. En estos casos pueden aparecer respuestas erróneas.

Ejercicio 4.1

Calcula numéricamente las integrales en el intervalo $[-2,2]$ de los primeros 20 polinomios de Taylor centrados en 0 de la función e^{-x^2}. Calcula también numéricamente la integral de dicha función en el mismo intervalo.

Ejercicio 4.2

Escribe un programa que calcule las sumas de Riemann de una función f en un intervalo $[a,b]$.

4.2 Integrales impropias.

Como recordarás de las clases teóricas, la integral que en principio se define para funciones continuas en intervalos cerrados y acotados, puede extenderse a funciones continuas definidas en intervalos de longitud infinita, y a funciones que no están acotadas en un intervalo de longitud finita. Es lo que se conoce como *Integración impropia*. Con el programa *Mathematica* , el trabajar con integrales impropias no supone ningún problema, ya que las trata exactamente igual que las integrales de funciones continuas en intervalos cerrados y acotados.

Con la misma orden `Integrate` se pueden calcular integrales impropias. Simplemente prueba con una función como $f(x) = \frac{1}{\sqrt{1-x^2}}$ en el intervalo $[-1,1]$ o con la función $g(x) = e^{-x^2}$ en $[0,+\infty[$. En este segundo caso, escribiríamos

```
In[11]:=
        ∫_0^+∞ Exp[-x^2]dx
Out[11]=
        √π
        ──
        2
```

Intenta calcular las integrales anteriores en el intervalo dado y después intenta calcular una primitiva de esas funciones. Como verás, la primitiva de la primera función ya la conocías, en cambio en la primitiva de la segunda aparece una función que, supongo, no conocías.

¿Cuanto vale $\int_1^\infty \frac{1}{x}\,dx$? Compruébalo y verás cómo responde *Mathematica* cuando una función no es impropiamente integrable:

```
In[12]:=
        Integrate[1/x dx,{x,1,Infinity}]
Out[12]=
        Integrate::idiv :
        Integral of 1/x does not converge on {1,∞}.
```

4.3 Longitudes, áreas y volúmenes

Algunas de las aplicaciones de la integral que se han visto en las clases teóricas son calcular longitudes de curvas, y volúmenes y superficies de cuerpos de revolución.

4.3.1 Cálculo de áreas planas

Si f y g son dos funciones integrables definidas en un intervalo $[a,b]$, el área entre las dos funciones es

$$\int_a^b |f(x)-g(x)|\,dx.$$

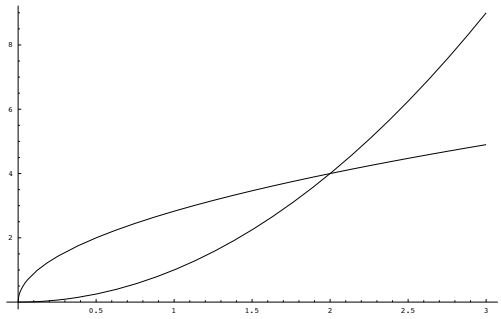
Esta integral no es fácil de resolver directamente por la presencia del valor absoluto que dificulta el cálculo de una primitiva. La solución a este problema es la misma que aplicamos siempre: quitemos el valor absoluto. Para esto tenemos que dividir el intervalo $[a,b]$ en subintervalos en los que sepamos cuál de las dos funciones es la mayor.

Por ejemplo, para calcular el área entre las curvas $y=x^2$ e $y^2=8x$ necesitamos saber cuál es mayor. En primer lugar, veamos cuándo coinciden

```
In[13]:=
      Solve[{y==x^2,y^2==8x},{x,y}]
Out[13]=
      {{x->0,y->0},{x->2,y->4}}
```

Puedes echarle un vistazo a la gráfica de las funciones para ver cuál es mayor

```
In[14]:=
      Plot[{x^2,√8x},{x,-1,3}]
```




```
Out[14]=  - Graphics -
```

o, simplemente, evaluar en un punto y ver cuál vale más: $1^2=1$ o $\sqrt{8}$. Visto que $y=\sqrt{8x}$ es mayor, el área pedida es

```
In[15]:=
      Integrate[√8x-x^2,{x,0,2}]
Out[15]=
      8
      3
```

Ejercicio 4.3

Calcular:

a) área limitada por $y = xe^{-x^2}$, el eje OX, la ordenada en el punto $x = 0$ y la ordenada en el máximo.
b) área de la figura limitada por la curva $y = x^3 - x^2$ y el eje OX.
c) área comprendida entre la curva $y = \tan(x)$, el eje OX y la recta $x = \pi/3$.
d) área del recinto limitado por las rectas $x = 0$, $x = 1$, $y = 0$ y la gráfica de la función $f : \mathbb{R} \to \mathbb{R}$ definida por $f(x) = \frac{1}{(1+x^2)^2}$.

4.3.2 Longitudes de curvas

Sea f una función de clase 1 en el intervalo $[a, b]$. La longitud del arco de la curva $y = f(x)$ entre $x = a$ y $x = b$ es

$$l = \int_a^b \sqrt{1 + f'(x)^2}\, dx.$$

Ejemplo 4.1. La circunferencia de radio 1 (centrada en el origen) está formada por los puntos

$$\left\{(x, y) \in \mathbb{R}^2 : \ x^2 + y^2 = 1\right\}.$$

Despejando, la semicircunferencia superior es la imagen de la función $f(x) = \sqrt{1 - x^2}$ con $x \in [-1, 1]$. Su longitud es

```
In[16]:=
      f[x_]=Sqrt[1-x^2]
Out[16]=
      √1-x^2
In[17]:=
      Integrate[Sqrt[1+f'[x]^2],{x,-1,1}]
Out[17]=
      π
```

y la longitud de la circunferencia completa es, obviamente, 2π.

Esta fórmula es un caso particular de la longitud de una curva en el plano. Una curva es una aplicación de $[a, b]$ en $\mathbb{R}^2$ de la forma

$$x \mapsto (f(x), g(x)), \ x \in [a, b]$$

y su longitud es

$$\ell = \int_a^b \sqrt{f'(x)^2 + g'(x)^2}\, dx.$$

Por ejemplo, la curva $x \mapsto (\cos(x), \operatorname{sen}(x))$ con $x \in [0, 2\pi]$ es una parametrización de la circunferencia unidad. Su longitud será

```
In[18]:=
        ∫_0^2Pi √(Sin[x]^2+Cos[x]^2) dx
Out[18]=
        2π
```

Ejercicio 4.4

Calcular la longitud de las siguientes curvas:

a) $y = \frac{1}{3}(x^2+2)^{3/2}$, $x \in [0,1]$,

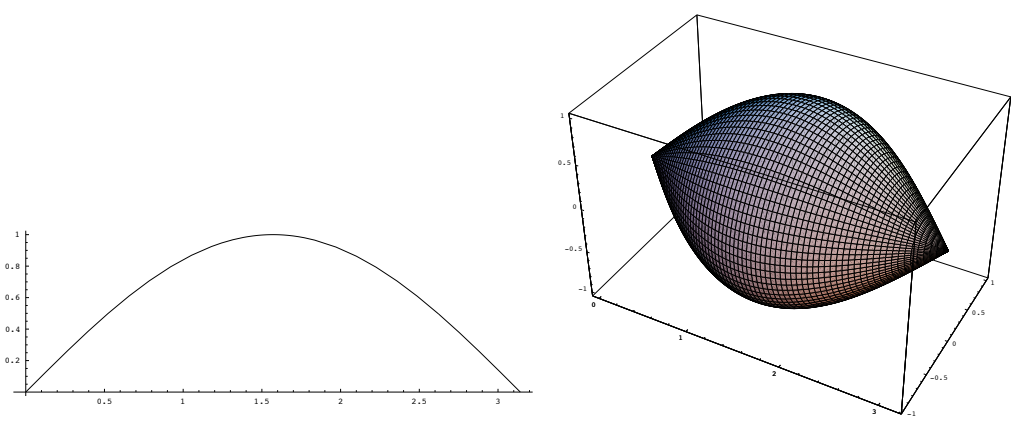
b) $y = \frac{1}{2}(e^x + e^{-x})$, $x \in [0,3]$,

c) $y = \ln\left(1 - x^2\right)$, con $\frac{1}{3} \le x \le \frac{2}{3}$,

d) $\rho = 3(1+\cos(\theta))$, $\theta \in [0, 2\pi]$.

4.3.3 Volumen de sólidos de revolución

Los cuerpos de revolución o sólidos de revolución son regiones de $\mathbb{R}^3$ que se obtienen girando una región plana alrededor de una recta llamada eje de giro. Aquí tienes el sólido de revolución engendrado al girar alrededor del eje OX la gráfica de la función seno entre 0 y π:



Función seno Función seno girada

Figura 4.2

Método de los discos: secciones perpendiculares al eje de giro

Sea $f : [a,b] \to \mathbb{R}$ una función continua; el volumen del sólido generado al girar el área bajo la curva $y = f(x)$ respecto del eje OX es

$$V = \pi \int_a^b f(x)^2\,dx$$

Ejemplo 4.2. Como primera aplicación vamos a calcular el volumen de una esfera de radio R. Esta esfera la podemos obtener girando la circuferencia $x^2 + y^2 = R^2$ o, lo que es lo mismo, la función $f(x) = \sqrt{R^2 - x^2}$ entre $-R$ y R.

```
In[19]:=
        Pi ∫_-R^R (R^2-x^2)dx
Out[19]=
        4πR^3
        ─────
          3
```

De la misma forma podemos calcular el volumen de un cono recto de altura h y radio R que se obtiene girando la recta $y = \frac{Rx}{h}$ entre 0 y h. Su volumen es

```
In[20]:=
        Pi∫_0^h (Rx/h)^2dx
Out[20]=
        πhR^2
        ─────
          3
```

Ejercicio 4.5

a) Calcular el volumen del sólido engendrado al girar alrededor del eje OX la parte de la curva $y = \text{sen}^2(x)$ comprendida entre 0 y π.

b) Calcular el volumen del sólido engendrado al girar alrededor del eje OX la función $f(x) = \frac{18}{x^2+9}$ en $[0, +\infty[$.

c) Calcular el volumen del sólido engendrado al girar alrededor del eje OX la región limitada por la parábolas $y^2 = x$, $x^2 = y$.

Método de las láminas o de los los tubos

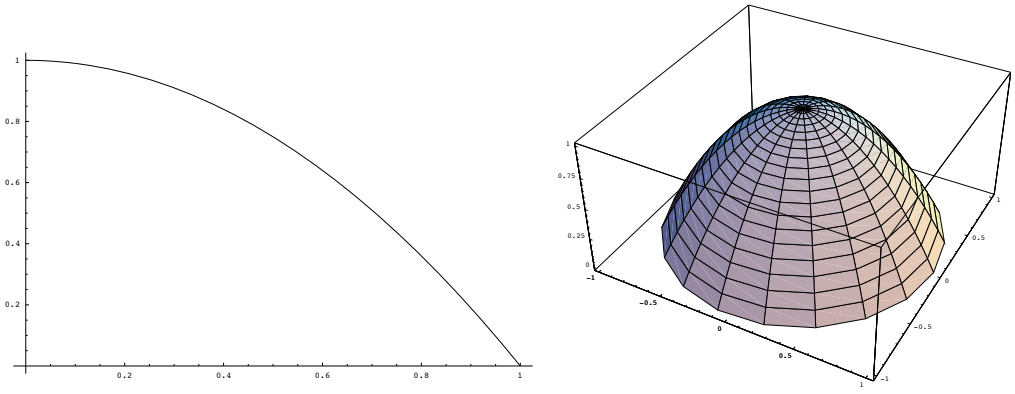
Si giramos respecto del eje OY la función $1 - x^2$ en el intervalo $[0, 1]$ obtenemos lo que puedes ver en la siguiente figura.

¿Cuál es su volumen?

Sea $f : [a, b] \to \mathbb{R}$ una función continua y *positiva*. El volumen del sólido generado al girar el área bajo la curva $y = f(x)$ respecto del eje OY es

$$V = 2\pi \int_a^b x f(x)\,dx.$$

Ejemplo 4.3. El *toro* es la superficie que se obtiene al girar una circunferencia de centro $(a, 0)$ y radio R (con $a > R$) alrededor del eje OY. En la figura siguiente tienes el aspecto que tiene para $a = 2$ y $R = 1$.



Función $1 - x^2$ Función $1 - x^2$ girada

Figura 4.3

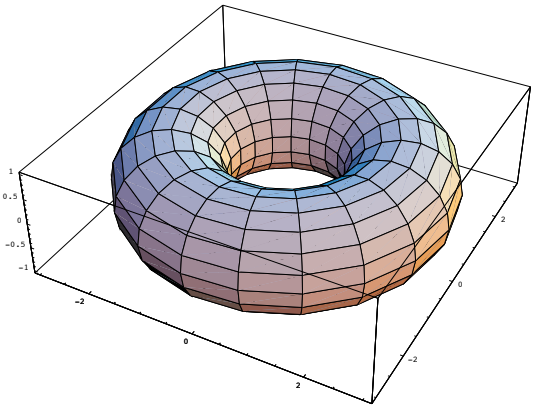


Figura 4.4 Toro

¿Cuál es su volumen? La mitad superior del toro se obtiene girando la función $y = \sqrt{R^2 - (x-a)^2}$ con $x \in [a-R, a+R]$. Utiliza *Mathematica* para comprobar que su volumen es $2a\pi^2R^2$.

Ejercicio 4.6

a) Calcular el volumen del toro engendrado al girar el círculo de centro $(2,0)$ y radio 3 alrededor del eje OY.
b) Calcular el volumen del sólido engendrado al girar alrededor del eje OY la región limitada por la parábolas $y^2 = x$, $x^2 = y$.

4.3.4 Área de sólidos de revolución

Igual que hemos visto cómo podemos calcular el volumen de una figura obtenida girando una función respecto a alguno de los ejes, también podemos calcular el área de la figura obtenida al girar respecto al eje OX una función.

Sea $f : [a,b] \to \mathbb{R}$ una función de clase 1; el área de la superficie generada haciendo girar alrededor del eje OX el arco de curva $y = f(x)$ en $[a,b]$ es

$$\text{Área} = 2\pi \int_a^b f(x)\sqrt{1 + f'(x)^2}\, dx.$$

Ejercicio 4.7

Calcular el volumen del sólido generado al rotar respecto al eje OX las siguiente curvas:

a) $y = \sec(x),\ x \in [0, \frac{\pi}{3}]$
b) $y = \sqrt{\cos(x)},\ x \in [0, \frac{\pi}{2}]$
c) $y = 9 - x^2$
d) $y = e^x,\ x \in [0, \ln(3)]$

Ejercicio 4.8

Calcular el volumen del sólido generado al rotar respecto al eje OY las siguiente curvas:

a) $y = 1/x,\ x \in [1, 3]$
b) $y = \frac{1}{1+x^2},\ x \in [0, 1]$
c) $y = e^{x^2},\ x \in [1, \sqrt{3}]$

Ejercicio 4.9

a) Calcular la superficie de una esfera de radio R.
b) Calcular la superficie de la figura que se obtiene al girar la función $y = \tan(x),\ x \in [0, \pi/4]$ alrededor del eje OX.

Ejercicio 4.10

Calcular:
a) La integral de $f(x) = \frac{1}{x^2}$ con $x \in [1, +\infty]$.
b) El volumen y la superficie lateral del sólido obtenido al girar la gráfica de la anterior función respecto del eje OX.
c) Ídem a los dos anteriores con $g(x) = \frac{1}{x}$ con $x \in [1, +\infty]$.

Gráficos en 3D

5

5.1 El comando Plot3D

En el segundo capítulo pudimos comprobar que *Mathematica* es una herramienta muy potente para "dibujar" gráficas de funciones de una variable. Ahora aprenderemos a representar gráficamente funciones reales de dos variables. La principal aplicación de la representación gráfica de una función de dos variables será dar una idea aproximada de la variación de dicha función, lo que será especialmente útil para buscar extremos.

Al igual que para funciones de una variable, suele ser cómodo asignarle un nombre (normalmente f, g, h...). Para definir la función $\text{sen}(xy)$ haremos lo siguiente:

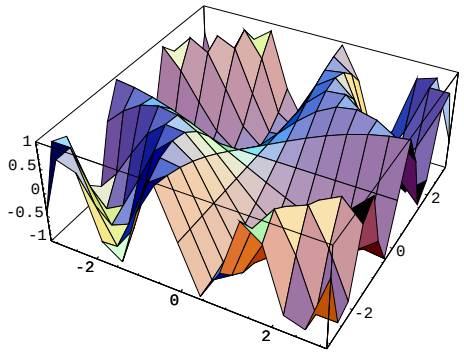
```
In[1]:=
      f[x_,y_]=Sin[x*y]
Out[1]=
      Sin[x y]
```

y usaremos f de igual forma a como lo hacíamos con funciones de una variable. El comando que sirve para representar gráficas de funciones de dos variables es

`Plot3D[`*función de* `x,y,{x,a,b},{y,c,d}]`	Gráfica de la función en $[a,b]\times[c,d]$

Veamos un ejemplo:

```
In[2]:=
      Plot3D[f[x,y],{x,-Pi,Pi},{y,-Pi,Pi}]
```



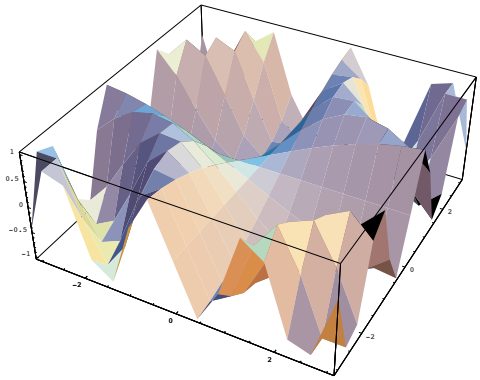

```
Out[2]=  - SurfaceGraphics -
```

5.1.1 Opciones del comando Plot3D

a) `Mesh->True/False`

Se dibuja (True) o no (False) la retícula o malla sobre la que se construye la gráfica. En el ejemplo anterior:

```
In[3]:=
      Plot3D[f[x,y],{x,-Pi,Pi},{y,-Pi,Pi},Mesh->False]
```

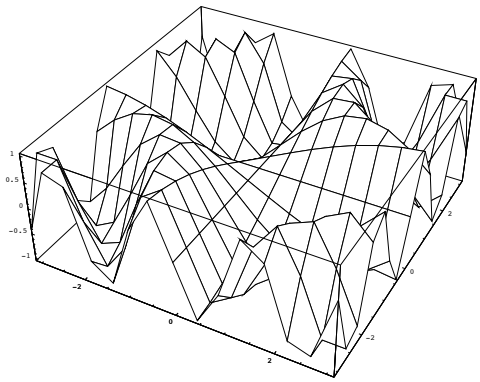


```
Out[3]=  - SurfaceGraphics -
```

b) `Shading->True/False`

Se colorea (True) o no (False) la malla anterior. Si unimos las opciones `Mesh->False` y `Shading->False`, no aparecerá gráfico alguno. Un ejemplo:

```
In[4]:=
      Plot3D[f[x,y],{x,-Pi,Pi},{y,-Pi,Pi},Shading->False]
```

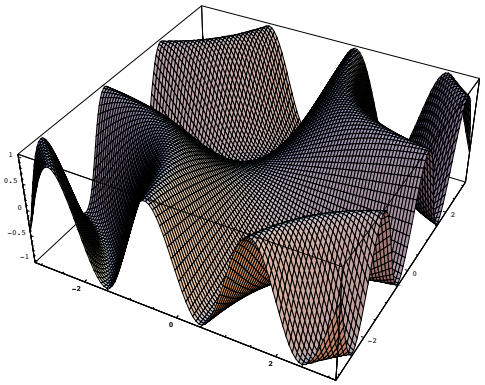


```
Out[4]=  - SurfaceGraphics -
```

c) `PlotPoints->`*`número de puntos`*

Representa el número de puntos que usará *Mathematica* para dibujar la gráfica. Un número muy alto producirá un gráfico más "suave", pero aumentará considerablemente el tiempo empleado por *Mathematica* para realizarlo: hay que tener en cuenta que `PlotPoints->100` indica a *Mathematica* que calcule la función en 100×100 puntos pero que 200 implica pasar a $200x200 = 40000$ puntos.

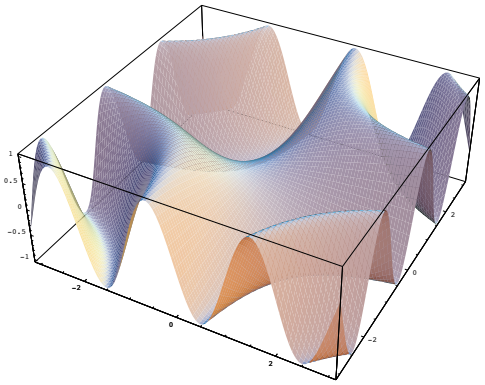
```
In[5]:=
      Plot3D[f[x,y],{x,-Pi,Pi},{y,-Pi,Pi},PlotPoints->100]
```



```
Out[5]=  - SurfaceGraphics -
```

Si el número de puntos es elevado, además del tiempo, hay que tener en cuenta que la malla puede tapar la gráfica de la función por lo que es conveniente eliminarla.

```
In[6]:=
      Plot3D[f[x,y],{x,-Pi,Pi},{y,-Pi,Pi},PlotPoints->100,Mesh->False]
```

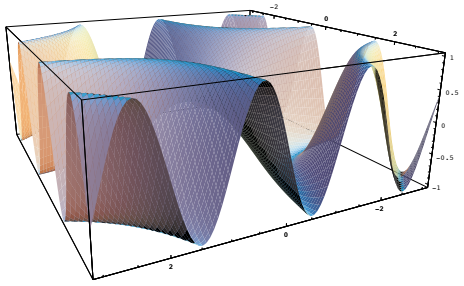


```
Out[6]=  - SurfaceGraphics -
```

d) `ViewPoint->{x,y,z}`

Establece el punto de vista desde el que se dibujará la gráfica. La cámara se sitúa en el punto de coordenadas (x,y,z). Por ejemplo `ViewPoint->{1,0,0}` indica a *Mathematica* que mire desde el eje OX y `ViewPoint->{0,0,1}` significa mirar desde arriba.

```
In[7]:=
      Plot3D[f[x,y],{x,-Pi,Pi},{y,-Pi,Pi},Mesh->False
        PlotPoints->100,ViewPoint->{0.75,1.5,0.5}]
```



```
Out[7]=  - SurfaceGraphics -
```

También existe la opción de seleccionar el punto de vista desde una ventana que permite elegir dicho punto de forma interactiva. A dicha ventana se accede a través del menú `Input`, submenú `3D ViewPoint Selector`. Una vez que tengas decidido el punto de vista mediante los deslizadores, "Paste' ' te lo pegará donde tengas el cursor.

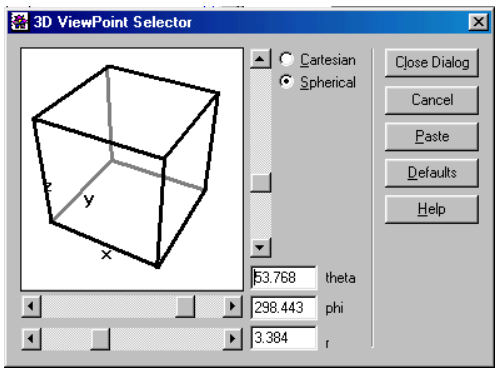


Figura 5.1 Punto de vista

Ejercicio 5.1

Representa la gráfica de las siguientes funciones.

a) $f(x,y) = (x^2+y^4)e^{1-x^2-y^2}$ con $x,y \in [-2,2]$.

b) $f(x,y) = \text{sen}(x-2xy)$ con $x \in [0,\pi]$, $y \in [-\pi,\pi]$.

c) $f(x,y) = \frac{xy}{x^2+y^2}$ con $x,y \in [-1,1]$.

d) $f(x,y) = \ln\left(1+x^2+y^2\right)$ con $x,y \in [-2,2]$.

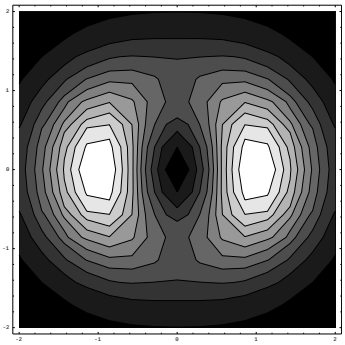
5.2 Gráficos de contorno. Curvas de nivel.

Una forma cómoda de ver la gráfica de una función es mediante curvas de nivel. Se trata de representar en el plano una figura tridimensional como es la gráfica de una función de dos variables, permitiendo ver las zonas de crecimiento y decrecimiento de dicha función. El comando que representa funciones de dos variables mediante curvas de nivel es

`ContourPlot[`*función de* `x,y,{x,a,b},{y,c,d}]`	curvas de nivel en $[a,b]\times[c,d]$

Por ejemplo,

```
In[8]:=
      ContourPlot[(3 x^2+ y^2)Exp[1-x^2-y^2], {x,-2,2},{y,-2,2}]
```

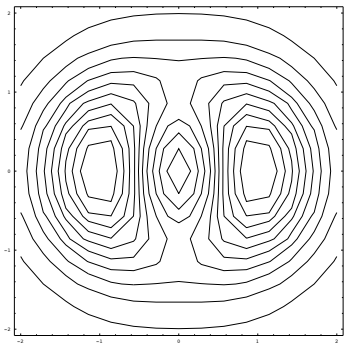


```
Out[8]=  - ContourGraphics -
```

Entre otras, las opciones del comando `ContourPlot` son:

a) `ContourShading->False`
Dibuja sólo las curvas de nivel y no utiliza la escala de grises.

```
In[9]:=
      ContourPlot[(3 x^2+ y^2)Exp[1-x^2-y^2],{x,-2,2},{y,-2,2},
        ContourShading->False]
```



```
Out[9]=  - ContourGraphics -
```

b) `PlotPoints->`*`número de puntos`*

Representa el número de puntos que usará *Mathematica* para dibujar la gráfica. Al igual que en `Plot3D`, un número muy alto producirá un gráfico más "suave", pero aumentará considerablemente el tiempo empleado por *Mathematica* para realizarlo.

Ejercicio 5.2

Representa los gráficos de contorno de las funciones del ejercicio anterior, comparando la información que obtienes de una y otra forma.

Ejercicio 5.3

Consideremos la función $f : [-2,2] \times [-2,2] \to \mathbb{R}$ dada por $f(x,y) = (x^2 + y^4)e^{1-x^2-y^2}$. Utiliza los comandos anteriores para obtener información sobre la ubicación de sus posibles extremos relativos.

5.3 Gráficos en coordenadas paramétricas

Ya aprendimos a representar gráficamente curvas planas expresadas en forma paramétrica. Ahora veremos cómo representar curvas alabeadas (esto es, curvas en el espacio) y superficies paramétricas. El comando de *Mathematica* para ello es `ParametricPlot3D`.

`ParametricPlot3D[{`$x(t), y(t), z(t)$`},{`t, a, b`}]`	gráfica de la curva
`ParametricPlot3D[{`$x(u,v), y(u,v), z(u,v)$`},{`u, a, b`},{`v, c, d`}]`	gráfica de la superficie

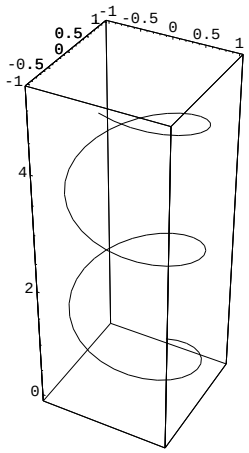
Cuando tenemos las ecuaciones paramétricas de una curva en el espacio

$$x = x(t),\ y = y(t),\ z = z(t),\ \text{ con } t \in [a,b]$$

usamos la primera de las órdenes anteriores: `ParametricPlot3D[{`$x(t), y(t), z(t)$`},{`t, a, b`}]`.

Por ejemplo, para dibujar una hélice:

```
In[10]:=
      ParametricPlot3D[{Sin[t],Cos[t],t/3},{t,0,5Pi}]
```



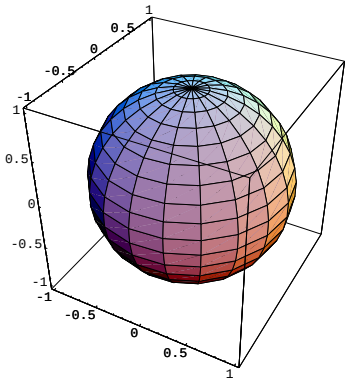
```
Out[10]=  - Graphics3D -
```

Si lo que tenemos son las ecuaciones paramétricas de una superficie

$$x = x(u,v),\ y = y(u,v),\ z = z(u,v) \quad \text{con } u \in [a,b] \text{ y } v \in [c,d]$$

se usa `ParametricPlot3D[{`$x(u,v), y(u,v), z(u,v)$`},{`u,a,b`},{`v,c,d`}]`. Dibujemos una esfera:

```
In[11]:=
      ParametricPlot3D[{Cos[u]Cos[v],Sin[u]Cos[v],Sin[v]},
        {u,0,2 Pi},{v,-Pi/2,Pi/2}]
```



```
Out[11]=  - Graphics3D -
```

Ejercicio 5.4

Dibuja las siguientes curvas y superficies dadas en coordenadas paramétricas.

a) $\gamma(t) = \left(\text{sen}(t), \text{sen}(2t), \frac{t}{5}\right)$, $t \in [0, 5]$,

b) $\gamma(u, v) = (\cos(u)(3 + \cos(v)), \text{sen}(u)(3 + \cos(v)), \text{sen}(v)))$, con $u, v \in [0, 2\pi]$,

c) $\gamma(u, v) = (\text{sen}(u), \text{sen}(v), v)$ con $u \in [-\pi, \pi]$, $v \in [0, 5]$,

d) $\gamma(u, v) = (u\cos(v)\,\text{sen}(u), u\cos(u)\cos(v), -u\,\text{sen}(v))$.

Ejercicio 5.5

Utiliza el comando `ParametricPlot3D` para representar un cilindro de radio 5. ¿Cómo se puede representar un cono truncado?

Ejercicio 5.6

Utiliza el comando `ParametricPlot3D` para representar un paraboloide ($z = x^2 + y^2$), un paraboloide hiperbólico ($z = x^2 - y^2$) y un cono ($z^2 = x^2 + y^2$).

5.3.1 Superficies de revolución

En el caso particular en que la superficie que queremos representar se obtiene girando una curva respecto a un eje se puede usar el paquete `SurfaceOfRevolution`. Para cargarlo se escribe

```
In[12]:=
      <<Graphics`SurfaceOfRevolution`
```

y desde este momento está disponible la orden `SurfaceOfRevolution`

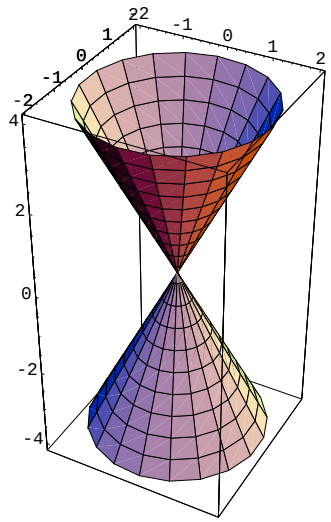
`SurfaceOfRevolution[f(x),{x,a, b}]`	gira la gráfica de la función f
`SurfaceOfRevolution[{f(t),g(t)},{t,a,b}]`	gira la curva $((f(t), g(t))$
`SurfaceOfRevolution[{f(t),g(t),h(t)},{t,a,b}]`	gira la curva $((f(t), g(t), h(t))$

La primera orden dibuja la superficie de revolución obtenida al girar la curva f en el plano XZ entre a y b. La segunda y la tercera funcionan análogamente pero estamos dando una curva en el plano XZ (segundo caso) o en el espacio (último caso). En cualquier caso el giro se hace respecto al eje Z. Si se quiere girar respecto a la recta que une el origen con el punto $\{x_1,x_2,x_3\}$ se puede utilizar la opción `RevolutionAxis`.

`SurfaceOfRevolution[f(t),{t,a,b},` `RevolutionAxis->{x1,x2,x3}]`	gira la gráfica de la función f respecto del vector (x_1, x_2, x_3)

Veamos algunos ejemplos. Un cono se consigue fácilmente girando una recta:

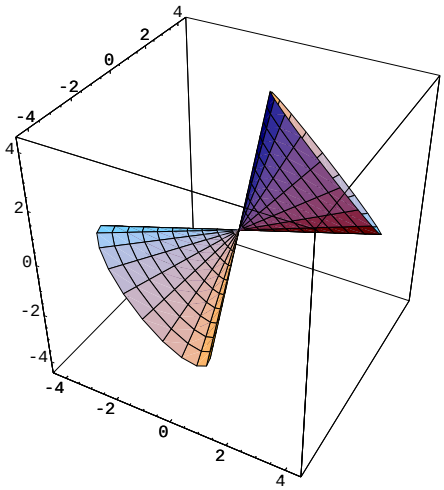
```
In[13]:=
      SurfaceOfRevolution[2 x,{x,-2,2}]
```



```
Out[13]=  - Graphics3D -
```

Si quisiéramos girar respecto a la bisectriz del primer octante, o sea, alrededor de la recta que pasa por el vector (1,1,1) y el origen:

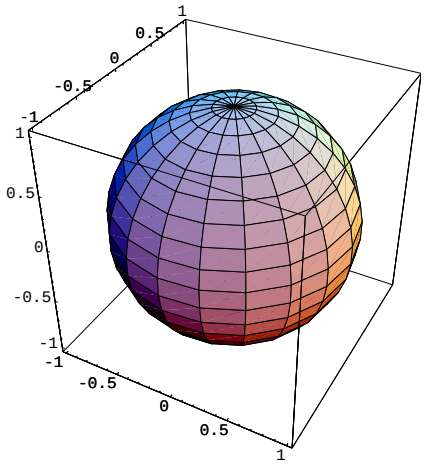
```
In[14]:=
      SurfaceOfRevolution[2x,{x,-2,2},
        RevolutionAxis->{1,1,1}]
```



```
Out[14]=  - Graphics3D -
```

Dibujar una esfera o un elipsoide es ahora simplemente cuestión de girar media circunferencia

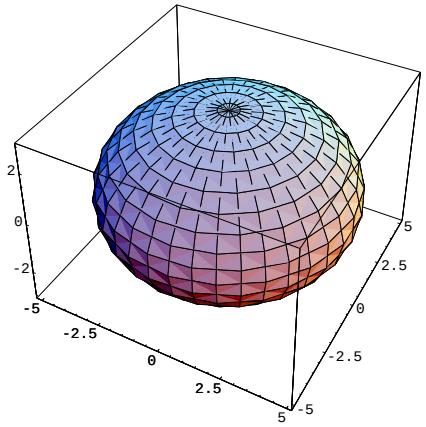
```
In[15]:=
      SurfaceOfRevolution[{Cos[x],Sin[x]},{x,0,Pi},
        AspectRatio->1]
```



```
Out[15]=  - Graphics3D -
```

o media elipse

```
In[16]:=
      SurfaceOfRevolution[{5Cos[x],3Sin[x]},
        {x,-Pi,Pi},AspectRatio->1]
```

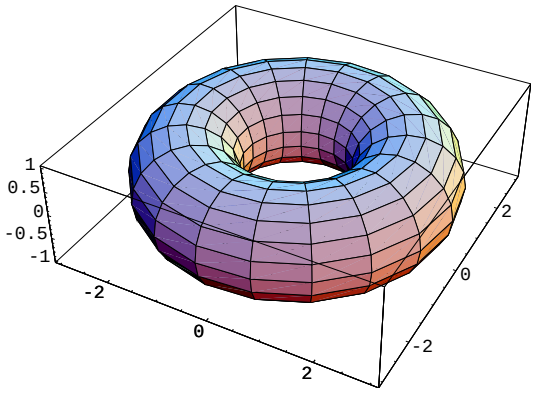


```
Out[16]=  - Graphics3D -
```

Por último, comentar que, como habrás visto en el último ejemplo, las mayoría de las opciones del comando `Plot3D` siguen siendo válidas.

Ejercicio 5.7

A la figura siguiente, obtenida al girar una circunferencia, se la llama toro. Intenta dibujarla.



Diferenciabilidad

6

En esta práctica veremos cómo calcular extremos relativos y condicionados de funciones de varias variables. Se trata de usar el programa *Mathematica* como apoyo para realizar los cálculos necesarios en los métodos de optimización vistos en las clases teóricas.

6.1 Derivadas parciales

El primer objetivo será aprender a calcular derivadas parciales, gradientes, hessianos... de funciones de varias variables, y aplicar todo ello al cálculo de extremos relativos y condicionados.

Ya conocemos de prácticas anteriores la forma de calcular derivadas de funciones de una variable: mediante el comando `D[f[x],{x,n}]` obtenemos la derivada n-ésima de la función f. Para funciones escalares de varias variables (esto es, funciones que "salen" de $\mathbb{R}^n$ y "llegan" a $\mathbb{R}$), el comando `D[f[x,y,...],{x,n1},{y,n2},...]` nos devuelve la derivada parcial de f respecto de x *n1*-veces, respecto de y *n2*-veces...pudiendo omitirse el número de veces si éste es 1. También puede usarse la paleta. En la figura tienes los símbolos que nos permiten escribir derivadas de cualquier orden y derivadas de segundo orden.

`D[f[x,y,...],{x,n1},{y,n2},...]`	derivada parcial de f n_1 veces respecto de x, n_2 veces respecto de y, etc.

Calculemos como ejemplo $\frac{\partial^3 f}{\partial x^2 \partial y}$ y $\frac{\partial^2 f}{\partial x \partial y}$, donde $f(x,y) = \operatorname{sen}(x)\cos(y^2)$:

```
In[1]:=
      f[x_,y_]=Sin[x] Cos[y^2];
      ∂x,x,y f[x,y]
      ∂x,y f[x,y]
Out[1]=
      2y Sin[x]Sin[y^2]
     -2y Cos[x]Sin[y^2]
```

Evidentemente esta no es la forma más cómoda de calcular una derivada parcial de orden 15. Para eso es mejor utilizar `D[f[x,y],{x,8},{y,7}]`.

Por ejemplo para calcular el gradiente de una función de varias variables basta construir el vector formado por las derivadas parciales primeras de la función respecto de todas las variables:

```
In[2]:=
      J[x_,y_]={∂x f[x,y],∂y f[x,y]}
Out[2]=
      {Cos[x]Cos[y^2], -2 y Sin[x] Sin[y^2]}
```

Usando la paleta, el matriz hessiana se calcula de forma análoga:

```
In[3]:=
```
$$\begin{pmatrix} \partial_{x,x}\texttt{f[x,y]} & \partial_{x,y}\texttt{f[x,y]} \\ \partial_{y,x}\texttt{f[x,y]} & \partial_{y,y}\texttt{f[x,y]} \end{pmatrix}$$

Recuerda que las derivadas cruzadas de segundo orden coinciden para funciones "suficientemente buenas".

Observación 6.1. La definición del gradiente y de la matriz hessiana tienen el inconveniente de no ser reutilizables si cambiamos de función. Quizá es más práctico definir gradiente y matriz hessiana de la siguiente forma:

```
In[4]:=
grad[f_][x_,y_]={∂x f[x,y],∂y f[x,y]};
hess[f_][x_,y_]=
```
$$\begin{pmatrix} \partial_{x,x}\texttt{f[x,y]} & \partial_{x,y}\texttt{f[x,y]} \\ \partial_{y,x}\texttt{f[x,y]} & \partial_{y,y}\texttt{f[x,y]} \end{pmatrix};$$

6.1.1 Plano tangente

Ya que sabemos cómo se calculan las derivadas parciales de una función, vamos a intentar representar gráficamente el plano tangente y cómo se obtiene a partir de las derivadas parciales.

Comencemos, por ejemplo, con la función

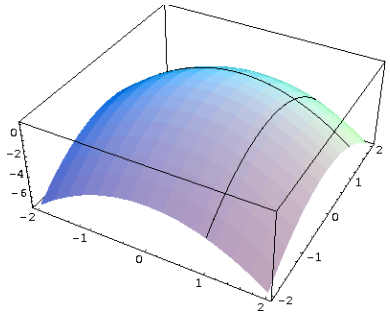
```
In[5]:=
f[x_,y_]=1-(x^2+y^2)
```

y vamos a calcular en el punto $(1,1)$ sus derivadas parciales. La definición de derivada parcial respecto a la primera variable era

$$\frac{\partial f}{\partial x}(1,1) = \lim_{h\to 0} \frac{f(1+h,1) - f(1)}{h}$$

Lo que hacemos es trabajar únicamente con la función definida sobre la recta que pasa por $(1,1)$ y es esa función (de una variable) la que derivamos. La segunda derivada parcial tiene una definición análoga. Dibujemos la gráfica de la función y la imagen de cada una de esas dos rectas:

```
In[6]:=
      Plot3D[f[x,y],{x,-2,2},{y,-2,2},Mesh->False,DisplayFunction->Identity];
      ParametricPlot3D[{1,t,f[1,t]},{t,-2,2},DisplayFunction->Identity];
      ParametricPlot3D[{t,1,f[t,1]},{t,-2,2},DisplayFunction->Identity];
      Show[%%%,%%,%,DisplayFunction->$DisplayFunction]
```



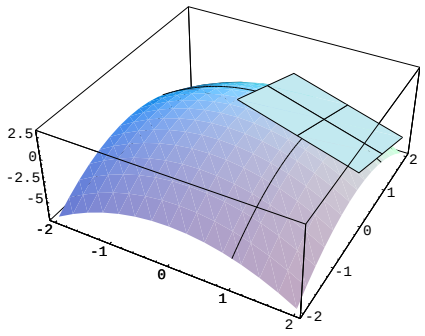
```
Out[6]=  - Graphics3D -
```

Los vectores tangentes a las dos curvas que hemos dibujado son los dos vectores que generan el plano tangente. Por tanto su ecuación será

$$\begin{aligned} z &= f(1,1) + \frac{\partial f}{\partial x}(1,1)(x-1) + \frac{\partial f}{\partial y}(1,1)(y-1) \\ &= f(1,1) + \langle \nabla f(1,1), (x-1, y-1) \rangle \end{aligned}$$

Para acabar nos falta dibujar el plano tangente y unirlo con la gráfica anterior:

```
In[7]:=
      plano[x_,y_]=f[1,1]+grad[f][1,1].{x-1,y-1};
      Plot3D[plano[x,y],{x,0,2},{y,0,2},PlotPoints->3,
        DisplayFunction->Identity];
      Show[%%%%,%,DisplayFunction->$DisplayFunction]
```



```
Out[7]=  - Graphics3D -
```

Ejercicio 6.1

¿Cómo se podría hacer esto con derivadas direccionales en lugar de derivadas parciales?

Ejercicio 6.2

Sea $f(x,y) = \ln\left(1 + x^2 + 2x + y^2\right)$. Calcula el gradiente, la matriz hessiana de f y comprueba que es armónica, esto es, que

$$\frac{\partial^2 f}{\partial x^2}(x,y) + \frac{\partial^2 f}{\partial y^2}(x,y) = 0.$$

Ejercicio 6.3

Comprueba que las funciones

$$f(x,y) = 2\arctan\left(\frac{y}{x + \sqrt{x^2+y^2}}\right) \text{ y } g(x,y) = \arctan\left(\frac{y}{x}\right)$$

tienen las misma derivadas parciales.

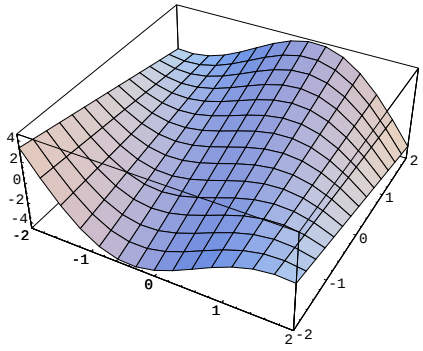
Ejercicio 6.4

Calcula el plano tangente y la recta normal a cada una de las superficies en el punto P:
a) $z^2 - 2x^2 - 2y^2 - 12 = 0$, $P = (1,-1,4)$.
b) $z = \ln\left(x^2 + y^2\right)$, $P = (1,0,0)$.
c) $z + e^z + 2x + 2y - x^2 - y^2 = 3$, $P = (1, 1+\sqrt{e}, 1)$.

6.1.2 Campos de vectores

Sabemos que el gradiente de una función en cada punto es el vector formado por las derivadas parciales e indica la dirección de máxima pendiente. Recordemos que el comando `ContourPlot` dibuja las curvas de nivel de la gráfica de una función, por tanto el gradiente debería ser perpendicular a estas. Para verlo necesitamos algunos comandos que podemos usar cargando el paquete `PlotField`:

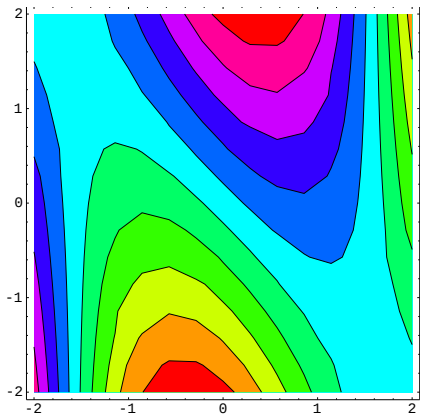
```
In[8]:=
      <<Graphics`PlotField`
In[9]:=
      f[x_,y_]=2(x+y)Cos[x]
Out[9]=
      2(x+y)Cos[x]
In[10]:=
      Plot3D[f[x,y],{x,-2,2},{y,-2,2}]
```




```
Out[10]=  - Graphics -
```

Dibujemos ahora las curvas de nivel. La opción `ColorFunction->Hue` utiliza colores para diferenciar las distintas alturas en lugar de la escala de grises que habíamos visto.

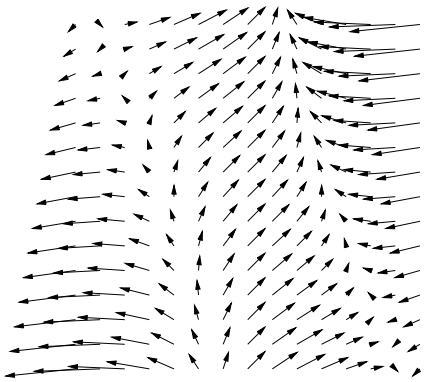
```
In[11]:=
      ContourPlot[f[x,y],{x,-2,2},{y,-2,2},ColorFunction->Hue]
```




```
Out[11]=  - Graphics -
```

La orden `PlotGradientField` dibuja los vectores $\left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right)$. Las dos opciones que aparecen (`ScaleFunction` y `ScaleFactor`) sirven para controlar el tamaño con que se dibujan los vectores.

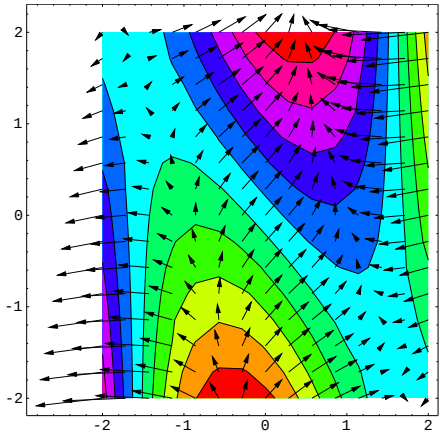
```
In[12]:=
      PlotGradientField[f[x,y],{x,-2,2},{y,-2,2},
        ScaleFunction->(.1#&),ScaleFactor->None]
```



```
Out[12]=  - Graphics -
```

Si mostramos en la misma figura los dos gráficos anteriores podemos ver como el gradiente es perpendicular a las curvas de nivel.

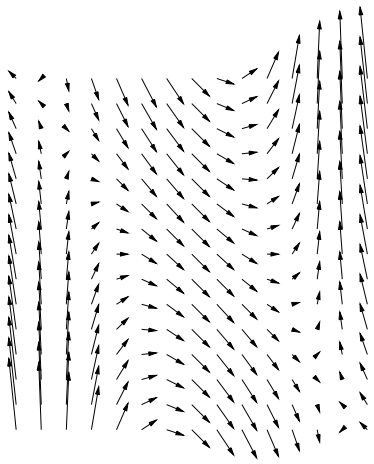
```
In[13]:=
      Show[%,%11]
```




```
Out[13]=  - Graphics -
```

También existe la orden `PlotHamiltonianField` que dibuja los vectores $\left(-\frac{\partial f}{\partial y}, \frac{\partial f}{\partial x}\right)$. Estos son perpendiculares al gradiente y siguen la dirección marcada por las curvas de nivel.

```
In[14]:=
      PlotHamiltonianField[f[x,y],{x,-2,2},{y,-2,2},ScaleFunction->(.1#&),
        ScaleFactor->None,DisplayFunction->Identity]
```



```
Out[14]=  - Graphics -
```

Ejercicio 6.5

Calcular las derivadas parciales de:
a) $f(x,y,z) = x^{y+z},\ \forall x \in \mathbb{R}^+, y, z \in \mathbb{R}$
b) $f(x,y,z) = (x+y)^z,\ \forall x, y \in \mathbb{R}^+, z \in \mathbb{R}$
c) $f(x,y) = \operatorname{sen}(x \operatorname{sen}(y)),\ \forall x, y \in \mathbb{R}$

Ejercicio 6.6

Sea $f : \mathbb{R}^2 \setminus \{(0,0)\} \to \mathbb{R}$ dada por $f(x,y) = \log(x^2 + y^2)$ para todo $(x,y) \neq (0,0)$. Se pide:
a) Calcúlese el gradiente de f en todo punto así como la matriz hessiana.
b) Compruébese que

$$\frac{\partial^2 f}{\partial x^2}(x,y) + \frac{\partial^2 f}{\partial y^2}(x,y) = 0\ \forall (x,y) \in \mathbb{R}^2 - \{(0,0)\}.$$

Ejercicio 6.7

Sea $f : \mathbb{R}^2 \to \mathbb{R}$ una función diferenciable con continuidad. Si $v = f(x,y)$ donde $x = \rho\cos(\theta),\ y = \rho\operatorname{sen}(\theta)$, comprobar que

$$\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} = \frac{\partial^2 v}{\partial \rho^2} + \frac{1}{\rho^2}\frac{\partial^2 v}{\partial \theta^2} + \frac{1}{\rho}\frac{\partial v}{\partial \rho}.$$

6.2 Extremos relativos.

El método para encontrar extremos relativos de funciones de varias variables consiste en buscar primero los puntos críticos, es decir, puntos donde se anula el gradiente, y después estudiar la matriz hessiana en esos puntos. Los resultados que conocemos nos aseguran que todos los puntos extremos de una función están entre los puntos críticos, con lo que una vez calculados éstos nos dedicaremos a estudiar la matriz hessiana en ellos, viendo si es definida, indefinida, semidefinida... Para ello podemos usar el criterio de los valores propios: si todos son del mismo signo, la matriz es definida y hay extremo; si aparecen valores propios de distinto signo es indefinida y hay punto de silla; en otro caso, la matriz es semidefinida y el criterio no decide.

Ejemplo 6.2. Calculemos los extremos relativos de la función $f(x,y) = x^3 + 3xy^2 - 15x - 12y$. Primero definimos la función, su gradiente y su hessiano o, mejor aún, utilizamos las definiciones que habíamos hecho al comenzar el tema para funciones de dos variables: `grad[f][x,y]` y `hess[f][x,y]`.

```
In[15]:=
      f[x_,y_]=x^3 + 3 x y^2 -15 x -12 y;
```

Para calcular los puntos críticos podemos usar el comando `Solve`, ya que el gradiente está formado por polinomios de grado bajo; guardaremos el resultado que nos de `Solve` en una variable, *pcrit*, que mas tarde usaremos:

```
In[16]:=
      pcrit=Solve[grad[f][x,y]==0,{x,y}]
Out[16]=
      {{x -> -2, y ->-1},{x -> -1, y -> -2},
      {x -> 1, y ->2},{x -> 2, y -> 1}}
```

Hemos obtenido así 4 puntos críticos entre los que estarán los extremos. Calculemos ahora los valores propios de la matriz hessiana en los cuatro puntos críticos obtenidos (observa la utilización del comando `/.` que ahora comentaremos)

```
In[17]:=
      Eigenvalues[hess[f][x,y]] /. pcrit
Out[17]=
      {{-6, -18}, {6,-18}, {-6, 18}, {6, 18}}
```

El comando `/.` sirve para asignar valores; en el caso anterior, asignamos la lista de valores que hay en `pcrit` (que son los puntos críticos de f) al comando `Eigenvalues[H[x,y]]`, que calcula los valores propios de la matriz hessiana en un punto (x,y). De esta forma, lo que hemos obtenido es una lista formada por cuatro parejas de valores propios de `H[x,y]` correspondientes a los cuatro puntos críticos. Así, la matriz hessiana en el primer punto crítico, $(-2,-1)$, tiene sus dos valores propios negativos, con lo que es definida negativa y por tanto hay *máximo*; los puntos críticos

segundo y tercero, $(-1,-2)$ y $(1,2)$, son *puntos de silla*, ya que sobre ellos la matriz hessiana es indefinida; por último, el cuarto es un punto de *mínimo*. Si queremos saber lo que vale la función f sobre sus puntos críticos, basta teclear

```
In[18]:=
      f[x,y]/.pcrit
Out[18]=
      {28, 26, -26, -28}
```

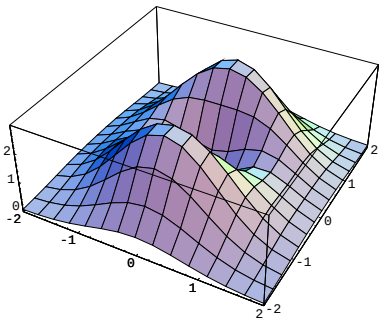
que nos da una lista con los valores de f en los puntos críticos, o bien escribir directamente `f[-2,-1], f[-1,-2]`, etc...
Podemos resumir diciendo que la función f tiene máximo valor relativo 28 en el punto $(-2,-1)$, mínimo valor relativo -28 en $(2,1)$, y dos puntos de silla en $(-1,-2)$ y $(1,2)$.

Ejemplo 6.3. Calculemos los extremos relativos de $g(x,y) = (x^2+3y^2)e^{1-x^2-y^2}$.
Comenzamos definiendo g:

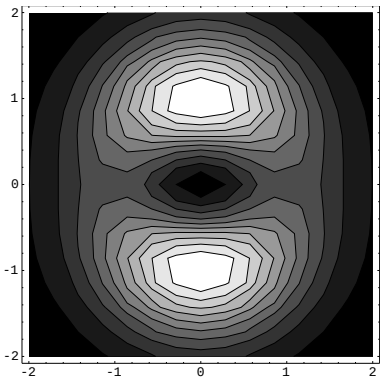
```
In[19]:=
      g[x_,y_]=(x^2+ 3 y^2) Exp[1-x^2-y^2];
```

Si intentamos calcular los puntos críticos resolviendo el sistema `grad[g][x,y]==0` con la orden `NSolve`, *Mathematica* informa que no puede resolverlo, pues aparecen exponenciales, y `NSolve` (igual que `Solve` o `Reduce`) sólo sirven para polinomios y funciones sencillas. En este punto, tenemos dos posibilidades. La primera, trazar la gráfica de f (como superficie o con curvas de nivel) para tener una idea de dónde están los extremos, y usar el comando `FindRoot` con las aproximaciones iniciales que nos da la gráfica.

```
In[20]:=
      Plot3D[g[x,y],{x,-2,2},{y,-2,2}]
```



```
Out[20]=  - SurfaceGraphics -
In[21]:=
      ContourPlot[g[x,y],{x,-2,2},{y,-2,2}]
```



```
Out[21]=  - ContourGraphics -
```

A la vista de las gráficas podemos calcular aproximaciones a los extremos y calcularlos con `FindRoot`, pero ¿qué nos asegura que no hay más extremos que no entran en el rango de la función, o que no vemos por la imprecisión de los gráficos?.
La segunda posibilidad es "ayudar' ' a *Mathematica* a resolver el sistema realizando operaciones algebraicas en el sistema de ecuaciones `grad[g][x,y]==0`, para convertirlo en un sistema polinómico. Esto no siempre puede hacerse, pero en este caso sí: basta multiplicar el sistema por $e^{-(1-x^2-y^2)}$ (que no se anula), y resolver el nuevo sistema que es polinómico:

```
In[22]:=
      h[x_,y_]=Simplify[grad[g][x,y]*Exp[x^2+y^2-1]]
Out[22]=
      {2x(1-x^2-3y^2),2y(3-x^2-3y^2 )}
```

Las soluciones de `h[x,y]==0` serán los puntos críticos de f, que almacenaremos en la variable *pcrit*:

```
In[23]:=
      pcrit=Solve[h[x,y]==0,{x,y}]
Out[23]=
      {{x -> -1, y -> 0}, {x -> 0, y -> 0}, {x -> 1, y -> 0},
      {y -> -1, x -> 0},{y-> 1, x -> 0}}
```

Estudiemos el hessiano en los cinco puntos:

```
In[24]:=
      Eigenvalues[hess[g][x,y]]/.pcrit
Out[24]=
      {{-4, 4}, {2 E, 6 E},{-4, 4}, {-12, -4}, {-12, -4}}
```

Tenemos dos puntos de máximo: $(0,-1)$ y $(0,1)$; un punto de mínimo: $(0,0)$; y dos puntos de silla: $(-1,0)$ y $(1,0)$. La función f vale sobre sus puntos críticos

```
In[25]:=
      g[x,y]/.pcrit
Out[25]=
      {1, 0, 1, 3, 3}
```

con lo que, vale 3 en los dos máximos y 0 en el mínimo.

Ejercicio 6.8

Calcular los extremos relativos de las siguientes funciones.
a) $f(x,y) = x^2 + y^2 - 2x + 4y + 20$.
b) $f(x,y) = x^3 + x^2y + y^2 + 2y + 5$.
c) $f(x,y) = (x^2 + y^2)e^{x^2-y^2}$.
d) $f(x,y,z) = x^2 + y^2 + z^2 - 2x$.
e) $f(x,y) = x^3y^3 - y^4 - x^4 + xy$.

6.3 Extremos condicionados.

Usaremos el método de los multiplicadores de Lagrange para calcular extremos condicionados. Se trata de optimizar una función de varias variables en el conjunto de puntos que verifiquen una cierta ecuación o ecuaciones.

Dada una función suficientemente diferenciable, y una curva o superficie en forma implícita, el método consiste en encontrar los puntos de dicha curva o superficie que verifican un sistema auxiliar, llamado "sistema de Lagrange", que dará el equivalente a los puntos críticos; después habrá una condición sobre el hessiano equivalente a la condición para extremos relativos. Hay que notar que no todas las condiciones son válidas, sino que se ha de imponer una hipótesis técnica, pero en la práctica supondremos hecha la comprobación.

Ejemplo 6.4. Calculemos los extremos de la función $f(x,y) = x^2 - y^2$ condicionados a la ecuación $g(x,y) = x^2 + y^2 - 1 = 0$.
Comenzamos definiendo la función f, la condición, y la función auxiliar de Lagrange

```
In[26]:=
      f[x_,y_]=x^2-y^2;
      g[x_,y_]=x^2 + y^2-1;
      F[x_,y_,λ_]=f[x,y]-λ*g[x,y]
      z=F[x,y,λ]
```

Definimos gradiente y hessiano de la función de Lagrange

```
In[27]:=
      J[x_,y_,λ_]={∂x z, ∂y z};
      H[x_,y_,λ_]=( ∂x,x z   ∂x,y z
                    ∂y,x z   ∂y,y z );
```

y resolvemos el sistema de Lagrange, esto es, `J[x,y,λ]==0` y `g[x,y]==0`. Como estamos con ecuaciones polinómicas, usaremos `Solve`:

```
In[28]:=
      pcrit=Solve[{J[x,y,λ]==0,g[x,y]==0},{x,y,λ}]
Out[28]=
      {{λ -> -1, y -> -1, x -> 0}, {λ -> -1, y -> 1, x -> 0},
        {λ -> 1, x -> -1, y-> 0}, {λ -> 1, x -> 1, y -> 0}}
```

Obtenemos 4 puntos críticos, y para ver si son o no extremos, estudiamos la matriz hessiana auxiliar `H[x,y,λ]` restringida al espacio tangente a la curva en los puntos críticos. Desafortunadamente, no podemos aplicar de una vez la asignación de valores, sino que deberemos ir punto a punto

```
In[29]:=
      G[x_,y_]={{∂x g[x,y],∂y g[x,y]}};
In[30]:=
      nucleo=NullSpace[G[x,y]/. pcrit[[1]]]
Out[30]=
      {{1,0}}
```

(observa que `pcrit[[1]]` nos da el primer punto crítico)

```
In[31]:=
      nucleo.(H[x,y,λ]/.pcrit[[1]]).Transpose[nucleo]
Out[31]=
      {{4}}
```

luego el primer punto es un mínimo condicionado;

```
In[32]:=
      nucleo=NullSpace[G[x,y]/. pcrit[[2]]]
Out[32]=
      {{1,0}}
In[33]:=
      nucleo.(H[x,y,λ]/.pcrit[[2]]).Transpose[nucleo]
Out[33]=
      {{4}}
```

el segundo punto también es mínimo;

```
In[34]:=
      nucleo=NullSpace[G[x,y]/. pcrit[[3]]]
In[35]:=
      nucleo.(H[x,y,λ]/.pcrit[[3]]).Transpose[nucleo]
Out[35]=
      {{-4}}
```

el tercer punto es un máximo;

```
In[36]:=
      nucleo=NullSpace[G[x,y]/. pcrit[[4]]]
Out[36]=
      {{0,1}}
In[37]:=
      nucleo.(H[x,y,λ]/.pcrit[[4]]). Transpose[nucleo]
Out[37]=
      {{-4}}
```

otro máximo.
El valor de la función en los extremos:

```
In[38]:=
      f[x,y]/.pcrit
Out[38]=
      {-1, -1, 1, 1}
```

esto es, vale 1 en los máximos y -1 en los mínimos.

Ejemplo 6.5. Calcular los extremos relativos de $f(x,y,z) = x^2 + y^2 + z^2$ condicionados a la superficie $g(x,y,z) = 2x + 2y - z + 3 = 0$.
Comenzamos con las definiciones:

```
In[39]:=
      f[x_,y_,z_]=x^2+ y^2+z^2;
      g[x_,y_,z_]=2x+2y-z+3;
      F[x_,y_,z_,λ_]=f[x,y,z]-λ*g[x,y,z];
      k1=F[x,y,z,λ];
      J[x_,y_,z_,λ_]={∂x k1,∂y k1,∂z k1};
      H[x_,y_,z_,λ_]={{∂x,x k1,∂x,y k1,∂x,z k1},
      {∂y,x k1,∂y,y k1,∂y,z k1},{∂z,x k1,∂z,y k1,∂z,z k1}};
```

Resolvemos el sistema de Lagrange

```
In[40]:=
      pcrit=Solve[{J[x,y,z,λ]==0,g[x,y,z]==0},{x,y,z,λ}]
Out[40]=
      {{x -> -(2/3), y -> -(2/3), z -> (1/3), λ -> -(2/3)}}
```

y estudiemos el hessiano de la función auxiliar en el espacio tangente a la superficie. Primero calculamos una base de dicho tangente con el comando `NullSpace`:

```
In[41]:=
      k2=g[x,y,z];
      G[x_,y_,z_]={{∂x k2,∂y k2,∂z k2}};
      nucleo=NullSpace[G[x,y,z]/.
      pcrit[[1]]]
Out[41]=
      {{1, 0,2}, {-1, 1, 0}}
```

Ahora calculamos la matriz hessiana restringida al tangente

```
In[42]:=
      M=nucleo.
      (H[x,y,z,λ]/.pcrit[[1]]).Transpose[nucleo]
Out[42]=
      {{10, -2}, {-2, 4}}
```

y estudiamos si es definida mediante sus valores propios

```
In[43]:=
      N[Eigenvalues[M]]
Out[43]=
      {3.39445, 10.6056}
```

Como ambos son positivos, el único punto crítico calculado es un mínimo condicionado, con valor

```
In[44]:=
      f[x,y,z]/.pcrit
Out[44]=
      {1}
```

Ejercicio 6.9

Calcular los extremos condicionados en los siguientes casos:
a) $f(x,y) = x^2 - xy + y^2$ condicionados a $x^2 + y^2 - 4 = 0$.
b) $f(x,y) = x^3 + xy^2$ condicionados a $xy = 1$.
c) $f(x,y,z) = xyz$ condicionados a $x^2 + y^2 + z^2 = 1$.

Ejercicio 6.10

a) Hallar la mínima distancia de $(0,0)$ a $x^2 - y^2 = 1$.
b) Entre todos los ortoedros de volumen 1, determinar el que tiene superficie lateral mínima.
c) Calcular las dimensiones de un ortoedro de superficie lateral 2, para que su volumen sea máximo
d) Se quiere construir un canal cuya sección sea un trapecio isósceles de área dada S. Calcular la profundidad del canal y el ángulo que deben formar las paredes con la horizontal para que la superficie mojada sea mínima. (Solución: se trata de la mitad de un hexágono regular)

6.4 Extremos absolutos

En conjuntos compactos tenemos garantizada la existencia de extremos absolutos para las funciones continuas. Esto nos simplifica su cálculo: los extremos absolutos son extremos relativos, si se encuentran en el interior del dominio, o extremos condicionados en la frontera. Vamos a buscar posibles candidatos a extremos relativos o a extremos condicionados y entre ellos tienen que estar los máximos y mínimos absolutos.

Dependiendo del aspecto de la frontera utilizaremos el método de los multiplicadores de Lagrange o trabajeremos directamente con la función. Comencemos con un ejemplo de este último caso.

Ejemplo 6.6. Vamos a calcular los extremos absolutos de la función $f(x,y) = x^2(y-x)$ en el triángulo de vértices $(0,0)$, $(1,0)$ y $(1,1)$.
En el interior, buscamos puntos críticos de la función:

```
In[45]:=
      Clear[f,x,y]
In[46]:=
      f[x_,y_]=x^2(x-y);
In[47]:=
      Solve[grad[f][x,y]==0,{x,y}]
Out[47]=
      Solve::svars: Equations may not give solutions for all "solve"
      variables.
      {{x->0},{x->0}}
```

Las únicas soluciones son los puntos de la forma $(0,y)$ que *no* pertenecen al interior del dominio. La frontera del conjunto está formada por tres segmentos. Vamos a estudiarlos uno a uno. En primer lugar el segmento que el origen y el punto $(1,0)$. En dicho segmento $y = 0$ y $x \in [0,1]$. Los posibles extremos son

```
In[48]:=
      Solve[D[f[x,0],x]==0,x]
Out[48]=
      {x->0}
```

Nos sale el origen. En segundo lugar, el segmento que une $(1,0)$ y $(1,1)$. En este caso $x = 1$ e y varía entre 0 y 1:

```
In[49]:=
      Solve[D[f[1,y],x]==0,y]
Out[49]=
      { }
```

No hay solución. El último segmento une el origen y el punto $(1,1)$. En este caso $x = y$,

```
In[50]:=
      Solve[D[f[x,x],x]==0,x]
Out[50]=
      {{ }}
```

Todo el segmento está formado por puntos críticos. A estos puntos tenemos que añadir los extremos del intervalo (los vértices del triángulo): siempre cabe la posibilidad de que una función de una variable alcance un extremo absoluto en uno de ellos.
Para terminar sólo te queda evaluar. Calcular `f[0,0]`, `f[1,0]`, `f[1,1]` y `f[x,x]`. ¿Cuáles son los extremos absolutos de la función?

Ejemplo 6.7. Calculemos los extremos absolutos de la función $f(x,y) = x^2 - y^2$ en el círculo de centro el origen y radio uno.
De nuevo estamos ante una función continua en un dominio cerrado y acotado por lo que tenemos garantizada la existencia de extremos absolutos. Los puntos críticos en el interior son

```
In[51]:=
      Clear[f,x,y]
      f[x_,y_]=x^2-y^2;
      Solve[grad[f][x,y]==0,{x,y}]
Out[51]=
      {x->0,y->0}
```

Bien, ya tenemos un punto. ¿Qué hacemos con la frontera? Calculamos los puntos críticos de la función de Lagrange

```
In[52]:=
      F[x_,y_,λ_]=f[x,y]+λ(x^2+y^2-1);
      Solve[{D[F[x,y,λ],x]==0,D[F[x,y,λ],y]==0,x^2+y^2==1},{x,y,λ}]
Out[52]=
      {{x->0,y->0},{x->1,y->0},{x->0,y->1},{x->-1,y->0},{x->0,y->-1}}
```

Evalúa la función f en estos puntos para averiguar dónde alcanza los extremos absolutos.

Ejercicio 6.11

Calcula los extremos absolutos de
a) $f(x,y) = x^2 - xy + y^2$ en el conjunto $\left\{(x,y);\ x^2+y^2 = 4\right\}$,
b) $f(x,y) = xyz$, en el conjunto $x^2 + y^2 + z^2 \leq 1$.

Integrales múltiples

7

Ya conocemos el comando `Integrate` para realizar integrales de funciones de una variable y, como podrás imaginar por lo visto hasta ahora, pasar a varias variables sólo será cuestión de poner algunas variables más y alguna coma. En general, la principal diferencia entre una y varias variables no es tanto la función a integrar como el recinto de integración (por ejemplo, no hay una manera directa de decirle a *Mathematica* que queremos integrar en un círculo). Veremos en esta tema cómo podemos ayudarnos de las capacidades gráficas de *Mathematica* para escribir los recintos de integración.

Expondremos sólo el caso de dos variables; para integrales triples los comandos son totalmente análogos.

Como recordarás de clase, sabemos calcular la integral de una función $f(x,y)$ en recintos de alguno de los siguientes tipos:

$$A = \{(x,y) \in \mathbb{R}^2 : \ a \le x \le b, \ g_1(x) \le y \le g_2(x)\}$$

o

$$B = \{(x,y) \in \mathbb{R}^2 : \ a \le y \le b, \ h_1(y) \le x \le h_2(y)\}$$

Las correspondientes integrales serían, vía el teorema de Fubini,

$$\int_A f(x,y)\ \mathrm{d(x,y)} = \int_a^b \left(\int_{g_1(x)}^{g_2(x)} f(x,y)\, dy \right) dx$$

y

$$\int_B f(x,y)\ \mathrm{d(x,y)} = \int_a^b \left(\int_{h_1(x)}^{h_2(x)} f(x,y)\, dx \right) dy.$$

Ya hemos visto que la orden `Integrate` calcula la integral de manera exacta y que `NIntegrate` calcula una aproximación numérica. Estos comandos siguen siendo válidos para funciones de varias variables.

`Integrate[f(x,y),{x,a,b},{y,c,d}]`	Integral de la función en $[a,b]\times[c,d]$
`NIntegrate[f(x,y),{x,a,b},{y,c,d}]`	Integral aproximada de la función en $[a,b]\times[c,d]$

El caso más fácil se presenta cuando integramos en rectángulos. En este caso las funciones g_i o h_i son constantes. Por ejemplo, si queremos calcular la integral de $f(x,y) = x^2 + y^2$ en $[0,2]\times[1,5]$ tendríamos que hacer lo siguiente:

```
In[1]:=
      Integrate[x^2 + y^2,{x,0,2},{y,1,5}]
Out[1]=
      280/3
```

Pasemos a integrales un poco más complicadas. Supongamos que queremos integrar la función $f(x,y) = x$ en el conjunto

$$A = \{(x,y) \in \mathbb{R}^2 : \; x^2 + y^2 \leq 1, x \geq 0, y \geq 0\}.$$

El primer paso es intentar averiguar cómo es el conjunto, para lo que nos haría falta dibujar $x^2 + y^2 = 1$. Para ello, necesitamos un paquete de *Mathematica* para dibujar curvas que vienen dadas por una ecuación (llamada "ecuación implícita"). La orden para cargar este paquete extra es:

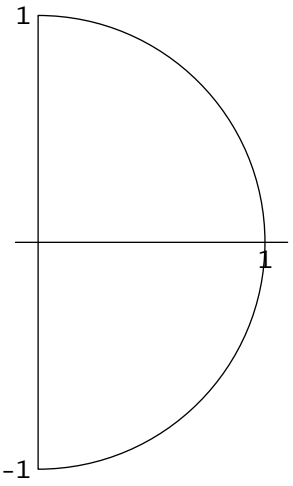
```
In[2]:=
      <<Graphics`ImplicitPlot`
```

(observa que no produce ninguna salida). Ahora ya podemos pedir que nos pinte $x^2 + y^2 = 1$ para $x \in [0,1]$.

`ImplicitPlot[`*ecuación*`,{x,a,b}]`	Representa los puntos que cumplen la *ecuación*

Los valores a y b donde se representa el rango de variación de la variable x lo tendremos que calcular "a ojo". En nuestro caso

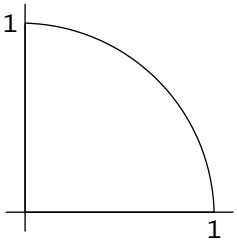
```
In[3]:=
      ImplicitPlot[x^2+y^2==1,{x,0,1}]
```




```
Out[3]=  - Graphics -
```

Cómo se puede ver, *Mathematica* dibuja la circunferencia para "y" positivo y negativo, en cambio sólo nos interesa la parte del primer cuadrante. Para representar esta parte podemos restringir el rango de la imagen con la opción `PlotRange` (la orden `ImplicitPlot` tiene básicamente las mismas opciones que la orden `Plot`).

```
In[4]:=
      ImplicitPlot[x^2+y^2==1,{x,0,1}, PlotRange->{0,1}]
```




```
Out[4]=  - Graphics -
```

Después de ver el conjunto A donde estamos integrando, es claro cómo podemos poner una variable en función de la otra. Por ejemplo, si ponemos y en función de x, para un valor concreto de x entre 0 y 1, y varía entre 0 y $\sqrt{1-x^2}$. Podemos calcular ahora el valor de la integral:

```
In[5]:=
      Integrate[x,{x,0,1},{y,0,Sqrt[1-x^2]}]
Out[5]=
      1/3
```

Como se puede observar el orden para evaluar la integral es de derecha a izquierda, así que si y depende de x, el rango de x va primero.

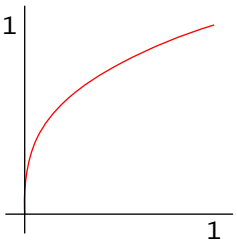
Observarás que no hemos cambiado a coordenadas polares. Para integrales en varias variables, interesa cambiar de variable cuando el dominio o la función son complicados. En el primer caso, si no podemos escribir el dominio de una las formas (1) o (2) tendremos que cambiar de variable nosotros (*Mathematica no* lo hace automáticamente). El caso de funciones complicadas no nos debe preocupar, pues *Mathematica* se encarga de resolver la integral en la inmensa mayoría de los casos, y si no se puede resolver directamente, siempre queda la posibilidad de aproximar el valor de integral.

Veamos un par de ejemplos.

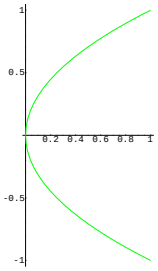
Ejemplo 7.1. *Calcular la integral de la función* $f(x,y) = e^{x/y}$*, en el conjunto* $A = \{(x,y) \in \mathbb{R}^2 : 0 \le y^3 \le x \le y^2\}$.

Estamos ante un conjunto del tipo 2. Tenemos la variable x en función de y. Vamos primero a ver los puntos de corte de las ecuaciones $y^3 = x$, $y^2 = x$ y luego las dibujaremos para hacernos una idea de cuál es el dominio de integración:

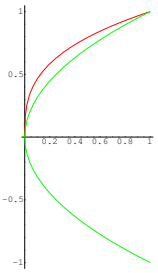
```
In[6]:=
      <<Graphics`ImplicitPlot`
In[7]:=
      Solve[y^3 == y^2,y]
Out[7]=
      {{y->0},{y->0},{y->1}}
In[8]:=
      graf1=ImplicitPlot[y^3==x,{x,0,1},PlotStyle->RGBColor[1,0,0]]
```




```
In[9]:=
      graf2=ImplicitPlot[y^2==x,{x,0,1}, PlotStyle->RGBColor[0,1,0]]
```




```
In[10]:=
      Show[graf1,graf2]
```



De lo hecho se deduce que la integral que queremos calcular es

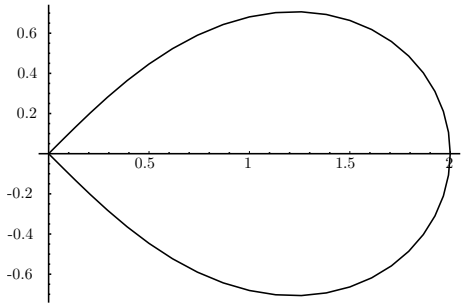
```
In[11]:=
      Integrate[Exp[x/y],{y,0,1},{x,y^3, y^2}]
Out[11]=
      3/2 - E/2
```

Ejemplo 7.2. *Calcular la integral de la función* $f(x,y) = x^2 + y^2$ *en el conjunto*

$$A = \{(x,y) \in \mathbb{R}^2 : \ (x^2+y^2)^2 \leq 4(x^2-y^2),\ x \geq 0\}.$$

En este caso la función es relativamente sencilla pero el dominio no lo es tanto. No hay una manera fácil de hacer la integral directamente, así que es mejor cambiar a coordenadas polares y luego plantear la integral. Dibujemos el recinto:

```
In[12]:=
      ImplicitPlot[(x^2+y^2)^2==4(x^2-y^2),{x,0,3}]
```



Las condiciones del dominio escritas en coordenadas polares son:

$$\rho^4 \leq 4\rho^2(\cos^2(\theta) - \operatorname{sen}^2(\theta)),\ \rho\cos\theta \geq 0.$$

Después de operar, nos quedaría

$$\rho \leq 2\sqrt{\cos^2(\theta) - \operatorname{sen}^2(\theta)} \ \text{ y } \ \theta \in [-\pi/4, \pi/4].$$

Ahora ya estamos en una de los casos conocidos y podemos plantear la integral, sin olvidar el cambio a coordenadas polares de la función y la "derivada" del cambio (en este caso ρ):

```
In[13]:=
      Integrate[ρ * ρ^2,{t, -Pi/4, Pi/4}, {ρ,0,2 Sqrt[Cos[t]^2 -Sin[t]^2]}]
Out[13]=
      π
```

Ejercicio 7.1

Calcular la integral de las siguientes funciones en los recintos determinados por las siguientes ecuaciones:

a) $f(x,y) = \sqrt{4x^2 - y^2},\ x = 1,\ y = 0,\ y = x$
b) $f(x,y) = xe^{-x^2/y},\ x = 0,\ y = 1,\ y = x^2$
c) $f(x,y) = y,\ y \geq 0, x^2 + y^2 = a^2,\ y^2 = 2ax, x = 0$

Observación 7.3. Respecto al último apartado del ejercicio anterior: debería dar lo mismo pero no. En el primer caso *Mathematica* asume que a es negativo.

```
In[14]:=
        FullSimplify[∫_0^(a*√(-1+√2)*√2) ∫_(y^2/(2a))^√(a^2-y^2) y dx dy]
Out[14]=
        (-3/2 + √2) a^3 + 1/3 (8 - 5√2) (a^2)^3/2
In[15]:=
        FullSimplify[∫_0^(a*√(-1+√2)*√2) ∫_(y^2/(2a))^√(a^2-y^2) y dx dy, a > 0]
Out[15]=
        1/6 (7 - 4√2) a^3
```

Ejercicio 7.2

Calcular el volumen del conjunto A en cada uno de los siguientes casos:
a) $A = \left\{(x,y,z) \in \mathbb{R}^3 :\ x^2 + y^2 \leq z \leq \sqrt{x^2 + y^2}\right\}$
b) $A = \left\{(x,y,z) \in \mathbb{R}^3 :\ \frac{x^2}{a^2} + \frac{y^2}{b^2} \leq 1,\ 0 \leq z \leq \sqrt{\frac{x^2}{a^2} + \frac{y^2}{b^2}}\right\}$
c) $A = \left\{(x,y,z) \in \mathbb{R}^3 :\ 0 \leq z \leq x^2 + y^2,\ x + y \leq 1,\ x, y \geq 0\right\}$
d) $A = \left\{(x,y,z) \in \mathbb{R}^3 :\ 0 \leq z \leq \sqrt{x^2 + y^2},\ x^2 + y^2 \leq 2y\right\}$
e) $A = \left\{(x,y,z) \in \mathbb{R}^3 :\ 0 \leq z \leq 4 - y^2,\ 0 \leq x \leq 6\right\}$
f) $A = \left\{(x,y,z) \in \mathbb{R}^3 :\ \sqrt{x} \leq y \leq 2\sqrt{x},\ 0 \leq z \leq 9 - x\right\}$
g) $A = \left\{(x,y,z) \in \mathbb{R}^3 :\ x^2 + y^2 \leq z^2,\ x^2 + y^2 + z^2 \leq 2z\right\}$

Ejercicio 7.3

Calcular el volumen limitado superiormente por el cono $4x^2 + 4y^2 - z^2 = 0$, inferiormente por el plano $z = 0$ y lateralmente por el cilindro $x^2 + (y-2)^2 = 4$.

Ejercicio 7.4

Calcular el volumen de los sólidos siguientes:
a) Intersección de los cilindros $x^2 + y^2 = r^2,\ y^2 + z^2 = r^2$.
b) El limitado por el plano $z = 0$, el cilindro $x^2 + y^2 = 2ax$ $(a > 0)$ y el cono $x^2 + y^2 = z^2$.

Avisos y mensajes de error

A

```
In[1]:=
      C=3
     Set::wrsym: Symbol C is Protected
```

La letra C no se puede usar como nombre de variable. *Mathematica* ya lo usa para algo.

```
In[2]:=
      Plot[Log[x],{ x,-2,2}]
     Plot::"plnr":Log[x] is not a machine-size real number at x = -2...
```

La función (en este caso logaritmo neperiano) no está definida en ese punto. Probablemente estemos intentando dibujar una función fuera de su dominio o, si la función la hemos definido nosotros, hay algún error en la definición de la función.

```
In[3]:=
      Plot[x^2 +1,{x,o,10}]
     Plot::"plln": Limiting value o in x, o, 2 Pi is not a machine-size real
     number.
```

Has escrito la letra “o” en lugar de 0.

```
In[4]:=
      Plot[sin[x],{x,0,Pi}]
     General::"spell": "Possible spelling error: new symbol name sin is
     similar to existing symbol Sin.
```

La función `sin` no está definida. La función seno es `Sin`.

```
In[5]:=
      Plot[Sin[x],{x,-Pi,Pi},PlotStyle->RGBColor[1,0,0]
     Syntax::"bktmcp":"Expression "Plot[ << 1>> " has no closing "]''.
```

Falta cerrar un corchete.

```
In[6]:=
      Solve[Cos[x]==0,x
     Solve::ifun: Inverse functions are being used by Solve, so some
     solutions may not be found.
Out[6]=
      {x ->{ -π/2}, {x -> π/2}}
```

Para resolver la ecuación $\cos(x) = 0$ tiene que usar la funcion arccos(su inversa) y *Mathematica* avisa de que es posible que falten soluciones

```
In[7]:=
      NIntegrate[Cos[x],{x,0,Pi}]
     NIntegrate::"ploss": Numerical integration
     stopping due to loss of precision. Achieved
     neither the requested PrecisionGoal nor
     AccuracyGoal; suspect highly oscillatory
     integrand, or the true value of the integral
     is 0. If your integrand is oscillatory try
     using the option Method->Oscillatory in NIntegrate.
```

Mathematica no puede asegurar que el valor de la integral que da sea exacto, probablemente debido a que la función oscila.

```
In[8]:=
      FindRoot[{x*y==1,x^2+y^2=1},{x,0},{y,0}]
     FindRoot::jsing: Encountered a singular Jacobian at the
     point {x, y} = {0.,0.}. Try perturbing the initial point(s).
```

No se puede tomar el punto $(0,0)$ como condición inicial. Intenta cambiar el punto por otro.

```
In[9]:=
      FindRoot[5 x^2+3,{x,1}]
     FindRoot::cvnwt: Newton's method failed to converge to the
     prescribed accuracy after 15 iterations
Out[9]=
      {x->-0.6}
```

Aunque en este caso la función no se anula nunca, *Mathematica* está avisando que el comando `FindRoot` no puede afirmar que el resultado que presenta sea un cero de la función.

```
In[10]:=
      A={{1,2,3},{4,5,6},{1,3,2}};
      B={{4,3},{7,3},{1,0}};
      A.B
     Dot::dotsh: Tensors {{1, 2, 3}, {4, 5, 6}, {1, 3, 2}} and
     {{4, 3}, {7, 3}} have incompatible shapes.
```

No se puede multiplicar una matriz 3x3 y una matriz 2x2.

```
In[11]:=
      Integrate[1/x^3,{x,-1,1}]
     Integrate::idiv: Integral of 1/x^3 does not converge on{-1,1}.
```

La función $\frac{1}{x^3}$ no es impropiamente integrable en el intervalo $[-1,1]$.

Índice alfabético