MATEMÁTICAS I

ELECTRÓNICOS

Práctica 2.1:
FUNCIONES DE UNA VARIABLE

Prof: José Antonio Verdoy González

DEPARTAMENTO DE MATEMÁTICA APLICADA
ESCUELA TÉCNICA SUPERIOR
DE INGENIERÍA DEL DISEÑO
UNIVERSIDAD POLITÉCNICA DE VALENCIA

1. Definición y Evaluación

El *Mathematica* tiene ya definidas muchísimas funciones que nos serán de gran utilidad a la hora de trabajar y aprender Matemáticas. En esta práctica vamos a conocer la forma de escribir y utilizar funciones como las *trigonométricas*, *hiperbólicas*, *exponenciales* y sus funciones inversas.

La forma de definir una función real de variable real por el usuario es la siguiente:

```
f[x_]:= expresión  o  f[x_]= expresión
```

El nombre de esta función así creada sería `f[x]`, el equivalente al nombre que se les suele adjudicar en textos y clases de matemáticas: $f(x)$, pero, en vez de utilizar los paréntesis es obligado utilizar corchetes. También es obligado escribir la variable independiente x con un guión bajo "`x_`" para que el *Mathematica* la reconozca como variable y no le sea asignado ningún valor. Y el signo "`:=`" es opcional y puede usarse solamente un "`=`" pero siempre y cuando nuestro objetivo sea definir una función **real de variable real**. *Mathematica* permite definir diferentes tipos de funciones donde la/s variable/s puede ser un gráfico, una matriz, etc. En esos casos es obligatorio utilizar el "`:=`". Pero ya que en este curso de primero las funciones más utilizadas son las usuales (funciones reales de variable real) se recomienda utilizar "`=`" para definir nuestras funciones. Además, de esta forma podremos ver el `Ouput[ ]` de la función definida y observar posibles errores, si los hubiere. Con "`:=`" *Mathematica* **no devuelve** `Ouput[ ]`.

Recordad: a la hora de definir la función, la variable x hay que escribirla con guión bajo. Pero afortunadamente, solo hay que ponerle ese guión bajo en el momento de la definición. Después, cuando la utilicemos para derivarla o integrarla, resolver ecuaciones, etc., bastará con llamarla por su nombre: `f[x]`.

Ejemplo 1 *Define la función* $f(x) = x^2 \ln(x+1)$. *Evalúala en* $x = 3$ *y calcula su función derivada.*

Solución:

```
In[]:= f[x_]=x^2Log[x+1]                    Out[]= x^2Log[x+1]
```

Nótese de qué color queda la variable independiente x, *y cuando el cálculo es exacto o aproximado.*

```
In[]:= f[3]                                 Out[]= 9Log[4]
In[]:= f[3]//N                              Out[]= 12.4766
In[]:= f[3.]                                Out[]= 12.4766
```

Y para calcular su función derivada, comentar que el comando que calcula funciones derivadas en Mathematica es `D[función,variable]`

```
In[]:= D[f[x],x]                            Out[]= x^2/(1+x)+2xLog[1+x]
```

Importante: La derivada de una función de una variable también puede calcularse con la notación típica de $f'(x)$. Siendo la tilde que hay que utilizar en *Mathematica*, la de la tecla *interrogación*.

```
In[]:= f'[x]                                Out[]= x^2/(1+x)+2xLog[1+x]
```

Y otra cosa, en ningún momento hay que escribir/ejecutar `x=3`, primero porque al asignarle a la variable x un valor, ésta dejaría de ser variable (por ejemplo, no se podría derivar o integrar respecto de x). Y segundo, ¿para qué escribir `x=3` si podemos escribir directamente el número 3?. De todas formas, las asignaciones, por si necesitamos que una variable tome algún valor determinado se realizarán de la forma "`x→3`", no con el signo igual.

Aún así, si por una de esas, "*maniatamos*" alguna variable de las más utilizadas $(x, y, z, ...)$ con alguna asignación $(x = 3, y = -2, z = \pi, ...)$ siempre podremos liberarlas para que puedan seguir siendo variables independientes (por ejemplo, para plantear y resolver un sistema de ecuaciones lineales). Para ello se utiliza la orden `Clear[var]`, es decir y en este caso `Clear[x,y,z]`. **Nota**: **NO** funciona la orden `ClearAll`.

2. Funciones elementales

2.1. Polinomios y expresiones racionales

Las funciones polinómicas y las fracciones algebraicas, obtenidas por cocientes de éstas, son muy utilizados en Matemáticas. Entre los comandos más útiles para la manipulación de este tipo de funciones tenemos:

- `Factor[polinomio]` Factoriza un polinomio con raíces enteras, fraccionarias y complejas.
- `Expand[polinomio]` Desarrolla los productos y potencias de polinomios.
- `Simplify[exp_racional]` Simplifica fracciones algebraicas.

Ejemplo 2 *Simplifica el cociente* $\dfrac{1-x^2}{1-x}$ *y factoriza el numerador de dicha fracción.*

Solución:

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

Nota: Estas tres instrucciones funcionan igual si se escriben al final de la expresión, de la forma `//comando`.

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

Ejemplo 3 *Desarrolla* $(x+2)^2(x-1)^4$ *y evalúa dicho polinomio en* $x = -3$.

Solución: *Para obtener el desarrollo de estas potencias de binomios hay que escribir*

```
In[]:= Expand[(x+2)^2(x-1)^4]           Out[]= 4-12x+9x^2+4x^3-6x^4+x^6
In[]:= (x+2)^2(x-1)^4//Expand           Out[]= 4-12x+9x^2+4x^3-6x^4+x^6
```

Y para calcular el valor del polinomio en $x = -3$ *no será necesario definir el polinomio como función, basta con utilizar el comando* ***sustitución*** "`/.`":

```
In[]:= (x+2)^2(x-1)^4/.x->-3            Out[]= 256
```

Recordad: para una simple sustitución de una variable por un valor numérico a, no es necesario definir una función $f(x)$ para luego calcular $f(a)$. Basta con utilizar el comando *sustitución* del *Mathematica*, "`/.`" y poner después: la variable a sustituir, una *flecha* y el valor a sustituir: $x \to a$. Para conseguir la "*flecha*" $\to$ en *Mathematica*: *guión* y *mayor*, "`->`", y se "*juntan*" al escribir el valor de a.

Ejemplo 4 *Factoriza, si es posible, los polinomios*:

$a)$ $2x^4 - 13x^3 + 16x^2 + 19x - 12$ $\qquad$ $b)$ $12x^5 - 3x^4 + 40x^2 - 3x + 13$.

Solución:

```
In[]:= Factor[2x^4-13x^3+16x^2+19x-12]     Out[]= (-4+x)(-3+x)(1+x)(-1+2x)
In[]:= Factor[12x^5-3x^4+40x^2-3x+13]      Out[]= 12x^5-3x^4+40x^2-3x+13
```

2.2. Funciones logarítmicas y exponenciales. Funciones hiperbólicas y sus inversas

Ya hemos utilizado el logaritmo neperiano o natural en algún ejemplo, no obstante lo vamos a incluir en la siguiente lista:

- `Log[x]` calcula el logaritmo natural o neperiano de `x`. $\ln x$.
- `Log[b,a]` calcula el logaritmo en base `b` de `a`. $\log_b a$.
- `Exp[x]` da el número e elevado a `x`. Es decir, la función exponencial e^x.

Ejemplo 5 *Calcula, de forma aproximada,* $\ln 3$, $\log_{10} 2$ *y* $e^{5 \ln 3}$. *Comprueba que utilizando las diferentes identidades que existen para este tipo de funciones, la segunda y tercera expresión valen, respectivamente:* $\dfrac{\ln 2}{\ln 10}$ *y* 3^5.

Solución: *Para ello, debemos escribir*

```
In[]:= Log[3.]                                  Out[]= 1.09861
In[]:= {Log[10,2.],Exp[5Log[3]]}                Out[]= {0.30103,243}
In[]:= {Log10[2.],Log[2]/Log[10.],3^5}          Out[]= {0.30103,0.30103,243}
```

Las funciones hiperbólicas, se escriben en *Mathematica* de la siguiente forma:

`Sinh[x]`, `Cosh[x]`, `Tanh[x]`, `Coth[x]`, `Sech[x]` y `Csch[x]`.

Mientras que sus funciones inversas respectivas son:

`ArcSinh[x]`, `ArcCosh[x]`, `ArcTanh[x]`, `ArcCoth[x]`, `ArcSech[x]` y `ArcCsch[x]`.

2.3. Valor absoluto y raíz cuadrada. Constantes básicas

También hemos visto ya la función $f(x) = \sqrt{x}$ que se escribe `Sqrt[x]`. Mientras que para cualquier otra raíz de otro índice $\sqrt[n]{x}$, habría que definirla como una potencia de exponente fraccionario: $\sqrt[n]{x}$ = `x^(1/n)`.

La función **valor absoluto** se escribe `Abs[x]` y como constantes básicas más usadas en el *Mathematica* tenemos:

`Pi` número π, `E` número e, `Infinity` símbolo del ∞ y la unidad imaginaria i, `I`.

3. Las funciones trigonométricas y sus inversas. Radianes y grados sexagesimales, también en formato decimal

Las seis conocidas funciones trigonométricas sen(x), cos(x), tg(x), ctg(x), sec(x) y cosec(x) se escriben con el *Mathematica*:

`Sin[x]`, `Cos[x]`, `Tan[x]`, `Cot[x]`, `Sec[x]` y `Csc[x]`.

Mientras que sus funciones inversas respectivas son:

`ArcSin[x]`, `ArcCos[x]`, `ArcTan[x]`, `ArcCot[x]`, `ArcSec[x]` y `ArcCsc[x]`.

Como ya sabemos y no debemos olvidar **las funciones trigonometricas y su inversas son funciones reales de variable real**. Es decir, sus argumentos (x) son valores reales y sus imágenes ($f(x)$, salida-ouput) son tambien valores reales. Mientras que los grados sexagesimales **no son valores reales**, los radianes SÍ. Luego, cuando escribamos `Cos[30]` el argumento o variable x que vale 30 no son grados, son **radianes**. Si por el contrario queremos calcular o hallar el valor del coseno de 30 grados sexagesimales (30°) deberemos escribir `Cos[30Degree]`

Ejemplo 6 *Calcula el coseno de* 30 *radianes y el de* 30°.

Solución: *Hallamos primero el coseno de* 30 *radianes*

```
In[]:= Cos[30]                    Out[]= Cos[30]
```

Recordad que Mathematica siempre calcula en forma exacta, si lo que queremos es una respuesta numérica (*tipo calculadora*) *debemos escribir*

```
In[]:= Cos[30]//N                 Out[]= 0.154251
In[]:= Cos[30Degree]              Out[]= Sqrt[3]/2
In[]:= Cos[30Degree]//N           Out[]= 0.866025
```

El factor de conversión `Degree` vale $\frac{\pi}{180}$ (`Degree` es una constante numérica que vale $\frac{\pi}{180} = 0.0174533$ y es el valor de 1° en radianes). Más adelante hablaremos sobre él pero por ahora solamente nos basta con saber que cuando queramos calcular la razón trigonométrica (ojo, no las **razones trigonométicas inversas**) de un ángulo en grados sexagesimales hay que "*acompañar*/*multiplicar*" su valor numérico con/por `Degree`.

Nota: Tambien podemos utilizar el "*símbolo*" del grado en *Mathematica* sin necesidad de multiplicar por la constante `Degree` sin más que pulsar: *Esc* + deg + *Esc* (*Esc* corresponde a pulsar la tecla Escape). Y recordad, cuando la variable no tenga unidad o símbolo estaremos hablando de radianes porque los grados sexagesimales sí se distinguen con su famoso "*circulito superior*".

Se recomienda utilizar la **ayuda** del *Mathematica* para manipular y obtener cálculos utilizando también grados sexagesimales y grados en decimales. Como por ejemplo, saber utilizar los comandos `DMSList[ ]`, `DMSString[ ]` y `FromDMS[lista]`, donde en este último comando, `lista` está formada por una secuencia `{g,m,s}` de grados, minutos y segundos (DMS).

Ejemplo 7 *Calcula, con* 8 *cifras significativas, el valor de* $\cos(114^\circ\ 35'\ 29'')$.

Solución: *Primero vamos a transformar los* DMS (*grados, minutos y segundos*) *en* ***grados decimales****. Para ello podemos utilizar* `FromDMS[lista]` *o incluso calcularlo a "mano".*

```
In[]:= ang=FromDMS[{114,35,29}]          Out[]= 412529/3600
In[]:= ang=114+35/60+29/3600             Out[]= 412529/3600
```

Y ahora, a calcular el coseno de dicho ángulo, pero éste tiene que estar en radianes.

```
In[]:= N[Cos[ang*Degree],8]              Out[]= -0.41614414
```

Nota: El ángulo dado en este último ejemplo es *casi* el doble de 1 radián (salvo décimas de segundo). Así pues, si calculásemos cos(2) directamente, obtendríamos un valor muy próximo al obtenido en este ejemplo. Comprobadlo.

Luego no es lo mismo dar el argumento en radianes que en grados, y aunque las calculadoras y programas matemáticos trabajen con ambas unidades siempre habrá que pensar que lo correcto (y usual) es utilizar los radianes, y cuando lo hagamos con **grados**, tener muy presente que estamos trabajando con ellos, con grados. En definitiva, el usuario siempre tiene que saber que unidad utiliza (radianes o grados) y la manera de expresárselo al "*programa*" que esté usando.

Está claro que el único inconveniente de los radianes es que los **ángulos notables** están todos en función de π y si queremos trabajar con valores numéricos de radianes, estos ángulos notables pasan a tener una expresión decimal infinita. Por ejemplo:

$$2\pi = 360^\circ = 6.2832 \text{ rad},\ \ \pi = 180^\circ = 3.1416 \text{ rad},\ \ \frac{\pi}{2} = 90^\circ = 1.5708 \text{ rad},\ \ 1 \text{ rad} = 57{,}296^\circ$$

$$\frac{\pi}{3} = 60^\circ = 1.0472 \text{ rad},\quad \frac{\pi}{4} = 45^\circ = 0.785398 \text{ rad},\quad \frac{\pi}{6} = 30^\circ = 0.523599 \text{ rad}.$$

Salvo este pequeño inconveniente (mínimo porque muchos de los ejemplos del curso se realizan con ángulos notables o múltiplos de ellos dados en radianes con ayuda de π) deberemos acostumbrarnos a trabajar con los radianes. Por ejemplo, el uso del radián simplifica muchos cálculos: si queremos calcular la longitud s de un arco (dado en radianes θ) de una circunferencia de radio r se utiliza la expresión $s = r.\theta$ y si lo que queremos es calcular el área del un sector circular se utiliza la expresión $A = r^2\,\dfrac{\theta}{2}$. Con grados las expresiones no son tan sencillas.

Más inconvenientes a la hora de trabajar con grados: 1° son $60'$ y $1'$ son $60''$, pero ahí se acabo el sistema sexagesimal. A partir de la unidad del segundo, sus divisores o submúltiplos no siguen la regla de "*cada* 60 *me llevo uno*" que nos recuerda a nuestra forma de medir el tiempo (horas, minutos y segundos). Todos sabemos que cuando se miden cantidades más pequeñas que el segundo se recure a las décimas, centésimas y milésimas de segundo, siendo este tipo de medición, de tipo decimal.

Este sistema decimal al que estamos más acostumbrados también se utiliza con los grados, "*pasando*", de alguna forma, de los minutos y segundos. Por ejemplo (y recurro a la medida del tiempo para que se vea más claro): ¿qué tiempo es 1.5h? son 1h y $30'$. Así 2.75h = 2h y $45'$, 0.1h = $6'$, 0.05h = $3'$ y 0.01h = $36''$. Cuestión: ¿cuánto tiempo en hora, minutos y segundos sería entonces 1.843h?

Luego, si cambiamos de unidad de medida del tiempo (en horas) a la medida de los ángulos (en grados) y eliminamos los minutos y segundos, la notación que aparecerá es la que se conoce como grados sexagesimales en formato decimal, más utilizada en ordenadores y calculadoras que la notación típica de grados, minutos y segundos.

Ejemplo 8 *Pasa* $25^\circ\ 33'\ 48''$ *a grados decimales*

Solución: *Esta claro que la parte entera de dicha cantidad son grados,* 25. *¿Y cómo transformamos la parte decimal? Diviendo por* 60 *los minutos y por* 3600 *los segundos. Vamos a utilizar una variable auxiliar* (*por ejemplo ang*) *para "guardar" el resultado obtenido por si después tenemos que usar dicho ángulo*

```
In[]:= ang=25+33/60+48/3600                    Out[]= 7669/300
In[]:= N[ang,7]                                Out[]= 25.56333
```

Luego $25^\circ\ 33'\ 48''$ *son* 25.56333 *grados decimales. Puede observarse como* 25.5° *son* $25^\circ\ 30'$ *y el resto...* 0.06333° *corresponden a* $3'\ 48''$.

Pero afortunadamente *Mathematica* posee un comando que puede realizar el cálculo directamente.

```
In[]:= FromDMS[{25,33,48}]                     Out[]= 7669/300
In[]:= FromDMS[{25,33.,48}]                    Out[]= 25.5633
```

La sintaxis del comando es: `FromDMS[{g,m,s}]` y devuelve el ángulo `{g,m,s}` en grados decimales, y si los argumentos se dan en valores enteros, la respuesta es en exacto o fracción, mientras que si algún argumento se da en formato decimal (por eso el punto decimal) la respuesta será en numérico o aproximado.

Y el proceso inverso, es decir, pasar de grados decimales a grados sexagesimales se realiza, también a mano (o a máquina).

Ejemplo 9 *Pasa* 25.56333° *grados decimales a grados sexagesimales.*

Solución: *Los* 25.56333° *grados decimales son ....* 25°*; tomamos solamente su parte decimal y ... si antes dividíamos por* 60 *ahora hay que multiplicar por* 60.

```
In[]:= 0.56333*60                              Out[]= 33.7998
```

33 *minutos. Y si seguimos tomando solamente la parte decimal y dividimos otra vez por* 60 (*no por* 3600*, porque dicho valor proviene de una cantidad ya dividida por* 60) *obtendremos los segundos.*

```
In[]:= (% - 33)*60                             Out[]= 47.988
```

Se utiliza `%` *o última salida* (*mejor que copiar a mano el valor*) *para minimizar el error de redondeo. Da* 47.988 *segundos. Y la respuesta final del ejemplo es* (*redondeando*) y *como cabía esperar*: $25^\circ\ 33'\ 48''$. *También, y directamente, con la orden* `DMSlist[ ]`.

```
In[]:= DMSlist[25.56333]                       Out[]= {25,33,47.998}
```

Dos **observaciones**: recordad la forma de "*introducir*" los grados sexagesimales en *Mathematica*: entre llaves y separando las unidades con comas; y que si queremos conocer la **sintaxis** de cualquier orden en *Mathematica*, se puede seleccionar dicha orden con el ratón y pulsar la tecla F1.

Hasta ahora hemos visto el "*trasiego*" entre grados sexagesimales y sus homónimos en formato decimal. Si **recordamos** ahora el valor numérico de `Degree` habremos cerrado el círculo, es decir, como pasar de grados a radianes y viceversa. `Degree` es realmente una **constante**, un **factor de conversión** que vale $\frac{\pi}{180}$ y que aparece "*detrás*" de la regla de tres:

$$\begin{array}{ccc} \pi & \longleftrightarrow & 180^\circ \\ x \text{ radianes} & \longleftrightarrow & y \text{ grados} \end{array}$$

Por ejemplo, para pasar 30 grados a radianes tendríamos que multiplicar 30 por $\frac{\pi}{180}$, o lo que es lo mismo, multiplicar 30 por `Degree`. Ya se vió la diferencia entre `30` y `30Degree` cuando calculamos `Cos[30]` y `Cos[30Degree]`. Entonces, en el proceso inverso de **pasar radianes a grados** habrá que multiplicar por $\frac{180}{\pi}$ o lo que es lo mismo, **dividir** por `Degree`.

Ejemplo 10 *Pasa* 180° *a radianes y expresa* 1 *radián en grados, minutos y segundos*

Solución:

```
In[]:= 180Degree//N                    Out[]= 3.14159
```

Y ahora, para expresar 1 *radián en grados, minutos y segundos vamos a utilizar la orden* `DMSList[ ]` *que convierte grados en decimales, en grados, minutos y segundos. Por ejemplo* 20.5° *son* $20^\circ\, 30'$.

```
In[]:= DMSList[20.5]                   Out[]= {20,30,0.}
In[]:= DMSList[1./Degree]              Out[]= {57,17,44.8062}
```

Es decir, $1\ rad = 57^\circ\ 17'\ 44.8062''$

Y por último recordemos también las funciones **trigonométricas inversas**. Son funciones reales de variable real (igual que las trigonométricas) luego su argumento será un número real y su **salida** (output) también, es decir, **radianes**. Luego, si estos radianes de salida se quisieran convertir a grados sexagesimales habría que realizar todo lo comentado en el ejemplo anterior, es decir, dividir por `Degree`.

Ejemplo 11 *Obtén el valor de* arctg(1.5) *y de* arctg(1)*, y convierte dichos ángulos, de radianes* (*resultado original*) *a grados sexagesimales.*

Solución: *El argumento* $x = 1.5$ *son radianes y el valor de* arctg(1.5) *también*:

```
In[]:= ArcTan[1.5]                     Out[]= 0.982794
```

Ahora, dividiendo el resultado por `Degree` *obtendremos el ángulo en grados decimales*

```
In[]:= %/Degree                        Out[]= 56.3099
```

Y si queremos pasar este ángulo a grados sexagesimales (*son* 56° *y ...*).

```
In[]:= DMSList[%]                      Out[]= {56,18,35.7569}
```

"*Todo junto*"

```
In[]:= DMSList[ArcTan[1.5]/Degree]     Out[]= {56,18,35.7569}
```

De igual forma para arctg(1)

```
In[]:= ArcTan[1.]/Degree               Out[]= 45.
```

4. Ejercicios propuestos

1. Obtén $\dfrac{20!}{10^4}$ y aproximaciones numéricas con 8 dígitos significativos para:

$$a)\ \sqrt[5]{8}\,, \qquad b)\ \left(\frac{3\pi-5}{5}\right)^{48}, \qquad c)\ e^3+\ln 580$$

2. ¿Qué resultado se obtiene si simplificas $\log_5 7+\log_{\frac{1}{5}} 7$? ¿Por qué?

3. Calcula de forma aproximada y con diez dígitos significativos

$$\mathrm{tg}(80^\circ\ 24'\ 15'')-\cos\left(\frac{5\pi}{4}\right)$$

4. Halla los valores de $\arccos(\frac{1}{2})$ y $\mathrm{arctg}(\frac{1}{2})$ en grados sexagesimales. Y utilizando la siguiente propiedad trigonométrica

$$\mathrm{arctg}(x)+\mathrm{arctg}\left(\frac{1}{x}\right)=\frac{\pi}{2}$$

da también el valor de $\mathrm{arctg}(2)$. Compruébalo con el *Mathematica* calculando dicho valor de la misma forma que se calculó $\mathrm{arctg}(\frac{1}{2})$.

Soluciones:

1) 243 290 200 817 664 $\quad a)$ 1.5157166 $\quad b)$ 0.0028328980 $\quad c)$ 26.448565

2) 0 $\quad$ 3) 6.622077717 $\quad$ 4) 60° y $26^\circ\ 33'\ 54.184''$. $\mathrm{arctg}(2)=63^\circ\ 26'\ 5.816''$

MATEMÁTICAS I

ELECTRÓNICOS

Práctica 2.2:
GRÁFICAS DE FUNCIONES. GRÁFICAS 2D

Prof: José Antonio Verdoy González

DEPARTAMENTO DE MATEMÁTICA APLICADA
ESCUELA TÉCNICA SUPERIOR
DE INGENIERÍA DEL DISEÑO
UNIVERSIDAD POLITÉCNICA DE VALENCIA

1. Introducción. Funciones explícitas e implícitas

En esta práctica describiremos diferentes posibilidades que ofrece *Mathematica* para obtener representaciones gráficas en dos dimensiones (2D) de funciones reales de variable real expresadas en forma *cartesiana*, *paramétrica* y también, en forma *polar*.

Además, *Mathematica* dispone de varias instrucciones para representar gráficamente curvas o elementos geométricos en el plano. Pero ¿qué diferencia hay entonces entre una curva 2D y la gráfica de una función? Depende del **tipo** de función, bien sea función **explícita** o función **implícita**.

Entonces, lo primero de todo será saber **distinguir** entre una función definida de forma explícita de las funciones definidas de forma implícita. Aunque, por supuesto que se debería recurrir a las definiciones precisas de ambos conceptos, aquí vamos a intentar diferenciarlos mediante un sencillo ejemplo: la curva o cónica *circunferencia*, **que como función explícita** no es una, sino **dos** funciones.

Sea la ecuación de la circunferencia $x^2 + y^2 = 4$, de centro el origen y radio $r = 2$. Esta igualdad es una función implícita y de forma genérica se suelen expresar de la forma $F(x, y) = 0$. Es decir: $F(x, y) = x^2 + y^2 - 4 = 0$. Ecuación o igualdad que **relaciona** las variables del plano cartesiano x e y. Cualquier relación o igualdad matemática, del tipo que sea, entre las variables x, y origina una *curva* plana (siempre y cuando existan valores numéricos que cumplan dicha igualdad, es decir, que podamos obtener una tabla de valores (x, y) cumpliendo la igualdad).

Y entonces surge la pregunta, ¿y si despejamos la variable y en función de la variable x, de la forma $y = f(x)$, para calcular más facilmente esta tabla de valores? A veces es posible, a veces no. Incluso siendo posible puede haber varias posibilidades para despejar y, es decir, para obtener las **funciones explícitas**.

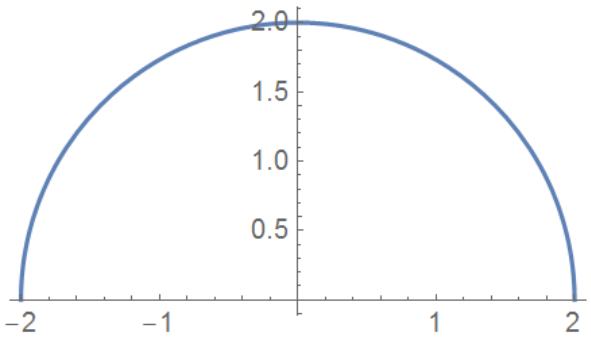
Siguiendo con nuestro ejemplo, despejando y de la ecuación *implícita* de la circunferencia, obtenemos **dos** funciones *explícitas*: $y_1 = f(x) = \sqrt{4 - x^2}$, $y_2 = g(x) = -\sqrt{4 - x^2}$. No son la misma, son "*parecidas*". Exactamente son **opuestas**. Pero lo más importante, son dos **funciones reales de variable real** diferentes y definidas en $[-2, 2]$. Y son **funciones** porque para cada valor de x perteneciente a su dominio, existe un **solo** valor de y, una sola imagen. Con la función implícita eso no ocurría.

Ejemplo 1 *A partir de la ecuación implícita $x^2 + y^2 = 4$, obtén las funciones explícitas correspondientes y represéntalas gráficamente.*

Solución: *Vamos autilizar para ello las ordenes* `Solve[ ]` *y* `Plot[ ]`. *La primera sirve para despejar variables y la segunda para representar gráficamente funciones explícitas.*

```
In[]:= Solve[x^2+y^2==4,y]
Out[]= {{y->-Sqrt[4-x^2]},{y->Sqrt[4-x^2]}}
In[]:= Plot[Sqrt[4-x^2],{x,-2,2},AspectRatio->Automatic]
```

obteniendo



Notas:

1. **Orden** `Solve[ ]`: Lo primero a tener muy en cuenta es que la ecuación hay que ponerla con un *doble igual* "`==`" y no con uno solo "`=`". Recordad, el símbolo "`=`" solo está permitido para asignar valores a variables o funciones. Segundo, hay que *expresar/decir* la variable que se quiere *despejar*. Y tercero, su resultado (ouput) es una *lista* (con uno, dos o más elementos) donde tampoco se utiliza el signo "`=`". Siempre podemos *copiar y pegar* de esa lista las expresiones que necesitemos.

2. **Orden** `Plot[ ]`: Como ya hemos dicho, sirve para **representar gráficamente funciones explícitas** del tipo $y = f(x)$ y como puede observarse, **no hace falta poner** "$y =$", basta con escribir la función a representar. Además, hay que escribir en dicha orden (y fijaros cómo) en qué intervalo o para qué rango de valores de x queremos representar la función. Con estos dos argumentos (función e intervalo), la orden `Plot[func,int]` ya funciona. Pero se ha utilizado además la opción `AspectRatio` que sirve para indicar la relación de escala entre la altura y la base del *rectángulo* o *caja* que contiene al gráfico, sin que las escalas de los ejes tengan que seguir dicha relación. Por defecto `Plot[ ]` emplea la relación $\frac{1}{GoldenRatio}$ donde *GoldenRatio* es el valor $\varphi = \frac{1+\sqrt{5}}{2} \approx 1.618$ (razón áurea). La opción `Automatic` utiliza la misma escala para ambos ejes. Si esta opción no se hubiese añadido, la semicircunferencia dibujada no se vería tal y como es, se vería achatada. Volved a ejecutar la orden sin la opción `AspectRatio->Automatic` y lo podréis observar.

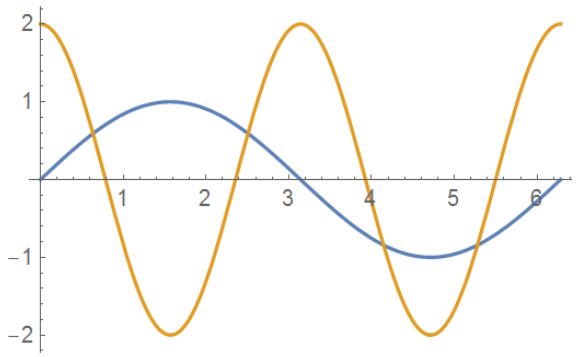
3. **Consejo**: Cada vez que os encontréis con una orden o comando nuevo, seleccionarlo, pulsar F1 y obtendréis una gran "*ayuda*" sobre su sintaxis, utilización y las diferentes y numerosas opciones que seguro tendrá. Y hablando de opciones, al igual que con `AspectRatio->Automatic`, cuando os aparezca una nueva opción en un comando y no sepáis exactamente *qué realiza*, eliminarla (con *cortar*) de la orden y volver a ejecutarla. Casi seguro que podréis saber para que sirve dicha opción.

Ejemplo 2 *Representa conjuntamente las funciones* $\operatorname{sen} x$ *y* $2\cos(2x)$ *con* $x \in [0, 2\pi]$.

Solución: *Para representar a la vez más de una función, éstas deben de ponerse entre llaves, como si de un* ***lista*** *de funciones se tratara.*

```
In[]:= Plot[{Sin[x],2Cos[2x]},{x,0,2Pi}]
```

obteniendo



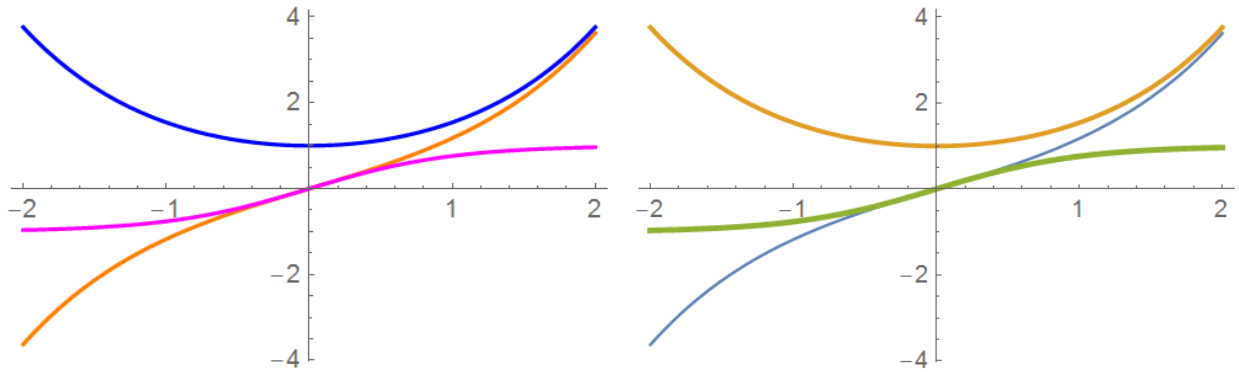
Surge la pregunta ¿cuál es cuál? Aunque con funciones ya conocidas, no debería de existir ninguna duda: la función $\operatorname{sen} x$ pasa por el origen de coordenadas, y una función como $\cos x$, si se multiplica por 2 pasa a estar acotada entre 2 y -2, *Mathematica* asigna un color a cada una de las funciones según el orden dispuesto. La primera en color azul, la segunda color entre ocre y mostaza, y la tercera color verde. **Importante**: según la versión del *Mathematica* utilizado, los *segundos* y demás colores pueden variar. Por suerte, el primero sigue siendo el azul. También podemos dibujar una a una y salir así de la duda.

También podemos asignar nosotros el color a cada gráfica (y así distinguirlas) con la opción `PlotStyle`, e incluso darles un poco más de grosor con la opción `Thickness`. Como siempre, se recomienda utilizar la ayuda del *Mathematica* para estudiar éstas y muchísimas más opciones.

Por ejemplo, con las siguientes opciones y funciones en `Plot[ ]`:

```
In[]:= Plot[{Sinh[x],Cosh[x],Tanh[x]},{x,-2,2},
            PlotStyle->{Orange,Blue,Magenta}]
In[]:= Plot[{Sinh[x],Cosh[x],Tanh[x]},{x,-2,2},
            PlotStyle->{Thickness->0.005,Thickness->0.008,Thickness->0.01}]
```

se originan las siguientes gráficas:

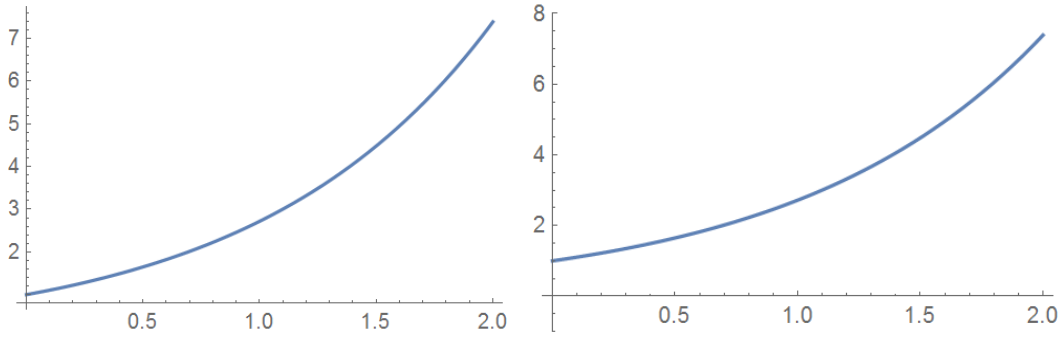


Ejemplo 3 *Representa la función e^x con $x \in [0, 2]$. Amplía el rango del eje de ordenadas lo necesario para observar claramente el eje abscisas y no confundirlo.*

Solución: *Con la sencilla orden* `Plot[Exp[x],{x,0,2}]` *se resuelve el ejemplo. Pero, puede observarse que el eje horizontal que aparece no es el eje de abscisas (eje OX, de ecuación $y = 0$). Es por ello que a veces es interesante elegir el rango del eje OY, al igual que se elige él del eje OX,* `{x,0,2}`*. Para ello se utiliza la opción* `PlotRange->{y0,y1}`.

```
In[]:= Plot[Exp[x],{x,0,2}]
In[]:= Plot[Exp[x],{x,0,2},PlotRange->{-1,8}]
```

obteniendo



2. Representación gráfica de funciones implícitas

Cuando nuestro objetivo sea representar gráficamente funciones implícitas de la forma $F(x, y) = 0$, el comando que hay que utilizar es `ContourPlot[ ]`. Y eso sí, utilizando como en el comando `Solve`, el *doble igual*. Su estructura es:

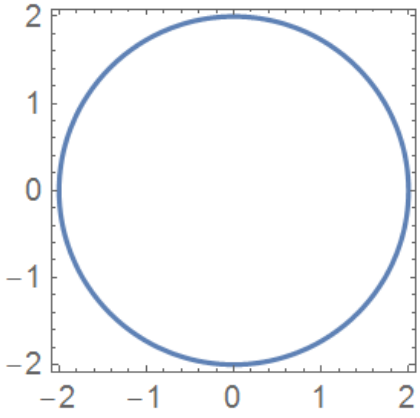
```
ContourPlot[F[x,y]==0,{x,a,b},{y,c,d}]
```

Ejemplo 4 *Representa gráficamente la circunferencia* $x^2 + y^2 = 4$.

Solución:

```
In[]:= ContourPlot[x^2+y^2==4,{x,-2,2},{y,-2,2}]
```

obteniendo



Ejemplo 5 *Representa conjuntamente y de color azul, las elipses con centro el origen y semiejes* $a = 4, b = 3$ *y reciprocamente* $a = 3, b = 4$.

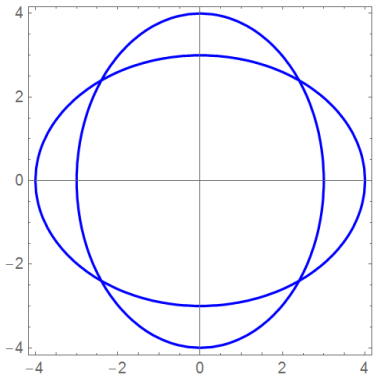
Solución: *La ecuación implícita de una elipse con centro el punto* (x_0, y_0) *y semiejes horizontal* a *y vertical* b *es*:

$$\frac{(x - x_0)^2}{a^2} + \frac{(y - y_0)^2}{b^2} = 1.$$

Así que las dos elipses que tenemos que representar son:

```
In[]:= ContourPlot[{x^2/16+y^2/9==1,x^2/9+y^2/16==1},{x,-4,4},{y,-4,4},
                   ContourStyle->Blue,Axes->True]
```

obteniendo



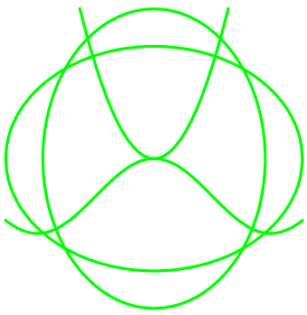
Dos cosas interesantes de esta nueva orden de *dibujo* `ContourPlot[ ]`: lo primero es que podemos conocer la curva representada sin más que acercar el puntero del ratón a la curva. ¡Compruébalo! Y dos, mientras que la orden `Plot[ ]` no admite en ningún caso funciones implícitas, la orden `ContourPlot[ ]` sí admite explícitas. Para ello hay que "*traducir*" la expresión general de una función explícita $y = f(x)$ al *Mathematica*: `y==f[x]`.

Ejemplo 6 *Al gráfico anterior, añádele las gráficas de las funciones* $\cos x - 1$ *y* x^2.

Solución:

```
In[]:= ContourPlot[{x^2/16+y^2/9==1,x^2/9+y^2/16==1,y==x^2,y==Cos[x]-1},
                   {x,-4,4},{y,-4,4},ContourStyle->Green,Frame->False]
```

obteniendo



3. Curvas 2D en coordenadas paramétricas

Después del estudio de las gráficas de funciones reales de variable real expresadas en forma explícita e implícita, haremos ahora una breve exposición de las peculiaridades que puede presentar el estudio de curvas expresadas por dos ecuaciones del tipo:

$$\{x = f(t), y = g(t)\} \text{ (curvas planas en paramétricas).}$$

Con frecuencia consideramos una curva en el plano como una línea trazada sobre un papel, tal como puede ser una línea recta, una curva parabólica o una circunferencia. Nos preguntamos ahora, ¿cómo podemos describir (analíticamente) una curva en el plano? Es evidente que debemos indicar de alguna manera los puntos por donde pasa, los puntos que forman la curva. En algunos casos podemos usar para ello las coordenadas cartesianas de los puntos $P(x, y)$ de la curva expresando y como una función de x, $y = f(x)$. También podemos dar una relación entre x e y que defina implícitamente a una variable en términos de la otra. Pero algunas curvas se describen mejor cuando las coordenadas x e y están dadas en términos de una tercera variable t llamada parámetro $\{x = f(t), y = g(t)\}$.

Imaginemos un objeto que se mueve en un plano y, a medida que transcurre el tiempo, describe un camino cerrado como el representado por la curva de la figura anterior. Si bien hacemos notar que esta curva no puede ser descrita por una ecuación de la forma $y = f(x)$ (¿por qué?) sabemos que las coordenadas x e y de la posición de la partícula dependen del instante de tiempo t. Por lo tanto, existirán funciones $f(t)$ y $g(t)$ dependientes de la variable (o parámetro) t, tales que $x = f(t)$ e $y = g(t)$. Este par de ecuaciones, que muchas veces es una forma muy conveniente para describir una curva, se llaman **ecuaciones paramétricas** de la curva en el plano:

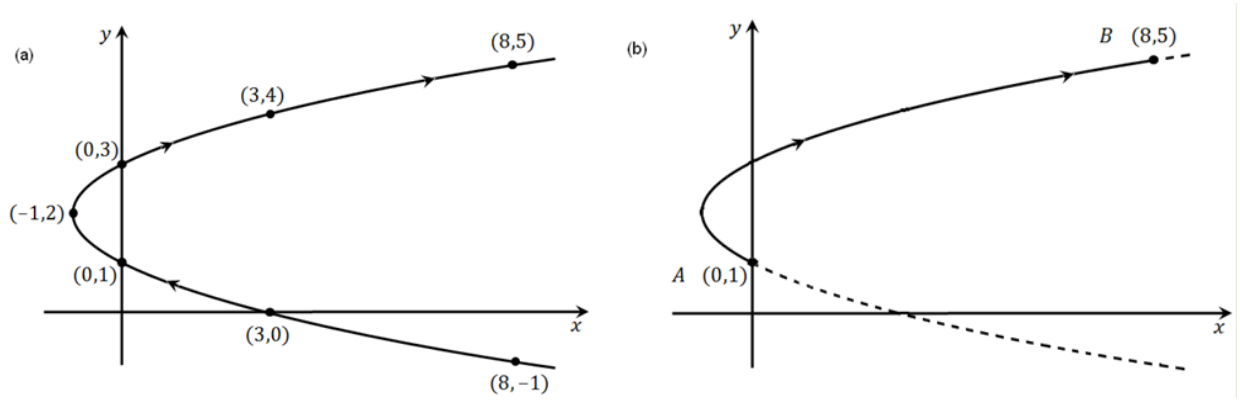
$$\begin{cases} x = f(t) \\ y = g(t) \end{cases}$$

Cada valor de t determina un punto (x, y) en el plano. Cuando t varía (en un intervalo de números reales) el punto $(x, y) = (f(t), g(t))$ se mueve generando una curva en el plano.

Ejemplo 7 *Describe y representa gráficamente (primero para $t \in \mathbb{R}$ y luego para $t \in [0, 4]$) la curva plana definida por las ecuaciones paramétricas*:

$$\begin{cases} x(t) = t^2 - 2t \\ y(t) = t + 1 \end{cases}$$

Solución: *A cada valor del parámetro $t \in \mathbb{R}$, le corresponde un único punto sobre la curva. Por ejemplo, para $t = 0$ se tiene $x(0) = 0$ e $y(0) = 1$, o sea que el punto de la curva correspondiente a $t = 0$ es $(0, 1)$. Podemos así evaluar x e y para varios valores del parámetro, por ejemplo, asignar a t los valores $-2, -1, 1, 2, 3, 4$, y luego situar los puntos $(x(t), y(t))$ en el plano. Si unimos estos puntos obtenemos una curva continua (figura* (a)*), en la que las flechas indican el sentido en el que se van generando los puntos de la curva a medida que t aumenta su valor. Si $t \in [0, 4]$, la curva que obtenemos es un "trozo" de la anterior ¿parábola? que empieza en el punto que corresponde al valor $t = 0$ del parámetro, o sea $A(0, 1)$, y termina en el punto que corresponde a $t = 4$, esto es, en $B(8, 5)$, como se muestra en la figura* (b).



Nota: Observando la figura, parece que la curva trazada es una parábola. ¿Cómo podemos comprobarlo? Una forma de hacerlo es reescribir las ecuaciones paramétricas de la curva usando (sólo) coordenadas cartesianas, esto es, buscar una relación entre x e y sin el parámetro t. Para ello debemos eliminar t en las ecuaciones dadas. En este ejemplo es posible hacerlo (no siempre lo es). Por ejemplo, despejando $t = y-1$ de la segunda ecuación y luego sustituyendo en la primera ecuación obtenemos la ecuación de una parábola:

$$x = t^2 - 2t = (y-1)^2 - 2(y-1) = y^2 - 4y + 3.$$

Tras este ejemplo teórico (o resuelto "*a mano*") podemos comentar que el comando del *Mathematica* que permite representar curvas en coordenadas parámetricas es:

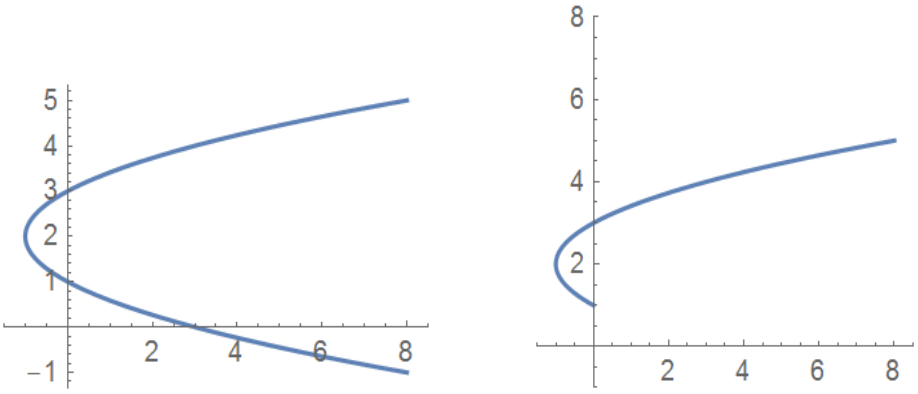
```
ParametricPlot[{x[t],y[t]},{t,t1,t2}]
```

Ejemplo 8 *Representa gráficamente la parábola del ejemplo anterior.*

Solución: *Primero debemos definir las funciones paramétricas que describen la curva. Se puede hacer por separado (dos instrucciones) o de una sola vez. Para esto último deben estar "recogidas" en una lista.*

```
In[]:= {x[t_]=t^2-2t,y[t_]=t+1}                    Out[]= {-2t+t^2,1+t}
In[]:= ParametricPlot[{x[t],y[t]},{t,-2,4}]
In[]:= ParametricPlot[{x[t],y[t]},{t,0,4},PlotRange->{-1,8}]
```

obteniendo



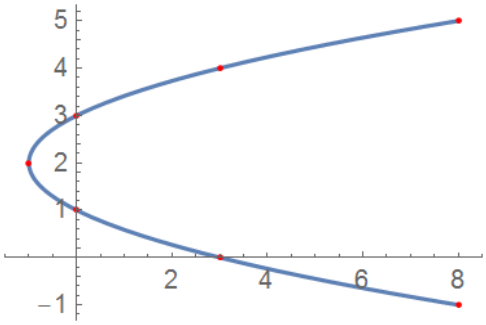
Incluso podríamos dibujar los puntos obtenidos anteriormente. Para ello utilizaremos el comando `ListPlot[lista]` que dibuja puntos y el comando `Show[g1,g2,g3,...]`

que sirve para representar conjuntamente diferentes gráficos $g1, g2$, etc. Muy útil cuando dichos gráficos **se han originado con ordenes distintas** (como es el caso) y a cada uno le hemos asignado un nombre. El rango o la amplitud el gráfico que genera el comando `Show` dependerá del **primer gráfico**, por lo que el orden de éstos en `Show` es muy importante.

Escribid:

```
In[]:= g1=ParametricPlot[{x[t],y[t]},{t,-2.1,4.1}];
In[]:= g2=ListPlot[{{8,-1},{3,0},{0,1},{-1,2},{0,3},{3,4},{8,5}},
                   PlotStyle->Red];
In[]:= Show[g1,g2]
```

y obtendremos:



Al hilo de los gráficos en coordenadas paramétricas, existe un comando en *Mathematica* muy apropiado para, entre otras muchas cosas, conocer el *sentido*, y *dónde* comienza y acaba (inicio y final) una curva paramétrica cuando $t \in [a, b]$; `Manipulate[ ]`.

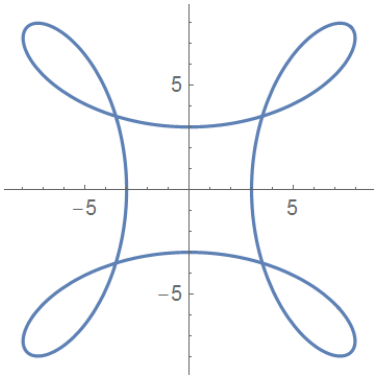
Ejemplo 9 *Representa gráficamente para* $t \in [0, 2\pi]$ *la curva paramétrica siguiente*:

$$\begin{cases} x(t) = 4\cos(-3t + \frac{\pi}{4}) + 7\cos(t + \frac{\pi}{4}) \\ y(t) = 4\,\mathrm{sen}(-3t + \frac{\pi}{4}) + 7\,\mathrm{sen}(t + \frac{\pi}{4}) \end{cases}$$

Solución: *Si escribimos*

```
In[]:= c1={4Cos[-3t+Pi/4]+7Cos[t+Pi/4],4Sin[-3t+Pi/4]+7Sin[t+Pi/4]};
In[]:= ParametricPlot[c1,{t,0,2Pi}]
```

obtenemos un "bonito lazo" cerrado.



Pero si agrupamos esta instrucción dentro de la orden `Manipulate[ ]`*, escribiendo*

```
In[]= Manipulate[ParametricPlot[c1,{t,0,k},PlotRange->9,Axes->False],
                 {{k,0.1,"t"},0.01,2Pi}]
```

podremos observar un cuadro interactivo donde existe un "botón" que al desplazarse va dibujando el lazo desde su inicio $t = 0$ hasta su final $t = 2\pi$.

4. Anexo: gráficas de objetos geométricos

Como curiosidad y sin otro afán más que el de aventurarse por las posibilidades gráficas del *Mathematica* comentar algunas ordenes o comandos de diversas y sencillas representaciones gráficas que tiene el programa.

Por ejemplo, y ya ha sido comentado, se pueden representar una cierta colección o lista de puntos mediante

`ListPlot[lista_puntos]`

Para obtener alguna figuras geométricas típicas se utiliza la función

`Graphics[objeto, opción]`

donde `objeto` es una de las diferentes figuras geométricas que tiene disponibles *Mathematica*, y `opción` una o varias de las posibilidades que se ofrecen para cada una de los `objetos`.

Entre los **objetos** más relevantes podemos citar ("buceando en la ayuda", la lista sería mucho mayor):

- `Point[{p,q}]` representa el punto de coordenadas cartesianas (p, q).
- `Line[{{p1,q1},{p2,q2},...}]` dibuja una línea poligonal uniendo los puntos de la lista.
- `Polygon[{{p1,q1},{p2,q2},...}]` representa un polígono cerrado cuyos vértices son los puntos de la lista.
- `Rectangle[{p1,q1},{p2,q2}]` dibuja un rectángulo donde (p_1, q_1) es el vértice inferior izquierdo y (p_2, q_2) es el vértice superior derecho.
- `Circle[{p,q},r]` traza una circuferencia de centro (p, q) y radio r. Puede utilizarse también para arcos de circunferencia y de elipse con las opciones adecuadas.
- `Text[texto,{p,q}]` escribe *texto* en el punto de coordenadas (p, q).

Recordad, primero la orden `Graphics[ ]` y dentro el objeto. Y si queremos ver varios juntos, con la misma orden, representaremos una lista de objetos (los objetos entre `{ }`).

5. Ejercicios propuestos

1. Representa conjuntamente la elipse $\dfrac{(x-2)^2}{16} + \dfrac{(y+1)^2}{9} = 1$ y la curva sinusoide definida por $4\,\mathrm{sen}(2x)$.

2. Utiliza los comandos `Table[ ]` y `Plot[ ]` para representar en el intervalo $[0, \pi]$ la familia de curvas

$$\{k \operatorname{sen} x : k = -3, -2.9, -2.8, \ldots, 2.9, 3\}$$

3. Representa gráficamente la curva cerrada denominada *astroide*, y que viene definida en coordenadas paramétricas por

$$\begin{cases} x(t) = 2\cos^3 t \\ y(t) = 2\,\mathrm{sen}^3\, t \end{cases} \quad t \in [0, 2\pi]$$

4. Con ayuda de los comandos `Table[ ]` y `Plot[ ]` representa gráficamente en $[-4, 4]$ la familia de curvas exponenciales

$$\{e^{kx^2} : k = -3, -2.9, -2.8, \ldots, 2.9, 3\}$$

   Utiliza la opción `PlotRange` para que el rango del eje de ordenadas sea el intervalo $[0, 6]$ y elimina los ejes cartesianos del dibujo.

5. Visita la dirección web

   http://www.neoteo.com/rostros-de-famosos-generados-con-wolframalpha

MATEMÁTICAS I

ELECTRÓNICOS

Práctica 2.3:
RESOLUCIÓN DE ECUACIONES Y CÁLCULO DE RAÍCES

Prof: José Antonio Verdoy González

DEPARTAMENTO DE MATEMÁTICA APLICADA
ESCUELA TÉCNICA SUPERIOR
DE INGENIERÍA DEL DISEÑO
UNIVERSIDAD POLITÉCNICA DE VALENCIA

1. Resolución de ecuaciones

Mathematica es capaz de resolver gran número de ecuaciones de forma exacta, es decir, proporcionando soluciones racionales e irracionales obtenidas de forma analítica por métodos directos como, por ejemplo, las soluciones de una ecuación de segundo grado. No obstante, en multitud de ocasiones no se conoce la manera de hallar todas las soluciones de una ecuación y es preciso recurrir a métodos numéricos de aproximación. Aquí veremos los **tres comandos** más utilizados para resolver ecuaciones y obtener esas soluciones, tanto de forma exacta como de forma aproximada.

Y lo primero que hay que recordar es la utilización del signo igual "=", la **igualdad** en *Mathematica*. A la hora de introducir una igualdad entre dos expresiones para, por ejemplo, construir una ecuación o representar graficamente una función implícita, **se tiene que utilizar un doble igual** "==". La utilización de **un solo igual** es para las asignaciones o identificaciones (por ejemplo, en variables `z=2+3I`, o en funciones `f[x_]=Sin[x]`).

1.1. El comando `Solve[ ]`. Resolución exacta

El comando `Solve[ ]` además de despejar variables, por ejemplo, la variable y respecto de x como ya hemos visto, tambien "*despeja*" una sola variable de una ecuación, vamos, **resuelve ecuaciones**. Y las resuelve de forma **exacta**. Por ejemplo, para una ecuación de segundo grado aplica la famosa fórmula $-b \pm \sqrt{b^2 \cdots}$ y obtiene las dos soluciones, pero repetimos, de forma exacta. Análogo para las ecuaciones de tercer y cuarto grado.

Ejemplo 1 *Resuelve las siguientes ecuaciones*

$a)\ \ 2x^2-3x-9=0, \quad b)\ \ 4x^3+x^2-11x+6=0, \quad c)\ \ 3x^4-5x^3+5x^2-5x+2=0.$

Solución: *Escribiendo*

```
In[]:= Solve[2x^2-3x-9==0,x]                Out[]= {{x->-(3/2)},{x->3}}
In[]:= Solve[4x^3+x^2-11x+6==0,x]           Out[]= {{x->-2},{x->3/4},{x->1}}
In[]:= Solve[3x^4-5x^3+5x^2-5x+2==0,x]
Out[]= {{x->-I},{x->I},{x->2/3},{x->1}}
```

se obtienen todas las soluciones de forma exacta, hasta las complejas. Estas últimas pueden ser omitidas de la solución, es decir, si solamente queremos obtener soluciones reales tenemos que escribir

```
In[]:= Solve[3x^4-5x^3+5x^2-5x+2==0,x,Reals]      Out[]= {{x->2/3},{x->1}}
```

Importante: Como puede observarse, la sintaxis utilizada es `Solve[ecuac,var]` y aunque se puede omitir la variable cuando la ecuación es de una sola variable, se aconseja ponerla, porque para otros usos del comando `Solve[ ]` sí es necesario que aparezca. También se puede observar la forma en que es "*devuelta*" la solución o soluciones, vamos, el `Output` de esta orden: una lista de elementos donde la solución viene dada después de una $x \to \cdots$, **nunca** de forma asignada $x = \cdots$.

Una primera pregunta que se nos puede plantear es ¿`Solve[ ]` resuelve también sistemas de ecuaciones? Ante esta cuestión y cualquier otra que se os plantee, ¡ADELANTE, probad, **de forma lógica** con el comando en cuestión, y a ver que ocurre!

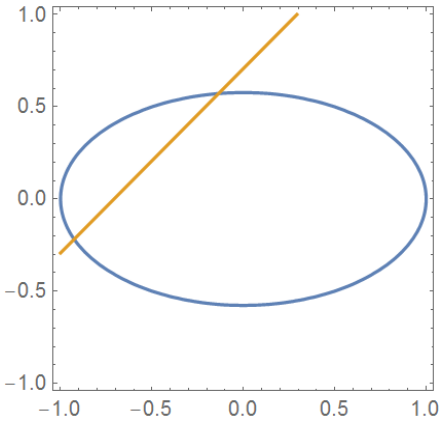
Ejemplo 2 *Halla los puntos de corte de la elipse* $x^2+2y^2=1$ *con la recta* $y=x+\frac{\sqrt{2}}{2}$.

Solución:

```
In[]:= Solve[{x^2+2y^2==1,y==x+Sqrt[2]/2},{x,y}]
Out[]= {{x->0,y->1/Sqrt[2]},{x->-((2Sqrt[2])/3),y->-(1/(3Sqrt[2]))}}
```

Copiando y pegando la orden `Solve[ ]`, cambiando a la orden `ContourPlot[ ]` y añadiendo el rango del rectángulo para el gráfico, siempre estaremos *informados visualmente* de nuestro trabajo analítico.

```
In[]:= ContourPlot[{x^2+2y^2==1,y==x+Sqrt[2]/2},{x,-1,1},{y,-1,1}]
```



1.2. El comando `NSolve[ ]`. Resolución aproximada

Puede ocurrir que una vez hallada de forma exacta la solución de una ecuación nos preguntemos ¿y cuánto vale aproximadamente dicha solución? ¿cuánto vale la solución anterior $y = -\frac{1}{3\sqrt{2}}$? Podríamos hallarlo con el comando ya conocido `N[-(1/(3Sqrt[2]))]` pero si son varios los valores a aproximar, mejor que hacerlo uno a uno podemos utilizar el comando `NSolve[ ]`.

Con la misma sintaxis que su "*hermano*" para resolución exacta `Solve[ ]`, `NSolve[ ]` se permite el lujo de no tener que poner el doble igual en la ecuación (no es necesario), pero siempre que la ecuación esté dada de forma **normal**, es decir, igualada a 0.

Ejemplo 3 *Resuelve de forma exacta la ecuación* $x^3 - 3x^2 + 6x - 7 = 0$. *Y la vista de la solución exacta resuélvela de forma aproximada.*

Solución:

```
In[]:= NSolve[x^3-3x^2+6x-7,x]
Out[]= {{x->0.591134-1.87123 I},{x->0.591134+1.87123 I},{x->1.81773}}
```

Acabamos de observar como numerosas ecuaciones algebraicas se tendrán que resolver de forma aproximada. Y siguiendo con el ejemplo anterior, veamos como elegir el número de cifras significativas de estas soluciones aproximadas. Cuestión que no es baladí, ya que en Ingeniería, cuando se dan valores aproximados es muy importante conocer el error que se está cometiendo o bien, la diferencia que existe entre el valor exacto y el aproximado. Para ello, hay que elegir la opción `WorkingPrecision` **dentro** del comando u orden que estemos utilizando para obtener valores o soluciones aproximadas. Esta opción también es válida para otros comandos que ya iremos conociendo.

Ejemplo 4 *Obtén, de forma aproximada y con* 15 *cifras significativas, la única solución real de la ecuación* $x^3 - 3x^2 + 6x - 7 = 0$.

Solución: *Para hallar directamente lo que se pide hay que escribir*

```
In[]:= NSolve[x^3-3x^2+6x-7,x,Reals,WorkingPrecision->15]
Out[]= {{x->1.81773167388682}}
```

Ejemplo 5 *Halla, de forma aproximada, las tres raíces cúbicas de* $-\sqrt{8}$.

Solución:

```
In[]:= NSolve[x^3==-Sqrt[8]]
Out[]= {x->-1.41421},{x->0.707107-1.22474 I},{x->0.707107+1.22474 I}}
```

Pregunta: ¿Es normal que la soluciones complejas aparezcan por parejas y sean conjugadas?

Nota: Las dos ordenes expuestas hasta ahora son perfectas para ecuaciones algebraicas, hasta grado cuarto en resolución exacta, y de cualquier grado en resolución aproximada. Y también para ecuaciones "*sencillas*" con funciones trigonométricas, exponenciales o logarítmicas.

Ejemplo 6 *Resuelve de forma aproximada la ecuación* $x^6 + 6x - 7 = 0$. *Inténtalo de forma exacta y observa el diferente* ***Output*** *del Mathematica.*

Solución: *Utilicemos* `NSolve[ ]` *para resolver la ecuación de forma aproximada y después con* `Solve[ ]`.

```
In[]:= NSolve[x^6+6x-7==0]
Out[]= {{x->-1.59687},{x->-0.618967-1.41718 I},{x->-0.618967+1.41718 I},
        {x->0.917402-0.995661 I},{x->0.917402+0.995661 I},{x->1.}}
In[]:= Solve[x^6+6x-7==0]
```

Mathematica contesta de esta forma porque no existe una fórmula exacta para la resolución de ecuaciones de grado mayor o igual a 5 y utiliza (situa el ratón encima de las soluciones) *expresiones propias del programa* para dar la solución. Cuando nos aparezcan *respuestas* de este tipo mejor utilizar métodos aproximados como `NSolve[ ]`.

Ejemplo 7 *Resuelve la ecuación* $2^x + 2^{x+1} + 2^{x+2} = a$, *con* $a = 14$ *y* $a = 15$.

Solución: *Comentar que vamos a utilizar la opción* `Reals` *porque sino los comandos utilizados emplearían métodos de resolución en variable compleja, concepto éste que no se estudia este curso.*

```
In[]= Solve[2^x+2^(x+1)+2^(x+2)==14,x,Reals]
Out[]= {{x->1}}
In[]= Solve[2^x+2^(x+1)+2^(x+2)==15,x,Reals]
Out[]= {{x->(Log[3]+Log[5]-Log[7])/Log[2]}}
In[]:= %//N
Out[]= {{x->1.09954}}
```

Pero en cuanto las ecuaciones se "*complican*" un poco, estos dos comandos "*no están a la altura*". Vamos, que no son los apropiados para cierto tipo de ecuaciones.

Ejemplo 8 *Intenta resolver la ecuación*

$$\operatorname{tg} x + e^x = -2.$$

Solución: *Tanto* `Solve[ ]` *como* `NSolve[ ]` *nos dan el mismo mensaje avisándonos de que esta ecuación no puede resolverse con dichos métodos.*

```
In[]:= Solve[Tan[x]+Exp[x]==-2,x]
```

Solve: This system cannot be solved with the methods available to Solve.

```
Out[]= Solve[e^x + Tan[x]==-2,x]
```

Es por ello que necesitamos del comando "definitivo" que resuelva cualquier ecuación que se le ponga por delante.

1.3. El comando `FindRoot[ ]`. ¡El comando definitivo!

El comando `FindRoot[ecuación,{x,a}]`, siendo `a` (tenedlo muy presente, porque muy a menudo se os olvida ponerlo) un valor inicial donde el método numérico que subyace bajo este comando, **arranca**, **inicia** su algoritmo para encontrar **una sola solución aproximada** de la ecuación a resolver.

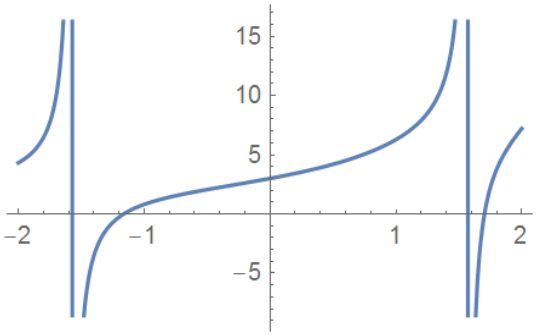
Y a la pregunta de ¿cómo elegir ese valor de `a`? se puede responder *a la gallega*, ¿sabéis mirar en una gráfica? Pues eso, ese valor de `a` será un valor próximo a la solución de la ecuación que podremos observar si la representamos gráficamente, mejor dicho, si representamos gráficamente la función $f(x)$ que define la ecuación en forma normal $f(x) = 0$ que queremos resolver. Las **soluciones** o **ceros** de la ecuación son las abscisas de los puntos de corte de la gráfica con el eje OX y que también se conocen con el nombre de **raíces de la función**.

Ejemplo 9 *Resuelve en el intervalo* $[-2, 2]$ *la ecuación del ejemplo anterior.*

Solución: *Primero, como ya hemos comentado, vamos a representar gráficamente la "ecuación" en forma normal y en el intervalo pedido para observar "dónde" están y cuántas son las raíces que buscamos.*

```
In[]:= Plot[Tan[x]+Exp[x]+2,{x,-2,2}]
```

obteniendo



Luego podemos observar claramente dos raíces: una cerca del valor -1 *y otra del* 2. *Voilà los valores a elegir para* `a`. *Por favor, no confundir como raíces las dos asíntotas verticales que aparecen en los valores de* x *donde no existe o no está definida la función* $\operatorname{tg} x$. *Entonces escribiendo (probad a copiar-pegar-modificar desde el comando* `Plot[ ]`)

```
In[]:= FindRoot[Tan[x]+Exp[x]+2,{x,-1}]
Out[]= {x->-1.16267}
```

tenemos una primera aproximación a la raíz negativa. Y ahora, ¡vamos a por la segunda raíz, pero con poderío!, exactamente con 16 *cifras significativas.*

```
In[]:= FindRoot[Tan[x]+Exp[x]+2,{x,2},WorkingPrecision->16]
Out[]= {x->1.703470620226527}
```

¿Y qué os parece obtener las dos raíces a la vez?

```
In[]:= FindRoot[Tan[x]+Exp[x]+2,{x,{-1,2}},WorkingPrecision->10]
Out[]= {x->{-1.162670856,1.703470620}}
```

Nota: Como ya hemos comentado, el comando `FindRoot[ ]` lleva implementado un algoritmo de cálculo numérico para obtener de forma aproximada raíces de funciones. Pues bien, este algoritmo puede ser elegido por el usuario entre algunos que ofrece *Mathematica* (métodos de bisección, de la secante, de la tangente, etc.) y eso podéis mirarlo, como siempre que queráis ampliar conocimientos, en la **ayuda** del comando.

Y para terminar vamos a ver que puede pasar si utilizamos `FindRoot[ ]` a *ciegas*.

Ejemplo 10 *Resuelve la ecuación* $e^x + 1 = 0$.

Solución: *Vamos a elegir el valor de* `a` *sin ningún "miramiento", cero, porque sí.*

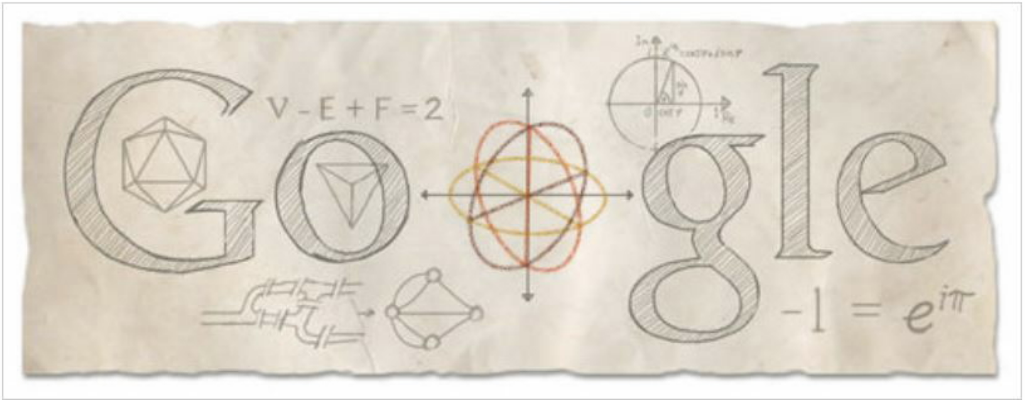
```
In[]:= FindRoot[Exp[x]+1==0,{x,0}]
Out[]= {x->-38.8617}
```

Y así nos va, obtenemos una solución que no es válida y entre medias, un mensaje de aviso consistente. ¿Por qué? Porque le hemos pedido a `FindRoot[ ]` *que resuelva una ecuación sin solución (al menos en los números reales) y eso no se debe hacer. Podemos observar como esta ecuación no tiene solución sin más que representar la gráfica de la función. Incluso si recurrimos a la propiedad de la función exponencial* e^x *que es positiva en todo su dominio, puede deducirse que* $e^x = -1$ *es imposible. Luego debemos contestar que no existe solución real para esta ecuación.*

Pero esta igualdad es muy famosa. Estamos hablando de la **identidad de Euler**, la ecuación más famosa de las matemáticas. En ella encontramos los conceptos de suma, multiplicación, exponenciación y la identidad. Y aparecen también cinco números fundamentales: el cero, el uno, π, el número e y la unidad imaginaria i.

$$e^{i\pi} + 1 = 0.$$

Entre otros sitios, esta igualdad puede verse en un doodle animado (gif) del 15 de abril de 2013, aniversario del nacimiento del gran matemático suizo Leonhard Euler.



Y podemos preguntarnos, ¿siendo una ecuación tan famosa, `FindRoot[]` no *puede* con ella? Como siempre que utilizemos `FindRoot[]` todo dependerá del *primer paso*. Eligiendo la unidad imaginaria i como valor de `a`, ya que la solución es compleja obtenemos

```
In[]:= FindRoot[Exp[x]+1==0,{x,I}]
Out[]= {x->1.03943*10^(-17)+3.14159 I}
In[]:= Chop[%]                          Out[]= {x->0.+3.14159 I}
```

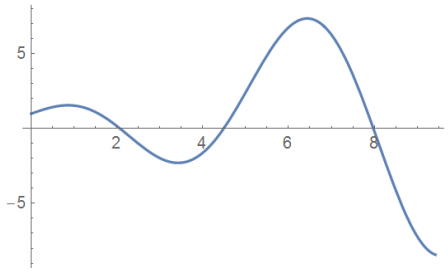
valor aproximado de la solución $\pi\, i$.

Nota: En métodos aproximados, valores del orden de 10^{-16}, 10^{-17}, etc. que surgen normalmente por errores de redondeo, se consideran valores iguales a cero. Ver `Chop[]`.

Ejemplo 11 *Halla las tres primeras raíces positivas de la ecuación* $x\cos x = -1$.

Solución: *Primero vamos a realizar una representación gráfica de la ecuación en forma normal en* $\mathbb{R}^+$ *para una elección adecuada de los valores de inicio.*

```
In[]:= Plot[x*Cos[x]+1,{x,0,3Pi}]
```



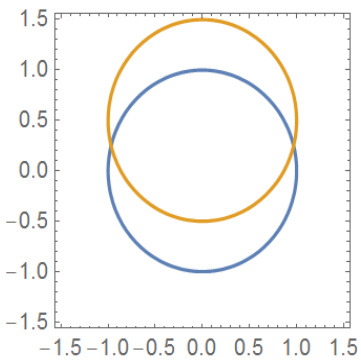

```
In[]:= FindRoot[x*Cos[x]+1,{x,{2,4.5,8}},WorkingPrecision->8]
Out[]= {x->{2.0739328,4.4876696,7.9796311}}
```

Ejemplo 12 *Resuelve de forma aproximada el sistema de ecuaciones*

$$\begin{cases} x^2+y^2=1 \\ \\ x^2+(y-\frac{1}{2})^2=1. \end{cases}$$

Solución: *Representemos gráficamente ambas circunferencias con el fin de elegir puntos cercanos a las soluciones. Y como ejercicio, obtened vosotros la segunda solución.*

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




```
In[]:= FindRoot[{x^2+y^2==1,x^2+(y-1/2)^2==1},{x,1},{y,0.25}]
Out[]= {x->0.968246,y->0.25}
```

2. Ejercicios propuestos

1. Resuelve las siguientes ecuaciones polinómicas:

   a) $x^3 - x^2 - x - 1 = 0$.

   b) $x^4 + 2x^3 - x - 1 = 0$.

   c) $x^4 + 2 = 0$.

2. Resuelve, de forma aproximada, las siguientes ecuaciones:

   a) $2x \ln x = 1$.

   b) $e^{-x} = \operatorname{sen}^2(x^3)$.

3. Utiliza `FindRoot[]` de forma adecuada para obtener el valor aproximado del número e con 15 dígitos significativos. Para ello ayúdate de la ecuación

   $$\ln x = 1.$$

   ¡Y recuerda! $\ln e = 1$ mientras que $\ln 1 = 0$. ¿Y qué vale $\ln 0$?

   Nota: También podríamos haber obtenido el mismo resultado con `N[E,15]`.

4. Averigua el número de raíces de la ecuación $e^{-2x} = \cos x$ en el intervalo $[-1, 6]$. Hállalas con 15 cifras significativas y sustituye cada una de ellas en la ecuación para comprobar que realmente son raíces de ésta.

5. Resuelve de forma aproximada el sistema

   $$\begin{cases} y = \ln x \\ \\ x^2 + y = 5x - 5. \end{cases}$$

Soluciones

2) *a*) 1.42153 *b*) Tiene infinitas soluciones. De menor a mayor y solamente las dos primeras: $x_1 = 0.886626$ $x_2 = 1.3778$.

3) 2.71828182845905

4) Tres raíces: $x_1 = 0$ $x_2 = 1.52325380992561$ $x_3 = 4.71246966688068$.

5) Tres soluciones, los puntos del plano de coordenadas (x, y):

$$(1.626,\, 0.486122), \quad (2.9217,\, 1.07217) \ \text{ y } \ (0.0069768,\, -4.96516).$$