Programación con Mathematica.

José Salvador Cánovas Peña.
Departamento de Matemática Aplicada y Estadística.
Universidad Politécnica de Cartagena.

15 de noviembre de 2010

Índice general

Capítulo 1

Introducción a Mathematica

Mathematica es un programa que permite hacer cálculos matemáticos complicados con gran rapidez. Para entendernos, es como una calculadora gigante a la que no sólo podemos pedirle que haga cálculos numéricos, sino que también hace derivadas, cálculo de primitivas, representación gráfica de curvas y superficies, etcétera.

Abordaremos en esta práctica una iniciación a Mathematica partiendo desde cero, intentando poner de manifiesto su utilidad a la hora de trabajar con expresiones matemáticas complicadas, bien sean éstas numéricas o simbólicas, permitiendo hacer operaciones en poco coste de tiempo y con bastante facilidad.

Pretendemos con esta práctica introducir al alumno en el manejo de este potente programa que puede servirle de utilidad en futuros cálculos que deba realizar. Esencialmente vamos a aprender a utilizar el programa, por lo que esta práctica trata de explicar como pedirle a Mathematica que haga aquello que nosotros deseamos. Además, el programa puede utilizarse para corregir los problemas propuestos al alumno en la clase de problemas.

A pesar de la utilidad del programa, debemos hacer hincapié en el hecho de que es necesario por parte del alumno un conocimiento matemático teórico de todas las funciones y sentencias que vamos a usar. Por ejemplo, aunque una calculadora multiplica números con suma facilidad, sólo nos percatamos de su potencia en cuanto conocemos dicha operación y somos capaces de realizarla de un modo mucho más lento. Con Mathematica ocurre lo mismo. Sólo conociendo teóricamente las operaciones que Mathematica realiza nos percataremos de su utilidad.

1.1. Preliminares

Cuando se arranca Mathematica, aparece una pantalla blanca vacía. En ella podemos escribir aquellas operaciones que queremos que realice. Una vez tecleada la operación, hemos de pulsar las teclas *shift + enter* para obtener el resultado. Por ejemplo, supongamos que queremos hacer la operación $2+2$. Teclearemos entonces

$$2+2$$

en la pantalla. A continuación pulsamos *mayúsculas + enter* o la tecla *intro* en el teclado numérico y a continuación aparecerá en pantalla

$$\begin{aligned} \text{In}[1] &:= 2+2 \\ \text{Out}[1] &= 4. \end{aligned}$$

Todas las operaciones realizadas por el programa cuando se pulsan las teclas mayúsculas + enter tienen asignadas un número de entrada marcado por In[·] y el mismo número de salida cuando se realiza la operación marcado por Out[·]. Podrá aparecer únicamente un nú mero de entrada, como veremos posteriormente. Al ir explicando las diferentes operaciones que Mathematica realiza, iremos escribiéndolas en la forma en que el programa lo escribe en la pantalla de ordenador.

Además de la suma se pueden realizar las siguientes operaciones algebraicas como si se tratara de una calculadora:

$x+y$	suma de números
$x-y$	resta de números
x/y	división de números
$x\ y\ x*y$	producto de números
$x\hat{}y$	potencia x^y

Cuando Mathematica realiza alguna de las siguientes operaciones, por ejemplo $1/3+2/7$, operará estos números ofreciendo siempre su valor exacto, es decir, se tiene

$$\begin{aligned} \text{In}[2] &:= 1/3+2/7 \\ \text{Out}[2] &= \frac{13}{21}. \end{aligned}$$

Sin embargo, a veces nos es más útil tener el valor de este número expresado con cifras decimales. Para ello se tienen las sentencias

$$x//\mathrm{N}\ \mathrm{N}[x]$$
$$\mathrm{N}[x, n].$$

Las primeras escriben el número x con seis cifras significativas, mientras que la segunda escribe dicho número con un número n de cifras significativas que nosotros prefijamos (en la versión 4.0 del programa y posteriores esta última sentencia no siempre funciona del modo deseado). Por ejemplo, si escribimos

$$\begin{aligned} \mathrm{In}[3] &:= 1/3 + 2/7\ //\mathrm{N} \\ \mathrm{Out}[3] &= 0{,}619048, \end{aligned}$$

obtendremos el resultado con 6 cifras significativas. Si por el contrario escribimos

$$\begin{aligned} \mathrm{In}[4] &:= N[1/3 + 2/7, 10] \\ \mathrm{Out}[4] &= 0{,}619047619 \end{aligned}$$

lo obtendremos con un número 10 cifras significativas.

En caso de las operaciones numéricas también tendremos una valor numérico aproximado con seis cifras significativas si en la operación escribimos algún número en forma decimal. Así, al teclear

$$\begin{aligned} \mathrm{In}[5] &:= 1./3 + 2/7 \\ \mathrm{Out}[5] &= 0{,}619048. \end{aligned}$$

Mathematica distingue así entre operaciones algebraicas exactas y operaciones numéricas aproximadas.

1.2. Paréntesis, corchetes y llaves

Mathematica distingue entre paréntesis, corchetes y llaves. Cada uno de estos elementos realiza una labor bien diferenciada en la estructura interna del programa. A grosso modo podemos indicar las siguientes generalidades:

- Los paréntesis se usan en las operaciones algebraicas para indicar la preferencia a la hora de hacer las operaciones. Así el paréntesis de

$$\begin{aligned} \text{In}[6] &:= (1+3)/7 \\ \text{Out}[6] &= \frac{4}{7} \end{aligned}$$

se usa para indicar que primero hacemos la suma 1+3 y luego dividimos entre 7. Hemos de señalar que Mathematica sigue el orden conocido de preferencia sobre las operaciones. Asi por ejemplo, si escribimos

$$\begin{aligned} \text{In}[7] &:= 1+3/7 \\ \text{Out}[7] &= \frac{10}{7} \end{aligned}$$

vemos como el resultado cambia notablemente al realizarse en primer lugar la división y posteriormente la suma.

- Los corchetes $[\cdot]$ se usan para escribir el argumento de una función bien sea matemática, bien sea una operación específica del programa. Por ejemplo la función $\sin x$ se escribe Sin[x], y para escribir un número x real con seis cifras significativas escribimos N[x].

- Las llaves $\{\cdot\}$ se utilizan para asignar valores numéricos a las variables, por ejemplo a la hora de calcular límites de funciones. También se usan para construir conjuntos o listas de objetos matemáticos, como por ejemplo matrices o vectores.

En general es conveniente tener claro en qué momento se han de emplear los paréntesis, los corchetes y las llaves, ya que si confundimos su uso y escribimos por ejemplo Sin{x} o Sin(x) en lugar de Sin[x], el programa nos lo hará saber mandándonos un mensaje de error de color azul.

1.3. Errores

Puede ocurrir que al teclear una operación en Mathematica y pulsar las teclas mayúscular + enter, el programa nos devuelva una salida conteniendo frases de color azul. Esto ocurre cuando hay algún tipo de error o problema que el programa detecta. Estos errores pueden ser básicamente de dos tipos:

- Errores en la sintaxis de una sentencia. Por ejemplo al escribir $[1+2]*3$ en vez de $(1+2)*3$ o $\mathrm{N}(\pi)$ en vez de $\mathrm{N}[\pi]$.
- Errores producidos porque la expresión matemática o la operación realizada tiene algún problema, aunque esté bien escrita. Por ejemplo, si intentásemos calcular el determinante de una matriz no cuadrada.

Otras veces, el programa puede devolver un resultado erróneo aunque no nos escriba frases azules. Es decir el programa no detecta ningún error a pesar de que éste existe. Por ésto es necesario saber qué estamos esperando de la operación que hemos pedido que el programa nos haga para así criticar el resultado y valorarlo en su justa medida. No debeis nunca de olvidar que el programa calcula rápidamente, pero es tonto.

1.4. Funciones matemáticas de interés

Mathematica posee una serie de funciones matemáticas predefinidas que se escriben del siguiente modo:

$\mathrm{Sqrt}[x] = \sqrt{x}$	$\mathrm{Exp}[x] = e^x$
$\mathrm{Log}[x] = \log x$	$\mathrm{Log}[b, x] = \log_b x$
$\mathrm{Sin}[x] = \sin x$	$\mathrm{Cos}[x] = \cos x$
$\mathrm{ArcSin}[x] = \arcsin x$	$\mathrm{ArcCos}[x] = \arccos x$
$\mathrm{Tan}[x] = \tan x$	$\mathrm{ArcTan}[x] = \arctan x$
$n!$ = factorial de n	$\mathrm{Round}[x] = \textit{parte entera de } x$
$\mathrm{Abs}[x] = \lvert x \rvert$	$\mathrm{FactorInteger}[n] = \textit{factores primos de } n$

Es importante destacar que hemos de escribir las funciones tal y como se detalla en la anterior tabla, respetando la sintaxis totalmente. Mathematica distingue entre letras mayúsculas y minúsculas, y todas las funciones

empiezan con letra mayúscula. Entonces podemos calcular

$$\begin{aligned} \text{In[7]} &:= \text{Sqrt[16]} \\ \text{Out[7]} &= 4 \\ \text{In[8]} &:= \text{Sqrt[2]} \\ \text{Out[8]} &= \sqrt{2} \\ \text{In[9]} &:= \text{Sqrt[2] //}N \\ \text{Out[9]} &= 1{,}41421 \\ \text{In[10]} &:= \text{N[Sqrt[7], 10]} \\ \text{Out[10]} &= 2{,}6457513111. \end{aligned}$$

Las constantes matemáticas más interesantes son:

$$\begin{aligned} \text{Pi} &= \pi \simeq 3{,}14159 \\ \text{E} &= e \simeq 2{,}71828 \\ \text{Infinity} &= \infty \\ \text{I} &= i = \sqrt{-1} \\ \text{Degree} &= \textit{conversión de grados a radianes} \end{aligned}$$

Por ejemplo, para calcular el seno de 20 grados escribiríamos

$$\begin{aligned} \text{In[11]} &:= \text{Sin[20 Degree] //N} \\ \text{Out[11]} &= 0{,}34202. \end{aligned}$$

Si quisiéramos mayor precisión en las anteriores constantes, debemos escribir

$$\begin{aligned} \text{In[12]} &:= \text{N[Pi,10]} \\ \text{Out[12]} &= 3{,}1415926536, \end{aligned}$$

que nos proporciona un valor del número π con diez cifras decimales.

1.5. Aprovechando cálculos anteriores

A veces es posible que tengamos que hacer una sucesión de cálculos consecutivos de manera que cada nueva operación se base en la anterior. Parece necesaria entonces una sentencia que nos remita a resultados anteriores sin

tener que escribir la operación de nuevo. Dicha sentencia es %. Por ejemplo, si queremos calcular $\cos(\sin 20°)$ tendríamos que calcular primero $\sin 20°$, para después calcular el coseno de dicha cantidad. Esta operación podríamos hacerla del modo siguiente:

$$\begin{aligned} \text{In[13]} &:= \text{Sin[20 Degree] //N} \\ \text{Out[13]} &= 0{,}34202 \\ \text{In[14]} &:= \text{Cos[\%]} \\ \text{Out[14]} &= 0{,}942079. \end{aligned}$$

Aquí Cos[%] nos calcula el coseno del resultado obtenido en la salida 13. Para remitirnos a un resultado obtenido dos pasos antes debemos escribir % %, y para resultados obtenidos k pasos antes escribiremos k símbolos %. Para remitirnos a un resultado obtenido en la salida n podemos también escribir %n. Por ejemplo Cos[%13] también nos remite a la salida 13.

1.6. Definición de variables y funciones

Supongamos que tenemos que hacer una serie de cálculos en los cuales interviene repetidamente una constante, como por ejemplo la constante de la gravitación universal, o al estudiar valores particulares de la función $f(x) = (1{,}45)^x + \sin x$. Es útil entonces definir variables con estos valores y funciones nuevas para minimizar el tiempo de escritura y hacer las operaciones de forma más ágil.

Las variables pueden ser designadas por letras o por sucesiones de letras. Supongamos que queremos definir la constante de la gravitación universal $G = 6{,}67 \times 10^{-11}$ con Mathematica. Entonces deberíamos hacer

$$\begin{aligned} \text{In[15]} &:= G = 6{,}67 \times 10\hat{} - 11 \\ \text{Out[15]} &= 6{,}67 \times 10^{-11} \end{aligned}$$

Si ahora tenemos dos cuerpos de masa 3 kilogramos separados a una distancia de 10 metros, la fuerza con la que se atraen dichos cuerpos se calcula

$$\begin{aligned} \text{In[16]} &:= G * 3\hat{}2/(10\hat{}2) \\ \text{Out[16]} &= 6{,}003 \times 10^{-12}. \end{aligned}$$

Para desposeer a G de su valor debemos teclear

$$\text{In[17]} := G = .$$

o bien

$$\text{In}[18] := \text{Clear}[G].$$

Algunas letras están asignadas ya por defecto por Mathematica y no pueden ser utilizadas para definir variables. Una de ellas es N. Si intentásemos escribir N=2, el programa devolverá una sentencia azul de error.

A la hora de trabajar con variables en Mathematica, hemos de tener en cuenta las siguientes reglas. Si x e y son dos variables que hemos definido con anterioridad, entonces

- $x\ y$ representará el producto $x \cdot y$.
- xy es una nueva variable formada por dos letras.
- $5x$ es el producto de 5 por x..
- $x\hat{}2y$ es el producto x^2y y no x^{2y}.

Por otra parte, si en una misma línea queremos definir varias variables, o escribir varias expresiones debemos separar estas con ";". Por ejemplo

$$\begin{aligned} \text{In}[19] &:= x = 1; y = 2; z = x + y \\ \text{Out}[19] &= 3. \end{aligned}$$

ha asignado el valor 1 a x, 2 a y y 3 a z, y sólo ha escrito el último valor. Si al final de la última expresión ponemos también ";" la operación no proporciona ninguna salida, es decir la expresión

$$\text{In}[20] := x = 1; y = 2; z = x + y;$$

no proporciona a continuación un Out[20] al pulsar mayúsculas + enter y asigna los mismos valores que en la sentencia anterior.

Para definir nuevas funciones hemos de usar la siguiente estructura:

$$\text{nombrefunción}[x_1\_, x_2\_, ..., x_n\_] := \text{expresión}.$$

El símbolo _ se usa para indicar que la letra que lo antecede es una variable de la función. Por ejemplo, para definir la función $f(x) = (1{,}45)^x + \sin x$ debemos escribir

$$\text{In}[21] := f[x\_] := 1{,}45\hat{}x + \text{Sin}[x].$$

Entonces si tecleamos

$$\begin{aligned} \text{In}[22] &:= f[1] \\ \text{Out}[22] &:= 3{,}18975. \end{aligned}$$

obtenemos el valor de dicha función en 1. Para eliminar la función debemos escribir

$$\text{In}[23] := \text{Clear}[f, x]$$

indicando tanto la variable como el nombre de la función.

Hemos visto como introducir funciones escalares, pero a menudo es necesario trabajar con funciones vectoriales, por ejemplo la función

$$f(x, y, z) = (xy, x \sin z),$$

que como vemos es una función $f : \mathbb{R}^3 \to \mathbb{R}^2$. Para ello definimos la función como

$$\text{In}[24] := f[x\_, y\_, z\_] := \{x * y, x * \text{Sin}[z]\}.$$

Si tecleamos ahora

$$\begin{aligned} \text{In}[25] &:= f[1, 1, Pi] \\ \text{Out}[25] &:= \{1, 0\} \end{aligned}$$

obtenemos un vector del plano. Para obtener las funciones coordenadas debemos escribir

$$\begin{aligned} \text{In}[26] &:= f[1, 1, Pi][[1]] \\ \text{Out}[26] &:= 1 \end{aligned}$$

o

$$\begin{aligned} \text{In}[27] &:= f[1, 1, Pi][[2]] \\ \text{Out}[27] &:= 0 \end{aligned}$$

según queramos trabajar con la primera o segunda coordenada.

1.7. Derivadas de funciones

Supongamos que tenemos una función de una variable real $f\,(x_1, x_2, ..., x_n)$ a la que queremos calcular su derivada o derivada parcial respecto de alguna de sus variables. El comando que realiza ese cálculo con Mathematica es

$$\mathrm{D}[f, x] \text{ ó } \mathrm{D}[f, x_i].$$

Por ejemplo si queremos calcular la derivada de $f\,(x) = \sin x$ escribiremos

$$\begin{aligned} \mathrm{In}[24] &:= \mathrm{D}[\mathrm{Sin}[x], x] \\ \mathrm{Out}[24] &= \mathrm{Cos}[x], \end{aligned}$$

especificando tanto la función como la variable respecto de la cual vamos a derivar. Para calcular la derivada parcial con respecto a la variable y de la función $f(x, y) = \sin(x + y)$ debemos escribir

$$\begin{aligned} \mathrm{In}[25] &:= \mathrm{D}[\mathrm{Sin}[x + y], y] \\ \mathrm{Out}[25] &= \mathrm{Cos}[x + y]. \end{aligned}$$

Para calcular la derivada n–ésima de $f\,(x)$, hemos de proceder con el comando

$$\mathrm{D}[f, \{x, n\}].$$

Así la segunda derivada de $f\,(x) = \sin x$ se calcula tecleando

$$\begin{aligned} \mathrm{In}[26] &:= \mathrm{D}[\mathrm{Sin}[x], \{x, 2\}] \\ \mathrm{Out}[26] &= -\mathrm{Sin}[x] \end{aligned}$$

y $\frac{\partial^3 f}{\partial y^3}$ de la función $f(x, y) = \sin(x + y)$ sería

$$\begin{aligned} \mathrm{In}[27] &:= \mathrm{D}[\mathrm{Sin}[x + y], \{y, 3\}] \\ \mathrm{Out}[27] &= -\mathrm{Cos}[x + y]. \end{aligned}$$

Si ahora queremos calcular derivadas parciales de funciones respecto de diferentes variables hemos de indicarlo del modo siguiente

$$\mathrm{D}[f, x_1, x_2, ..., x_n].$$

Así por ejemplo $\dfrac{\partial^2 f}{\partial x \partial y}$ de la función $f(x, y) = \sin(x+y)$ se calcula escribiendo

$$\begin{aligned} \mathrm{In}[28] &:= \mathrm{D}[\mathrm{Sin}[x + y], x, y] \\ \mathrm{Out}[28] &= -\mathrm{Sin}[x + y]. \end{aligned}$$

1.8. Representación gráfica de funciones

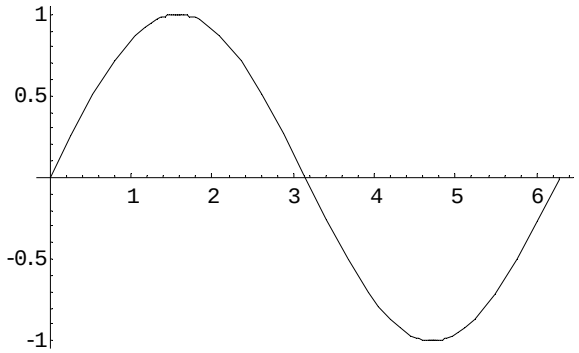
Mathematica permite hacer representaciones gráficas de funciones de una y varias variables. Para ello hemos de darle tanto la función, como el dominio de definición de ésta.

Para la representación gráfica de funciones reales de variable real, tenemos el comando

$$\text{Plot}[f[x], \{x, x_0, x_1\}],$$

donde indicamos la función, la variable de la función, y un intervalo $[x_0, x_1]$ donde hacer la representación. Así, para representar la función $f(x) = \sin x$ en el dominio $[0, 2\pi]$ escribimos

$$\text{In}[30] := \text{Plot}[\text{Sin}[x], \{x, 0, 2\text{Pi}\}].$$

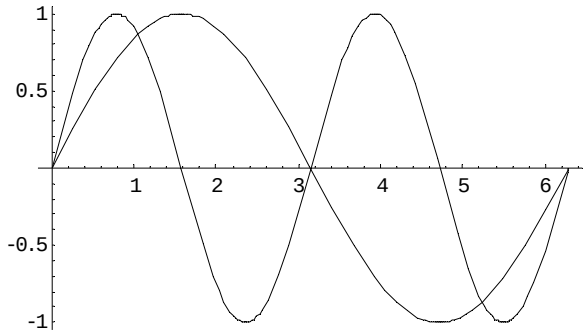


Para representar varias funciones a la vez hemos de escribir todas las funciones que deseemos representar entre llaves y separadas por comas, es decir

$$\text{Plot}[\{f_1[x], f_2[x], ..., f_n[x]\}, \{x, x_0, x_1\}].$$

Si escribimos entonces

$$\text{In}[31] := \text{Plot}[\{\text{Sin}[x], \text{Sin}[2x]\}, \{x, 0, 2Pi\}]$$



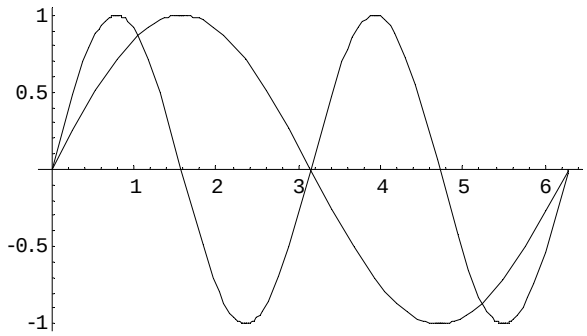
generaremos una representación gráfica simultánea de las funciones $\sin x$ y $\sin 2x$.

Para volver a representar gráfica una función ya representada previamente tenemos el comando

$$\text{Show}[\,\%n].$$

Así, si escribimos

$$\text{In}[33] := \text{Show}[\,\%31],$$



obtenemos una nueva representación gráfica simultánea de las funciones $\sin x$ y $\sin 2x$.

Capítulo 2

Programación con Mathematica

A continuación vamos a explicar algunas sentencias que se utilizan para hacer programas con Mathematica. Las principales herramientas que vamos a necesitar para hacer los programas son las siguientes:

- ¿Cómo establecer comunicación entre el ordenador y quien ejecuta el programa?
- ¿Cómo contruir bucles con Mathematica?
- ¿Cómo definir funciones y variables?
- ¿Cómo interrumpir la ejecución de un programa cuando éste esté bloqueado?

Vamos a explicar como hacer cada una de estas cosas, para poder construir nuestro programa. En primer lugar, vamos a contestar a la primera pregunta: para interrumpir un programa o una evaluación de Mathematica basta con pulsar las teclas

$$\text{Alt} + \text{coma}.$$

Para las otras cuestiones, se necesita una explicación más amplia que a continuación desarrollamos.

2.1. Aritmética de computadora

Esta sección ha sido desarrollada a partir de los programas del profesor David López.

2.1.1. Aritmética exacta y aritmética de redondeo

La aritmética de redondeo es mucho más rápida que la exacta, y por eso se utiliza en Cálculo Numérico. Para ilustrar esta diferencia veamos el siguiente experimeto para el cual necesitaremos la función Timing, que mide el tiempo de CPU que se tarda en realizar la suma

$$\sum_{i=1}^{n} \frac{1}{i^2}.$$

Para ello, escribimos

$$Timing[Sum[1/i\hat{}2, \{i, 1, n\}];]$$

y

$$Timing[Sum[1./i\hat{}2, \{i, 1, n\}]]$$

y damos valores a n. En el primer valor se trabaja con precisión infnita y da la salida

$$\{t, Null\},$$

donde t es el tiempo que se tarda en realizar la operación. La segunda expresión se trabaja con aproximaciones (observar el punto detrás del 1) y de nuevo t es el tiempo empleado en hacer los cálculos. Puede verse como a medida que aumentamos n aumenta mucho más el tiempo empleado con precisión infinita.

2.1.2. Aritmética de doble precisión

Cualquier calculadora u ordenador expresa los números reales con notación exponencial, habitualmente utilizando base 2. Al ser finitos los recursos de los que se disponen también han de serlo el número de cifras que se pueden utilizar. Se llama epsilon de la máquina al menor número positivo que sumado a uno es distinto de uno. En caso de Mathematica, el epsilon de la maquina ($MachineEpsilon) es $2{,}220446049250313 \times 10^{-16}$. Por ejemplo, si consideramos la desigualdad

$$1 + 2^{-n} > 1,$$

ésta siempre se cumple cuando trabajamos en precisón infinita. Pero si escribimos

$$1. + 2^{-n},$$

y trabajamos con precisión finita, se tiene que a partir de $n = 53$, la operación anterior nos dará el valor 0, mientras que para $n = 52$ obtenemos el epsión de la máquina, que equivalen a 16 cifras significativas en base 10. Esto no quiere decir en absoluto que no se pudan manejar números menores que 10^{-16} o mayores que 10^{16}, puesto que tenemos bits disponibles para el exponente. Esto nos da los siguientes números mínimo ($MinMachineNumber) y máximo ($MaxMachineNumber):

$$2{,}2250738585072014 * 10^{-308}$$

y

$$1{,}7976931348623157 * 10^{308}.$$

Así pues, el rango de números admisibles está entre 10^{-308} y 10^{308}. Cualquier número que sea mayor que el admisible producirá un .ºverflow"(error de desbordamiento), mientras que un número positivo menor que el admisible es tratado como 0, lo que se conoce como ünderflow"(error de subdesbordamiento).

2.1.3. Errores de redondeo

La utilización de una aritmética finita puede producir un error al no poder contar con el número suficiente de cifras para representar un número. Este error, conocido como de redondeo, puede ser debido a diferentes causas. Algunos números no pueden ser representados con un número finito de cifras decimales, entrando en esta categoría los racionales periódicos y los irracionales. Por ejemplo, $\sqrt{2.}^2 - 2$ debería valer cero, pero si lo ejecutamos obtenemos el valor

$$4{,}44089 * 10^{-16}.$$

Los errores de redondeo también pueden producirse porque fracciones decimales exactas pueden ser fracciones binarias periódicas. En esta categoría entran los racionales en cuyo denominador aparezcan potencias de 5 y (tal vez) de 2. El ejemplo más destacable es el de 1/10, fracción claramente exacta en aritmética decimal, pero que no lo es en aritmética binaria. Para observar este error de redondeo realizaremos un bucle en el que se mostrarán los resultados obtenidos al ir añadiendo sucesivamente la cantidad 1/10. Por ejemplo, si escribimos

$$Sum[1/10, \{n, 1, m\}] - Sum[1./10, \{n, 1, m\}],$$

vemos que para $m = 1$ obtenemos cero, pero para $m = 10$ ya se aprecian errores de redondeo.

2.2. Operaciones y definición de variables.

2.2.1. Definición de variables y funciones

Como sabemos, para definir una variable numérica basta con tomar una letra o conjunto de letras e igualarlas al valor deseado. Por ejemplo, si queremos asignar a la letra q el valor 6,5, escribiremos en la pantalla

$$q = 6{,}5$$

los que producirá la siguiente respuesta en la pantalla del ordenador

$$\begin{aligned} \text{In}[1] &:= q = 6{,}5 \\ \text{Out}[1] &= 6{,}5 \end{aligned}$$

Si lo que queremos es definir una función, hemos de hacerlo indicando la variable entre corchetes según el siguiente esquema:

$$\text{Nombre}[\text{var\_}] := \text{función}.$$

Así, para definir la función $f(x) = e^x \cos x$ hemos de escribir

$$\begin{aligned} \text{In}[2] &:= \text{f}[\text{x\_}] := \text{Exp}[x] * \text{Cos}[x] \\ \text{Out}[2] &= \text{Exp}[x] * \text{Cos}[x]. \end{aligned}$$

Otro tipo de variables bastante interesantes son las vectoriales. Podemos definir una variable vectorial, por ejemplo de dimensión tres haciendo la siguiente operación

$$\begin{aligned} \text{In}[3] &:= a[1] = 5 \\ \text{Out}[3] &= 5 \end{aligned}$$

$$\begin{aligned} \text{In}[4] &:= a[2] = 2 \\ \text{Out}[4] &= 2 \end{aligned}$$

$$\begin{aligned} \text{In}[5] &:= a[1] = 9 \\ \text{Out}[5] &= 9 \end{aligned}$$

y habremos introducido el vector $(5, 2, 9)$. Si queremos saber el valor de la variable a hemos de escribir

$$\begin{aligned} \text{In}[6] &:= ?a \\ & \quad \textit{Global}\,'a \\ & \quad a[1] = 5 \\ & \quad a[2] = 2 \\ & \quad a[3] = 9 \end{aligned}$$

o bien

$$\begin{aligned} \text{In}[7] &:= \text{Table}[a[i], \{i, 3\}] \\ \text{Out}[7] &= \{5, 2, 9\}. \end{aligned}$$

Podemos dejar sin asignar valores al vector. Si hacemos

$$\text{In}[8] := \text{Clear}[a[2]]$$

y

$$\begin{aligned} \text{In}[7] &:= \text{Table}[a[i], \{i, 3\}] \\ \text{Out}[7] &= \{5, a[2], 9\} \end{aligned}$$

dejando el valor de $a[2]$ vacio.

Para borrar las variables o las funciones utilizaremos la sentencia Clear[var o func]. Escribiendo

$$\text{Clear}[q]$$

desposeemos a q del valor 6.5 asignado anteriormente.

2.2.2. Funciones vectoriales

Las funciones pueden ser tanto escalares como vectoriales. Anteriormente hemos visto como definir funciones reales de variable real, pero se pueden definir funciones vectoriales de varias variables reales, como por ejemplo

$$f(x, y, z) = (e^{xy}z, 2xy).$$

La sintaxis es análoga al caso unidimensional, es decir, dicha función se definiría como

$$f[x\_, y\_, z\_] := \{Exp[x * y] * z, 2 * x * y\},$$

donde las variables se separan por comas y las llaves se utilizan para definir el vector de $\mathbb{R}^2$. Una vez introducida la función, si tecleamos por ejemplo

$$f[1, 1, 0]$$

obtendríamos como salida

$$Out[1] := \{0, 2\},$$

que se corresponde con el vector $(0, 1)$. La función anterior tiene dos funciones coordenadas

$$\begin{aligned} f_1(x, y, z) &= e^{xy}z, \\ f_2(x, y, z) &= 2xy, \end{aligned}$$

y si queremos trabajar con ellas individualmente no es necesario volverlas a introducir. Una vez $f[x, y, z]$ ha sido introducida en el programa, podemos utilizar las funciones coordenadas anteriores con la sintaxis

$$f[x, y, z][[1]]$$

para la primera y

$$f[x, y, z][[2]]$$

para la segunda. Por ejemplo, en el caso anterior, si tecleamos

$$f[1, 1, 0][[1]]$$

obtendríamos como salida

$$Out[2] := 0.$$

2.2.3. Operaciones

Como sabemos las cuatro operaciones básicas que tenemos son

$+$	Suma
$-$	Resta
$*$	Multiplicación
$/$	División

junto con el símbolo =. Podemos combinar esta operaciones para obtener otros tipos de operaciones y estructuras lógicas muy usadas en programación. Como las más interesantes resumimos las siguientes:

Operaciones de Test

==	Igual
! =	Distinto
<	Menor que
>	Mayor que
<=	Menor o igual que
>=	Mayor o igual que

Operaciones Lógicas

! o Not	Negación
&&	Y
||	O
True	Verdad
False	Falsedad

Por ejemplo, si escribimos $10 < 8$, aparecerá en pantalla

$$\begin{aligned} \text{In}[1] &:= 10 < 8 \\ \text{Out}[1] &= \text{False} \end{aligned}$$

indicando la falsedad de la afirmación. Si escribimos $7 > 4$ y $2 \neq 3$ aparecerá

$$\begin{aligned} \text{In}[2] &:= 7 > 4 \;\&\&\; 2 \; ! = 3 \\ \text{Out}[2] &= \text{True} \end{aligned}$$

indicando que las dos sentencias son verdaderas. Si ponemos una verdadera y una falsa y escribimos

$$\begin{aligned} \text{In}[3] &:= 7 > 4 \;\&\&\; 2 \; = 3 \\ \text{Out}[3] &= \text{False} \end{aligned}$$

mientras que

$$\begin{aligned} \text{In}[4] &:= 7 > 4 \;||\; 2 \; = 3 \\ \text{Out}[4] &= \text{True} \end{aligned}$$

Finalmente

$$\begin{aligned} \text{In}[5] &:= \text{Not}[7 < 4] \\ \text{Out}[5] &= \text{True} \end{aligned}$$

dado que lo contrario de $7 < 4$ es $7 > 4$, que es verdadero.

Otras combinaciones de símbolos que pueden resultar útiles son las siguientes:

$x++$	Aumenta el valor de x una unidad
$x--$	Disminuye el valor de x una unidad
$++x$	Preincrementa x una unidad
$--x$	Predisminuye x una unidad
$x+=k$	Aumenta el valor de x k unidades
$x-=k$	Disminuye el valor de x k unidades
$x*=k$	Multiplica x por k
$x/=k$	Divide x por k

Por ejemplo, si hacemos $x = 7$ y realizamos las siguientes operaciones, obtenemos

$$\begin{aligned} \text{In}[6] &:= x++ \\ \text{Out}[6] &= 7 \end{aligned}$$

donde el valor de x ahora es 8, pero en pantalla se escribe su antiguo valor. Si hubiéramos escrito

$$\begin{aligned} \text{In}[6] &:= ++x \\ \text{Out}[6] &= 8 \end{aligned}$$

ahora el valor de x sigue siendo 8 y el programa escribe el valor actual. De un modo parecido funciona con $--$. Si escribimos

$$\begin{aligned} \text{In}[6] &:= x+=2 \\ \text{Out}[6] &= 9 \end{aligned}$$

el valor de x será 9, igual que el que aparece en pantalla.

2.3. Construcción de programas con Mathematica.

Uno de los temas centrales en los programas que vamos a realizar es la construcción de bucles o procesos iterativos, los cuales han de ejecutarse un número de terminado de veces. Para la una correcta creación de bucles suele bastar con la sentencia For, que a continuación pasamos a explicar. También puede resultar de utilidad la sentncia If cuando se quieren poner condiciones a la hora de elegir entre diferentes opciones. Existen más sentencias para programar en Mathematica que puede encontrarse en las ayudas del programa.

2.3.1. For

Tomemos por ejemplo un proceso iterativo de orden uno $x_{n+1} = f(x_n)$ de los que aparecen al aplicar un método numérico de orden uno. Imaginemos por ejemplo que se trata de

$$x_{n+1} = \sqrt{x_n} + 1,$$

y que partimos de la condición inicial $x_0 = 1$. Por alguna razón, estamos interesados en saber el valor de esta sucesión en 1000, esto es x_{1000}. Esto supone hacer 1000 iteraciones del proceso iterativo, que como vamos a ver puede programarse con relativa facilidad en Mathematica con el comando **For**.

La sintaxis de **For** es la siguiente:

$$\text{For}[\text{inicio}, \text{test}, \text{incremento}, \text{expresión}],$$

que partiendo de un valor inicial, evalúa la expresión a la vez que incrementa el valor inicial hasta que el test falla. En particular, el cálculo de x_{1000} puede hacerse con el siguiente programa:

$$x = 1.; \text{For}[n = 1, n < 1001, n++, x = \text{Sqrt}[x] + 1]$$

Es preciso notar lo siguiente en el ejemplo anterior:

- Hemos escrito "$x = 1.$" y no "$x = 1$". El punto al final del uno indica que estamos trabajando con aproximaciones y no con precisión infinita. Esto hace que los tiempos de computación sean muy diferentes.

Mathematica puede trabajar con precisión infinita, pero el tiempo que puede tardar en hacer los cálculos puede ser enorme, e incluso puede que dichos cálculos no puedan realizarse.

- El For ejecuta desde 1 a 1000 el anterior proceso iterativo.
- Si se ejecuta el anterior programa, éste no devuelve ninguna salida. El For no escribe nada en pantalla. Para ello necesitamos el comando Print. Si al programa anterior le añadimos la sentencia

$$\text{Print}[\text{"}\textit{El valor en } 1000 \textit{ es igual a}\text{"}, x]$$

es decir ejecutamos el programa

$$x = 1.; \text{For}[n = 1, n < 1001, n++, x = \text{Sqrt}[x] + 1];$$
$$\text{Print}[\text{"}\textit{El valor en } 1000 \textit{ es igual a }\text{"}, x]$$

Obtenemos la salida

$$\textit{Out}[1] = \textit{El valor en } 1000 \textit{ es igual a } 2{,}61803$$

Nótese en la sintaxis del Print el papel jugado por las comilla, que sirven para escribir el texto, y la coma, que sirve para separar el texto del valor numérico a la hora de que Print se ejecute.

Si ahora nos pregruntamos por el valor de x_{100}, nos damos cuenta de que debemos ejecutar de nuevo el programa, ya que éste se ha perdido durante la ejecución. Es decir, el valos x_{100} se calculó en el paso 100, pero se borró al ejecutar el 101. Podemos almacenar todos los valores en un vector o Array, cuya sintaxis es la siguiente

$$\text{Array}[var, n]$$

o

$$\text{Array}[var, m, n]$$

donde var es el nombre de la variable que vamos a usar, n es el número de datos o longitud del vector y, en la segunda sentencia, m es el ordinal del dato inicial. Por defecto en la primera sentencia m toma el valor 1. Para obtener un valor concreto del vector, por ejemplo el k–ésimo, escribimos

$var[k]$. Modificamos el programa anterior para almacenar todos los valores obtenidos hasta llegar a 1000 de la siguiente manera

$$x0 = 1.; \text{Array}[x, 1000];$$
$$\text{For}[n = 1, n < 1001, n++, x[n] = \text{Sqrt}[x0] + 1, x0 = x[n]];$$
$$\text{Print}[\text{"}El\ valor\ en\ 1000\ es\ igual\ a\ \text{"}, x0]$$

o bien

$$\text{Array}[x, 0, 1001]; x[0] = 1.;$$
$$\text{For}[n = 1, n < 1001, n++, x[n] = \text{Sqrt}[x[n-1]] + 1];$$
$$\text{Print}[\text{"}El\ valor\ en\ 1000\ es\ igual\ a\ \text{"}, x[1000]]$$

En cualquiera de los dos casos, para obtener x_{100} bastará con ejecutar $x[100]$.

Finalmente, si queremos modifica bien el proceso iterativo o bien la condición inicial, con el anterior programa tendríamos que modificarlos sobre el mismo. Estas variables y funciones suele ser más util definirlas al principio, de manera que si queremos cambiarlas se pueda hace fácilmente y de forma global. Así, el programa anterior se podría refinar como:

$$\begin{aligned} f[x\_] &:= \text{Sqrt}[x] + 1; \\ n &= 1000; \text{Array}[x, 0, n+1]; x[0] = 1.; \\ \text{For}[i &= 1, i < n+1, i++, x[i] = \text{Sqrt}[x[i-1]] + 1]; \\ &\quad \text{Print}[\text{"}El\ valor\ en\ 1000\ es\ igual\ a\ \text{"}, x[n]] \end{aligned}$$

2.3.2. If

La sentencia "If" se emplea para establecer elementos condicionantes en un programa. Las diferentes sintaxis con las que puede manejarse son las siguientes.

$$\text{If}[\text{condición}, f].$$

Si la condición es verdadera el programa ejecutará f, y si es falsa no hará nada. Por ejemplo,

$$\begin{aligned} \text{In}[1] &:= \text{If}[2 > 1, 2 + 2] \\ \text{Out}[1] &= 4 \end{aligned}$$

pero si escribimos

$$\text{In}[2] := \text{If}[2 > 1, 2 + 2]$$

no se obtendrá ninguna salida. Otro tipo de expresión es la siguiente

$$\text{If}[\text{condición}, f_1, f_2].$$

Ahora, si la condición es cierta el programa ejecutará f_1 y si es falsa ejecutará f_2. Por ejemplo

$$\begin{aligned} \text{In}[3] &:= \text{If}[1 > 2, 2 + 1, 2 + 2] \\ \text{Out}[3] &= 4. \end{aligned}$$

2.4. Programando un método numérico sencillo

Vamos a diseñar un programa que aproxime numéricamente mediante el método de Euler la solución del problema de condiciones iniciales

$$\begin{cases} y' = ty, \\ y(t_0) = y_0, \end{cases}$$

introduciendo las condiciones iniciales t_0 e y_0, el instante final t_f en el queremos conocer el valor de la solución, y el número de pasos n con el que este valor se aproxima. Una posible programación es la siguiente

$\text{f}[t\_, y\_] := t * y;$
$n = 100; \text{Array}[y, 0, n]; y[0] = 1.; t0 = 0.; tf = 10.;$
$h = (tf - t0)/n;$
$\text{For}[i = 1, i < n + 1, i++, y[i] = y[i - 1] + h * \text{f}[t0 + (i - 1) * h, y[i - 1]]];$
$\text{Print}[\text{“El valor aproximado es ”}\ y[n]];$

obteniéndose la salida

$$\text{Out}[1] = \text{El valor aproximado es 1,64762}$$

2.5. Presentaciones gráficas

A menudo, aparte de los datos numéricos en sí, es interesante tener una representación gráfica de los mismos. En el caso de la aproximación numérica de soluciones de ecuaciones diferenciales, éstos son dados en forma de sucesión de números reales y_n, con n variando de 1 a un valor N, de manera que y_N es la aproximación a la solución para el tiempo t_f que hemos elegido. En el caso de sistemas de ecuaciones diferenciales, tendremos una sucesión de vectores de los cuales, cada coordenada se corresponde con la aproximación de la función solución en el instante de tiempo dado. En cualquiera de los dos casos, es conveniente tener una manera de presentar los datos obtenidos mediante la integración numérica de una manera gráfica.

En general, el comando ListPlot permite hacer tales presentaciones mediante la estructura, pero previamente, tenemos que tener una manera de introducir en Mathematica las sucesiones con las que deseamos trabajar.

Dada una sucesión de números reales y_n, ésta se puede introducir en Mathematica de dos maneras. La primera es hacerlo como una lista en una variable, que se haría con la sintaxis

$$\begin{aligned} In[1] &:= Y = \{y_1, y_2, ..., y_n\} \\ Out[1] &= \{y_1, y_2, ..., y_n\} \end{aligned}$$

No obstante, si tenemos muchos datos, digamos 1000, no es operativo introducirlos de esta manera. Para ello tenemos la sentencia Append, que tiene la sintaxis

$$\text{Append}[Y, y_{n+1}]$$

que agrega el dato y_{n+1} al final de la lista Y. Veamos cómo:

$$\begin{aligned} In[1] &:= Y = \{2, 5, 6\} \\ Out[1] &= \{2, 5, 6\} \\ In[2] &:= \text{Append}[Y, 8] \\ Out[2] &= \{2, 5, 6, 8\} \end{aligned}$$

Supongamos ahora que deseamos introducir los mil primeros términos de la sucesión dada por la recurrencia $y_{n+1} = y_n + 2$, donde $y_1 = 1$, es decir, $y_n = (1, 3, 5, 7, ...)$. Escribimos el siguiente programa

$$\begin{aligned} Y &= \{1\}; x0 = 1; \text{For}[i = 1, i < 1000, i++, x1 = x0 + 2; \\ Z &= \text{Append}[Y, x1]; x0 = x1; Y = Z] \end{aligned}$$

Una vez ejecutamos el programa, almacenaremos en Y el valor de la sucesión.

Una manera alternativa de introducir dicha sucesión es mediante los arrays, que en dimensión uno se escriben como

$$\text{Array}[Y, n].$$

Esta sentencia genera automáticamente un vector que tiene por coordenadas $Y[1]$,$Y[2]$,...,$Y[n]$, que en principio estará vacío de contenido. Para introducir la sucesión anterior escribiríamos el programa

$$\begin{aligned}&\text{Array}[Y, 1000]; x0 = 1; \text{For}[i = 1, i < 1001,\\ &i++, Y[i] = x0; x1 = x0 + 2; x0 = x1]\end{aligned}$$

Si deseamos que en el array anterior el índice empiece a contar de cero, debemos usar la forma

$$\text{Array}[Y, n, 0].$$

Diferentes alternativas para estas sentencias pueden consultarse en la ayuda del programa. Pasemos a ver cómo representar gráficamente la información con el comando ListPlot, que tiene cualquiera de las siguientes estructuras

$$\text{ListPlot}[Y]$$

ó

$$\text{ListPlot}[\text{Array}[Y, 1000]]$$

según hallamos introducido la sucesión de una u otra manera. Así, tanto el programa

$$\begin{aligned}&Y = \{1\}; x0 = 1; \text{For}[i = 1, i < 1000, i++, x1 = x0 + 2;\\ &Z = \text{Append}[Y, x1]; x0 = x1; Y = Z]; \text{ListPlot}[Y]\end{aligned}$$

como su alternativa

$$\begin{aligned}&\text{Array}[Y, 1000]; x0 = 1; \text{For}[i = 1, i < 1001, i++, Y[i] = x0;\\ &x1 = x0 + 2; x0 = x1]; \text{ListPlot}[\text{Array}[Y, 1000]]\end{aligned}$$

nos devuelven la gráfica dada en la figura 2.1.

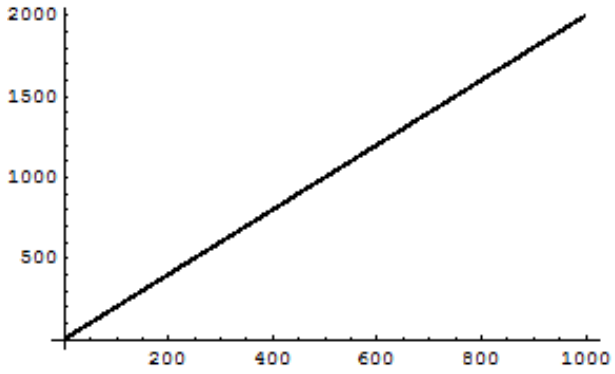


Figura 2.1: Representación gráfica proporcionada por la salida del comando ListPlot.

El comando ListPlot tiene la opción de unir dos puntos consecutivos con una recta. Tendría la sintaxis

$$\text{ListPlot}[Y, \text{PlotJoined} -> \text{True}]$$

ó

$$\text{ListPlot}[\text{Array}[Y, n], \text{PlotJoined} -> \text{True}]$$

Veamos como obtener una información gráfica de de aproximación que obtuvimos del problema de condiciones iniciales generado a partir de la ecuación $y' = ty$. Para ello, basta modificar el programa anterior según se indica

$\text{f}[t\_, y\_] := t * y;$
$n = 100; \text{Array}[y, 0, n]; y[0] = 1.; t0 = 0.; tf = 10.;$
$h = (tf - t0)/n;$
$\text{For}[i = 1, i < n + 1, i++, y[i] = y[i-1] + h * \text{f}[t0 + (i-1) * h, y[i-1]]];$
$\text{ListPlot}[\text{Array}[y, n, 0], \text{PlotJoined} -> \text{True}];$

obtenemos la figura 2.2 como aproximación de la solución con condición inicial $y(0) = 1$, donde el tiempo final es 10 y el número de pasos 1000.

Como puede apreciarse en la gráfica 2.2, en el eje x aparecen datos hasta 100, que se corresponden con el número de pasos. Si hubiésemos elegido 1000 pasos, aparecería obviamente 1000 al ser éstos en número de elementos en la

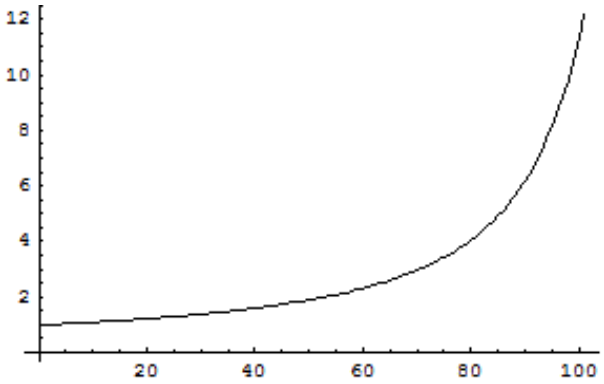


Figura 2.2: Aproximación de la solución del problema de condiciones inciales.

sucesión generada. Podemos hacer una representación gráfica donde el eje x recorra el intervalo de definición de la función solución de la siguiente manera

$$\begin{aligned}
&\mathrm{f}[t_, y_] := t * y;\\
&n = 100; \mathrm{Array}[y, 0, n]; y[0] = 1.; t0 = 0.; tf = 10.;\\
&h = (tf - t0)/n; r = \{\{t0, y[0]\}\};\\
&\mathrm{For}[i = 1, i < n + 1, i + +, y[i] = y[i - 1] + h * \mathrm{f}[t0 + (i - 1) * h, y[i - 1]];\\
&r1 = \mathrm{Append}[r, \{t0 + i * h, y[i]\}]; r = r1];\\
&\mathrm{ListPlot}[r, \mathrm{PlotJoined} - > \mathrm{True}];
\end{aligned}$$

es decir, definiendo una sucesión r de vectores del plano, y representando ésta como muestra la figura 2.3

A veces, es necesario tener que representar a la vez dos sucesiones. Por ejemplo, las soluciones aproximadas y exacta de una ecuación diferencial para conocer o estimar la bondad de la aproximación. Por ejemplo, la solución del problema

$$\begin{cases} y' = ty, \\ y(0) = 1, \end{cases}$$

es

$$y(t) = e^{t^2/2}.$$

Supongamos que queremos ver una representación gráfica conjunta de dicha función y su aproximación en el intervalo $[0, 1]$ mediante 100 pasos. Para

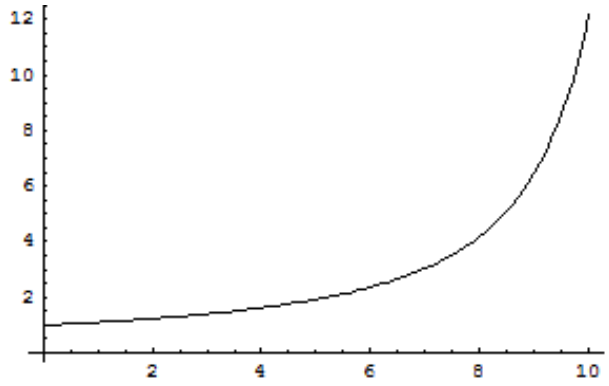


Figura 2.3: Representación de la solución en el dominio de definición.

ello, generamos los valores de la solución exacta en los 100 puntos donde obtenemos la aproximación numérica con el programa

$$g[t\_] := \text{Exp}[t\hat{}2/2]; t0 = 0.; tf = 1; n = 100; h = (tf - t0)/n;$$
$$graf = \{\{t0, g[t0]\}\}; For[i = 1, i < 101, i++, t1 = t0 + h; t0 = t1;$$
$$graf1 = \text{Append}[graf, \{t0, g[t0]\}], graf = graf1]$$

Igualmente, ejecutamos el programa anterior para obtener la solución aproximada almacenada en la variable r. Sin embargo, para representar ambas funciones a la vez hemos de activar un paquete especial de Mathematica tecleando

$$<< \text{Graphics`MultipleListPlot`}$$

A continuación utilizamos la sentencia MultipleListPlot, que permite representar varias sucesiones con la siguiente sintaxis

$$\text{MultipleListPlot}[\{r, graf\}, \text{PlotJoined} -> \text{True}]$$

que da lugar a la gráfica 2.4. Otras opciones para esta sentencia pueden verse en el manual del programa.

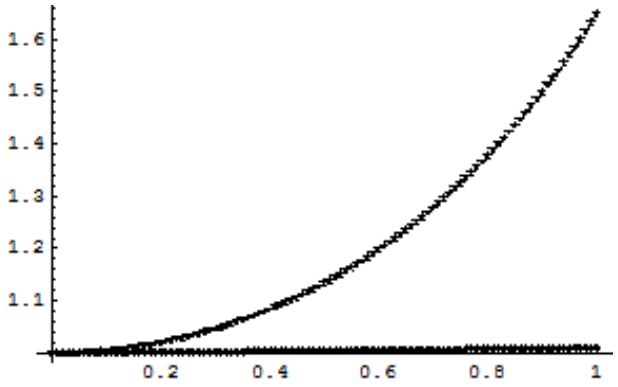


Figura 2.4: Representación conjunta de las soluciones aproximada y exacta del problema de condiciones iniciales.

Bibliografía

[1] M.L. Abell y J.P. Braselton, *Differential Equations with Mathematica,* Ed. AP Proffessional.

[2] Stephen Wolfram, *"The Mathematica Book",* Wolfram Media, Cambridge University Press.

[3] E. Castillo y otros, *Mathematica*, Ed. Paraninfo.