INTRODUCCIÓN A MATHEMATICA.

1.Cuestiones básicas

- Todo en inglés.
- Al arrancar nos sale una ventana que se llama notebook donde ya podemos trabajar.
- Mathematica trae incorporadas muchísimas funciones para poder realizar prácticamente cualquier cálculo de tipo matemático: integrar, derivar, resolver ecuaciones y sistemas, etc. Bastará llamar a la función correspondiente de forma adecuada. Pero también trae incorporado su propio lenguaje de programación (que recuerda bastante al lenguaje de programación C con el que realmente está escrito el propio Mathematica), con el que podremos construir programas de acuerdo a nuestras necesidades.
- Mathematica numera de forma automática la entradas (inputs) y las salidas (outputs).
- Las sentencias se agrupan en celdillas.
- Cualquier expresión o cálculo que queramos realizar debe escribirse en una celdilla de entrada. Mathematica colocará los resultados en una celdilla de salida. Las celdillas de entrada están en negrita y se pueden editar, mientras que las de salida (escritas por el propio programa) no se pueden editar.
- Para ejecutar una orden pulsar simultáneamente: mayúsculas+intro.
- Mathematica distingue entre mayúsculas y minúsculas, así que mucho cuidado
- Las operaciones aritméticas son: +, -,*, /, ^.
- No usar corchetes para las operaciones, ya que éstos están reservados para las funciones. De modo que, una expresión como:

  2*[3+5]

  no es correcta, ya que debemos usar paréntesis: 2*(3+5).
- Tampoco se pueden utilizar las llaves para operaciones aritméticas, ya que éstas sólo se usan para trabajar con vectores y matrices.
- Todas las funciones o comandos de Mathematica comienzan con letras mayúsculas y el argumento (o argumentos de la función) van encerrados entre corchetes. Por ejemplo, la función Print[] sirve para mostrar en pantalla un mensaje. Por ejemplo: Print[“Hola”] sacará en pantalla el texto encerrado entre comillas. Mientras que Print[*x*] mostrará en pantalla el valor que tenga en ese momento la variable *x*.
- Mathematica trae algunas constantes incorporadas, también comenzando por mayúsculas, como: Pi (para el número pi), E (para el número *e*), I (para la unidad imaginaria), etc.
- Funciones matemáticas incorporadas:
  - Abs[ ] para calcular el valor absoluto.
  - Sqrt[ ] para el cálculo de la raíz cuadrada.

- Sin[ ], Cos[ ], etc. para las funciones trigonométricas.
- Las funciones exponencial Exp[ ] y logaritmo neperiano Log[ ].

- Es un lenguaje interpretado, ya que se lee una línea y a continuación se ejecuta. Cada vez que ejecuta una línea nos va sacando en pantalla información sobre lo que está haciendo. Esto puede hacer que en un programa largo nos proporcione información no deseada. Si queremos evitar esto (se suele conocer por eco), basta con terminar cada línea de entrada por un punto y coma.

2. Sobre números y precisión

Mathematica permite trabajar de forma exacta así:

In: 3/4+2/3+1/2

Out:23/12

Si deseamos ver el valor aproximado de la expresión anterior, bastará poner:

N[3/4+2/3+1/2]

donde N viene de "**N**umerical Aproximation" y veremos en pantalla:

1.91667

y se muestra en pantalla con 5 cifras decimales por defecto, aunque en realidad tendrá muchas más.

Sin embargo, si ponemos:

3/4+2/3+1/2. o si ponemos 3/4+2/3+0.5 (insertando un punto decimal en algún número de los que intervienen en la cuenta), Mathematica hará los cálculos con números aproximados y el resultado será directamente un número aproximado (con decimales).

Los números irracionales los dejará tal cual mientras no le digamos lo contrario, por que así conservará el valor exacto de dichos números:

In: Sqrt[2]

Out: $\sqrt{2}$

In: N[Sqrt[2]]

Out: 1.41421

La expresión N[Sqrt[2]] es equivalente a Sqrt[2.] y ambas proporcionan la misma salida: 1.41421

3. Sobre variables y funciones con Mathematica.

Existen distintos tipos de variables:

a =7*3+5

a=Log[2]

a="esto es una prueba"

y funciones que se definen de la forma *nombre de la función*[*variable*_] , por ejemplo

$f[x\_] = 2*x+1$, $g[x\_] = 3*x\text{^}2+2*x-5$, $h[x\_] = \text{Cos}[2*\text{Pi}*x]$, z[x_,y_]=x*x+2*y, etc.

Una vez definida la función puede evaluarse en cualquier punto, por ejemplo, si ejecutamos la orden *f*[2], nos devuelve 5.

Las variables y las funciones permanecen en memoria durante toda la sesión de trabajo. Si queremos reutilizar una variable o función de forma completamente nueva en la misma sesión y no queremos que se puedan producir conflictos con el uso previo que se han hecho de esas funciones o variables basta con ejecutar:

Clear[nombre de la variable ó de la función]

4. Sobre vectores y matrices con Mathematica.

Una lista ó vector es un conjunto de datos dispuestos en una estructura de la forma: {dato1, dato2,…} .

Ej: vect={-3, 4, 5}

es la forma de trabajar con un vector llamado vect con 3 componentes.

El elemento i-ésimo de un vector se extrae de la siguiente forma: vect[[i]]. Así concretamente: vect[[2]]=4.

Las matrices tienen un tratamiento parecido. Así la matriz $A = \begin{pmatrix} -2 & 2 & 1 \\ 5 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ se introduce en Mathematica de la siguiente forma:

A={{-2, 2, 1}, {5, 2, 0}, {0, 0, 1}}.

Para referirnos a un elemento concreto de la matriz utilizaremos: A[[i,j]]. Así, en el ejemplo utilizado A[[3,1]]=0.

Por defecto, el programa nos muestra las matrices justo en el formato anterior. Tenemos el comando **MatrixForm**[A] para que nos muestre la matriz A en su formato más habitual en Matemáticas.

Se puede trabajar con vectores y matrices de igual forma que en Álgebra Lineal: suma, resta, producto de un escalar por un vector. Por ejemplo: A+B sumará dos matrices, A-B hará la diferencia, 5*A para multiplicar un escalar por una matriz, A.B para multiplicar matrices, etc.

El comando Det[A] nos calcula el determinante de una matriz.

El comando Inverse[A] calcula la inversa de una matriz.

El comando Transpose[A] calcula la matriz transpuesta.

El comando LinearSolve[A,B] resuelve un sistema lineal donde A es la matriz de los coeficientes del sistema y B es la matriz de los términos independientes.

El comando Dimensions**[**A**]** nos devuelve las dimensiones de la matriz A.

Un comando útil para crear matrices es Table:

Table[*expresión*, {*contador*, *inicio*, *fin*, *paso*}]

genera una lista con tantas componentes como valores tome el *contador* desde *inicio* hasta *fin* con el *paso* indicado. Así, por ejemplo:

b=Table[0,{i, 1, 5, 1}], crea un vector con cinco componentes todas nulas.

v=Table[3*i, {i, 1, 5, 2}] asigna a v la lista {3, 9, 15}.

Por otro lado:

Table[*expresión*, {*contador1*, *inicio1*, *fin1*, *paso1*},{*contador2*, *inicio2*, *fin2*, *paso2*}]

genera una matriz con tantas filas como indique *contador1* y tantas columnas como indique *contador2*.

Ej: A=Table[0,{i,1,3,1},{j,1,4,1}] genera una matriz nula de dimensiones 3×4.

5. Expresiones lógicas

Una expresión lógica es una expresión que sólo puede tomar los valores verdadero (True) y falso (False). Los operadores lógicos son:

= = (igual), ! = (distinto), > (mayor), < (menor), >= (mayor o igual), <= (menor o igual)

Ej: a=5
a!=0
TRUE
a>=3
TRUE
a==0
FALSE

Una expresión lógica puede estar compuesta a su vez de variables expresiones lógicas más simples, unidas por conectivos lógicos. Los conectivos lógicos más habituales son: p | | q para "p o q" (falsa sólo en el caso de que p y q sean falsas) y p&&q para "p y q" (verdadera sólo en el caso de que p y q sean verdaderas).

Ej:
a=3; b=4;
(a==3)||(b>5)
TRUE
(a==3)&&(b>5)
FALSE

6. Órdenes condicionales

La orden condicional más utilizada es If . La versión más simple es:

If [*condición lógica*, *proceso*]

Su forma de actuar es como sigue: si la *condición lógica* es verdadera, se ejecuta el comando contenido en *proceso*. En *proceso* se pueden escribir más de un comando o acción pero tendremos que separarlos por punto y coma.

Ej: ¿Qué hará el siguiente programa?

a=3;

If[a<5,Print["Hola"]]

7. Bucles

Un bucle es un proceso que se repite un cierto número de veces. Se pueden crear bucles con distintos comandos: **For, Do y While**.

En los bucles se usan *contadores*. Una expresión como $i = i + 1$ no tiene sentido en Matemáticas, pero esta sentencia en cualquier lenguaje de programación significa lo siguiente: hacer *i* igual al valor que tenga *i* en este momento más uno.

El formato general de un For es el siguiente:

For[*contador=inicio, condición-lógica, incremento-contador, proceso-a realizar*]

Su misión es realizar las órdenes que aparecen en *proceso-a-realizar* (si hubiera más de uno se separarán con ;) tantas veces como valores tome contador desde *inicio* y mientras la *condición-lógica* sea verdadera, variando el valor del contador según la cantidad que se ponga en *incremento-contador.*

Ej: Vamos a "castigar" al ordenador haciendo que escriba 100 veces el mensaje: "esto es una prueba"].
For[i=1 , i<=100, i=i+1, Print["esto es una prueba"]];

Nota: i++ equivale a i=i+1
i— equivale a i=i-1

Ej: For[k=40,k>=35,k--,Print[Log[k]]]. ¿Qué saldrá en pantalla?.

Ej: For[k=40,k<=35,k--, Print["El logaritmo neperiano de ",k," es ",N[Log[k]]]]. ¿Qué saldrá en pantalla?.

Ej: ¿Qué ocurre si ejecutamos el siguiente bucle?:
For[i=1,i>=1,i++,Print["Hola"]]

Ej. Hacer un programa que calcule la suma de los *n* primeros números naturales

Ej: Hacer un programa que calcule el factorial de un número.

Ej: Hacer un programa que calcule la suma de los cuadrados de los múltiplos de 3 comprendidos entre 1 y 1000.

Ej: Supongamos que tenemos una sucesión recurrente del siguiente tipo:
$a_n = a_{n-1} + 2a_{n-2}$ siendo $a_1 = 2,\ a_2 = 1$. Calcule el término que ocupa el lugar 100.

- Otra orden con la que también se pueden hacer bucles es con **While**. El formato de esta orden es de la siguiente forma:
  While [*condición-lógica, proceso-a-ejecutar*]
  que ejecuta las órdenes contenidas en *proceso-a-ejecutar*, mientras la *condición-lógica* sea cierta.

  Ej: ¿Qué hará el siguiente programa?:
  i=1;
  While[i<=10, Print[i^2]; i=i+1]

- Otra orden para construir bucles es **Do**. El formato es:
  Do[*proceso-a-ejecutar*, {*contador*, *inicio, fin, incremento*}

  Por ejemplo, otra forma de hacer lo mismo que en el ejemplo anterior pero utilizando la orden Do:
  Do [Print[i^2],{ i, 1, 10, 1}]

NOTA : Cuando deseemos salir de un bucle se utilizará la orden **Break[].** El proceso continuará con la orden que vaya a continuación del bucle.

Ej: Deseamos escribir un programa que escriba en pantalla los cuadrados de los números naturales comprendidos entre 30 y 40, siempre que el cuadrado no supere el valor 1500.

```
For[ i=30, i<=40, i++,
cuad=i^2;
If[cuad>1500, Break[]];
Print[cuad];
  ]
```

8. Gráficos con Mathematica.

Una de las características más interesantes del programa Mathematica es su capacidad para la realización de distintos tipos de gráficos. Estos pueden ser bidimensionales y tridimensionales, en color, incluso se puede incluir animación, simulando movimiento. Precisamente, este es uno de los aspectos ha favorecido más la difusión de este programa.

Para representar la gráfica de funciones de una sola variable, por ejemplo de la forma $y = f(x)$ cuando *x* valía entre valores determinados *xmin, xmax,* se utiliza la orden **Plot**[f[x],{x,xmin,xmax}]
donde *f(x)* puede estar definida con anterioridad, o en caso contrario bastará con sustituir aquí la correspondiente expresión de la función que se debe dibujar. *Mathematica* realiza de forma automática la elección de una escala adecuada en ambos ejes para que la gráfica se pueda ver lo mejor posible. Por ejemplo, si queremos dibujar la función $f(x) = \text{sen}(2x)$ en el intervalo $[0, \pi]$, podemos ejecutar:

```
Plot[Sin[2*x], {x,0, Pi}]
```

ó bien:

```
f[x_]=Sin[2*x];
Plot[f[x],{x,0, Pi}]
```