INTRODUCCIÓN A MAPLE V

INTRODUCCIÓN

Una vez abierta una hoja de trabajo (Worksheet) se puede ver que ésta se divide en cuatro tipos de regiones:
-Regiones de texto (en negro) con comentarios explicativos sobre los cálculos. El presente párrafo es un ejemplo de región de texto.
-Regiones de entrada o "input" (en rojo) , con las órdenes de cálculo que se pide al ordenador que ejecute.
-Regiones de salida o "output" (en azul), con los resultados de los cálculos pedidos, y que son devueltos junto con la correspondiente región de entrada.
-Regiones de gráfico, con los gráficos que devuelve Maple.

También es posible dividir una hoja de trabajo en secciones y subsecciones:

Esto es la sección A

Podemos definir una subsección de A

Esto es una subsección de la sección A

Aquí acaba la subsección

Para ordenar a Maple que ejecute un cálculo se escribe en la región de entrada, a la derecha del símbolo " >", lo que se desea calcular seguido del terminador punto y coma ";". La ejecución del cálculo se produce pulsando "Intro".

Operaciones matemáticas elementales. Aritmética.

Algunos ejemplos de cálculos elementales con Maple son los siguientes:

```
> cos(0);
> sin(0);
> 4+3/5;
> 4*3+2^3;
> 74!;
```

Para saber si un entero es primo se le aplica el comando "isprime"

```
> isprime(237);
> nextprime(2000);
> prevprime(2000);
```

Para obtener el n-ésimo número primo se escribe

```
> ithprime(5);
```

Si en vez del terminador punto y coma ponemos dos puntos ":", Maple ejecutará los cálculos pero no
los expresará en pantalla. Por ejemplo, para asignar a la variable x el valor 3, escribimos

```
> u:=3:
```
Efectuemos el siguiente cálculo:

```
> u-5;
```
Como se ve, aunque no lo mostró en pantalla, Maple asignó el valor 3 a la variable u para después efectuar el cálculo pedido.
Si lo que se desea es borrar la asignación hecha a una variable, basta reasignarle su propio nombre entrecomillado
```
> u := 'u';
```

Otros cálculo numérico sencillo es la descomposición en factores primos de un número dado. Así,
```
> 20!;
> ifactor(%);
```
El comando que ejecuta esta operación es "ifactor" (integer factorization). El símbolo % representa a la última salida o output (aunque no halla aparecido en pantalla), de modo que lo que hemos hecho es factorizar el número 2432902008176640000 .

El comando "expand" desarrolla una expresión. Así,

```
> expand(%);
> expand(23*(2^3)+27+2^5);
```

La expresión $\frac{2^{30}\sqrt{3}}{3^{20}}$ la escribimos en Maple del siguiente modo
```
> 2^30*sqrt(3)/3^(20);
```
La expresión en coma flotante de este número es
```
> evalf(%);
```
Si queremos que el número de dígitos sea distinto de 10, por ejemplo 15, basta escribir
```
> evalf(%%,15);
```
Los símbolos %% y %%% indican que estamos operando sobre la penúltima y antepenúltima salidas respectivamente.

Para modificar el número de dígitos con que se trabaja en coma flotante basta dar el valor deseado a la variable "Digits".

```
> Digits:=14:
```
Entonces se tendrán las expresiones con 14 dígitos.

```
> evalf(sqrt(3));
```
Para restablecer el número de decimales a 10 escribimos
```
> Digits:=10:
> evalf(sqrt(3));
```
Para convertir una expresión racional en decimal y viceversa, se utiliza el comando "convert".

```
> q:=50/7;
> evalf(q);
> r:=convert(q,float);
> convert(r,rational);
```

Otro comando utilizado en la manipulación de números es el valor absoluto

```
> abs(-4.6);
```

La parte entera, la parte fraccionaria y el redondeo de una expresión numérica se obtienen con los comandos

```
> trunc(2.57);
> frac(2.57);
> round(2.57);
```

El máximo y el mínimo de una familia de números se obtiene aplicando los comandos

```
> max(2,3.2,7/8,5);
> min(2,3.2,7/8,5);
```

El máximo común divisor (integer great common divisor) y el mínimo común múltiplo (integer least common multiple) de dos o más enteros se calculan con los comandos

```
> igcd(60,36,20);
> ilcm(60,36,20);
```

En una división el cociente y el resto se obtienen aplicando a dividendo y divisor los comandos

```
> iquo(23,5);
> irem(23,5);
```

Paquetes o bibliotecas.

El sistema Maple dispone de cerca de 3000 comandos. Mantenerlos disponibles simultáneamente requeriría una cantidad de memoria enorme. Por ello, al iniciar Maple, sólo se carga el denominado núcleo (kernel), que contiene los comandos y funciones de más uso. El resto se guarda en paquetes y bibliotecas que se van cargando según sean necesarios.
Para obtener el listado de paquetes de Maple, así como información sobre los mismos, basta escribir

```
> ?index,packages
```

y pulsar la tecla "Intro".
Si lo que se desea es cargar un determinado paquete, por ejemplo "plots", que sirve para dibujar gráficas, escribiremos

```
> with(plots);
```

Si no se desea que aparezcan escritos en pantalla los nombres de los comandos cargados basta poner dos puntos en vez de punto y coma al final del input. Una vez cargados, los comandos del paquete "plots" pueden ya ser utilizados. Por ejemplo, "animate3d", representa la gráfica de una superficie determinada por una función f(x,y), con animación respecto de un parámetro t.

```
> animate3d( cos(t*x)*sin(t*y), x=-Pi..Pi, y=-Pi..Pi, t=1..2
  );
> with(plots,animate3d):
```

Si queremos utilizar el comando "animate3d" sin llegar a cargarlo, ordenaremos

```
> plots[animate3d](cos(t*x)*sin(t*y),x=-Pi..Pi,y=-Pi..Pi,
  t=1..2);
```

De igual modo se puede utilizar el comando "divisors" del paquete "numtheory" para calcular los divisores de un entero dado

```
> numtheory[divisors](36);
```

Para eliminar los comandos de los paquetes que se hayan cargado hasta ahora (así como las asignaciones a variables o definiciones hechas por el usuario) se ordena

```
> restart;
```

Una observación importante es que si en una hoja de trabajo se carga un comando o se hace una asignación a una variable y, en la misma sesión, se abren otras hojas de trabajo, dichas operaciones quedan hechas para todas las hojas (salvo si la sesión se abre en modo Parallel Server, en cuyo caso cada hoja de trabajo mantiene sus propias asignaciones).

La ayuda en Maple.

Para pedir ayuda sobre un comando, por ejemplo "int" (integración), ordenamos:

```
> ?int
```

y pulsamos la tecla "Intro". Aparece entonces una descripción de dicho comando y ejemplos de su uso. Si el comando sobre el que buscamos información está contenido en un paquete (por ejemplo el comando "midpoint" del paquete "student"), se escribe

```
> ?student,midpoint
>
>
```

y se pulsa "Intro". Además en el menú Help de la barra de menús se encuentran dos opciones:
-"Topic search", que permite buscar ayuda sobre un comando Maple.
-"Full text search", que determina en qué lugares aparece una cadena de caracteres.

EXPRESIONES POLINÓMICAS Y FRACCIONES ALGEBRAICAS. EXPRESIONES CON POTENCIAS Y RAÍCES. VARIABLES.

Manipulación de expresiones polinómicas y fracciones algebraicas.

Definamos un polinomio q (multivariable en este caso),

```
> q:= 5*x^2*y^3+x*y+7*y-2;
```

Para determinar el número de incógnitas de q, su grado, o el grado de q respecto de la variable x, escribimos

```
> indets(q); degree(q); degree(q,x);
```

El coeficiente de q respecto de la variable y elevada a 3, o los coeficientes de mayor y menor grado de q respecto de la variable x se escriben con los comandos

```
> coeff(q,y,3); lcoeff(q,x); tcoeff(q,x);
```

Para desarrollar una expresión simbólica (con variables) como, por ejemplo, un producto de dos polinomios, se utiliza el comando "expand"

```
> expand((5*x^3+2*x+1)*(4*x+15));
```

ya que, de otro modo, la operación sólo quedaría indicada (algo que no ocurre si trabajamos con números):

```
> (5*x^3+2*x+1)*(4*x+15);
```

La simplificación de expresiones fraccionarias simbólicas tampoco es automática, como ocurre con los números, y requiere del uso del comando "simplify",

```
> simplify((x^3+x^2-2*x-2)/(x^4-1));
```

El cociente y el resto de la división de dos polinomios *a* / *b* en una variable, por ejemplo

```
> a:=x^4+5*x^2+2*x-6; b:=x^3-3*x^2+1;
```

se obtiene mediante los comandos

```
> quo(a,b,x); rem(a,b,x);
```

Efectivamente, veamos que el dividendo es igual al cociente por el divisor, más el resto:

```
> a=expand((x+3)*b + (-9+14*x^2+x));
```

Dado un polinomio (de una o más variables), la factorización del mismo sobre el cuerpo determinado por sus coeficientes se lleva a cabo con el comando "factor"

```
> factor(x^3-4*x^2+x+6);
```

Así, tenemos la factorización de $p(x) = x^3 - 4\,x^2 + x + 6$ en el cuerpo de los números racionales *Q*, que nos proporciona las raíces racionales del polinomio. En este caso, las raíces racionales son todas las raíces.

Sin embargo, existen polinomios como $q(x) = x^5 - 3\,x^4 - x^3 + 3\,x^2 - 2\,x + 6$, cuya factorización sobre *Q* es

```
> q=factor(x^5-3*x^4-x^3+3*x^2-2*x+6);
```

y cuyas raíces racionales (el 3 en este caso) no son todas las raíces ya que existen dos raíces irracionales $\{\sqrt{2}\ , -\sqrt{2}\}$ y otras dos complejas $\{I, -I\}$. Si escribimos

```
> q=factor(x^5-3*x^4-x^3+3*x^2-2*x+6,real);
```

obtenemos la factorización de *q* sobre el cuerpo de los números reales *R* , con las raíces reales $\{\sqrt{2}, -\sqrt{2}, 3\}$ evaluadas en coma flotante. Si escribimos

```
> q=factor(x^5-3*x^4-x^3+3*x^2-2*x+6,complex);
```

obtenemos la descomposición de *q* en factores definidos sobre el cuerpo de los números complejos *C*, con todas las raíces $\{\sqrt{2}, -\sqrt{2}, 3, I, -I\}$ evaluadas en coma flotante.

Para decidir si un polinomio es múltiplo de otro se aplica el comando "divide"

```
> divide(x^3-1,x+1);
> divide(x^2-y^2,x+y);
```

Efectivamente, si factorizamos $x^3 - 1$ en *Q*, queda

```
> factor(x^3-1);
```

y si factorizamos $x^2 - y^2$ en Q, obtenemos

```
> factor(x^2-y^2);
```

Para decidir si un polinomio, $x^2 - 2$ por ejemplo, es irreducible sobre los cuerpos Q, R ó C se escribe

```
> irreduc(x^2-2), irreduc(x^2-2,real),
  irreduc(x^2-2,complex);
```

Efectivamente, si factorizamos $x^2 - 2$ en Q, R y C, tenemos

```
> factor(x^2-2); factor(x^2-2,real); factor(x^2-2,complex);
```

Obsérvese que, al venir expresada en coma flotante, la factorización en R ó C de un polinomio que tenga raíces irracionales (es el caso de $x^2 - 2$) será tan solo una aproximación (tan precisa como sea necesario).

El máximo común divisor y el mínimo común múltiplo de polinomios con coeficientes racionales se obtiene aplicando los comandos "gcd" (greatest common divisor) y "lcm" (least common multiple).

```
> gcd(x^3-1,x^2-2*x+1); lcm(x^3-1,x^2-2*x+1);
```

como se comprueba si factorizamos

```
> factor(x^3-1); factor(x^2-2*x+1);
```

Si los polinomios tienen varias variables, se trabaja de la misma manera.

Para ordenar respecto del grado los términos de un polinomio univariable (expandido) se utiliza el comando "sort"

```
> sort(5+7*x^2+3*x-5*x^4+23*x^3);
```

Los polinomios multivariables (expandidos) se pueden ordenar de distintos modos. Destacamos los siguientes:

-Ordenación respecto del grado de una variable (por ejemplo la y) elegida por nosotros,

```
> sort(x^2-2*x*y^3-2*x*z^2+y^6+2*y^3*z^2+z^4,y);
```

-Ordenación respecto del grado total (los términos del mismo grado siguen el orden lexicográfico usual [x,y,z]),

```
> sort(x^2-2*x*y^3-2*x*z^2+y^6+2*y^3*z^2+z^4,tdeg);
```

-Ordenación respecto del grado total (los términos del mismo grado siguen el orden lexicográfico impuesto por nosotros)

```
> sort(x^2-2*x*y^3-2*x*z^2+y^6+2*y^3*z^2+z^4,[z,y,x],tdeg);
```

-Ordenación lexicográfica que nosotros impongamos (por ejemplo [z,y,x]) sin tener en cuenta el grado,

```
> sort(x^2-2*x*y^3-2*x*z^2+y^6+2*y^3*z^2+z^4,[z,y,x],plex);
```

Consideremos un polinomio (no necesariamente expandido),

```
> h:=(x+y)^4+(x-y)^3+(x^2+1)*y;
```

-Si lo que deseamos es agrupar los términos con respecto a la variable y, se aplica el comando "collect" al polinomio y a la variable

```
> collect(h,y);
```

Expresiones con potencias y raíces.

A continuación escribimos "restart" para borrar las asignaciones (:=) hechas anteriormente y "reiniciar" el trabajo.

```
> restart:
```

Para desarrollar la expresión potencial $a^{(b+c)}$ se escribe

```
> expand(a^(b+c));
```

Si lo que queremos es simplificar expresiones del tipo $a^b\, a^c$, $\frac{a^b}{a^c}$, se escribe

```
> simplify(a^b*a^c); simplify(a^b/a^c);
```

Las raíces cuadradas pueden introducirse con el comando "sqrt" o elevando a 1/2:

```
> sqrt(2)=2^(1/2);
```

También podemos evaluar en coma flotante una expresión radical

```
> evalf(sqrt(2),16);
```

Para simplificar expresiones numéricas radicales como $(\sqrt{2}+1)^2-3-2\sqrt{2}$ se utiliza el comando "simplify"

```
> simplify((sqrt(2)+1)^2-3-2*sqrt(2));
```

Si queremos simplificar expresiones radicales anidadas como $\sqrt{(\sqrt{x}+1)\,(\sqrt{x}+1)}$, utilizamos el comando "radsimp"

```
> radsimp(sqrt((sqrt(x)+1)*(sqrt(x)+1)));
```

Para racionalizar denominadores de expresiones como $\frac{1}{\sqrt{2}+1}$ se escribe

```
> rationalize(1/(sqrt(2)+1));
```

Variables. Sustitución y restricciones.

Dada una expresión en forma simbólica, por ejemplo un polinomio, las sustituciones en ella se realizan con el comando "subs":

```
> q:=x+y+4*z+3;
> subs(x=1,y=2,z=5,q);
```

La sustitución también se puede realizar en un conjunto de expresiones:

```
> subs(x=a,y=b,{x^2+2*y,x+2*y});
```

Si lo que se desea es una sustitución que no sea meramente sintáctica debemos recurrir al comando "algsubs"

```
> algsubs(y+z=a^2,x+y+2*z);
```

Para establecer restricciones sobre una variable, por ejemplo $t < 5$, basta escribir

```
> assume(t<5);
```

con lo que, en lo sucesivo, y mientras no se ordene lo contrario, el sistema considerará a t menor que 5. Podemos preguntarnos, por ejemplo, si la suma $t + 1$ es menor que 6 o menor que 5

```
> is(t+1<6);
> is(t+1<5);
```

En adelante, al operar con una variable sobre la que hay alguna restricción (como en el caso presente) obtendremos una tilde junto a ella:

```
> x+2+t*y;
```

Si deseamos aplicar varias restricciones a la vez, por ejemplo t perteneciente al intervalo

[0,5), escribimos

```
> assume(t<5);
> additionally(t>=0);
```

Para conocer las restricciones efectuadas sobre *t* aplicamos

```
> about(t);
```

Para borrar las restricciones basta con escribir

```
> t:='t':
> about(t);
```

La restricción de una variable a un campo numérico, por ejemplo los enteros, se escribe

```
> assume(n,integer);
> cos(n*Pi);
> about(n);
> n:='n':
> about(n);
```

TIPOS DE OBJETOS: SUCESIONES, CONJUNTOS, LISTAS, ARRAYS Y FUNCIONES

Sucesiones.

El comando "seq" (sequence o sucesión) permite definir sucesiones finitas cuyos elementos siguen una ley lógica de formación. Algunos ejemplos son los siguientes

```
> seq(n^2,n=1..5);
> seq((-1)^n,n=0..10);
```

Si deseamos escribir subíndices en los elementos de una sucesión, se pondrán entre corchetes

```
> seq(a[i]*x^i, i=1..8);
```

Los primeros 12 términos de una progresión aritmética de primer término 3 y razón 2 serán

```
> seq(3+2*n,n=0..14);
```

Los términos cuarto al noveno de una progresión geométrica de primer término 3 y razón 2 son

```
> seq(3*2^n,n=3..8);
```

Para calcular sumas y productos de los elementos de una sucesión como $\sum_{n=1}^{7} 2^{(-n)}$, $\prod_{n=1}^{7} 2^{(-n)}$ se utilizan los comandos "sum" y "product" (escribimos "Sum" y "Product" para que aparezcan los símbolos matemáticos del sumatorio y del producto)

```
> Sum(2^(-n), n=1..7)=sum(2^(-n), n=1..7);
> Product(2^(-n),n=1..7)=product(2^(-n), n=1..7);
```

Conjuntos.

Los conjuntos finitos pueden definirse enumerando sus elementos o, si siguen una ley lógica de formación, utilizando el signo "$", como veremos ahora:

```
> A:={3,6,9,12,15,18,21,24};
> B:={3*n $ n=1..8};
> C:={3*(-1)^n $ n=0..10};
```

Para que aparezca el conjunto denotado por A, para conocer el número de elementos del

conjunto A y para saber si es cierto que $A = B$ se escribe, respectivamente,

```
> A;
> nops(A);
> evalb(A=B);
```

Las uniones, intersecciones y restas entre conjuntos se realizan como sigue

```
> A union C;
> A intersect C;
> A minus B;
```

Listas.

Las listas se definen introduciendo los elementos en el orden deseado, colocándolos entre corchetes

```
> L:=[a,b,c,c,b,e,f];
```

Al contrario de lo que ocurre con los conjuntos, aquí se respeta el orden y la repetición de los elementos.

Para obtener la relación de elementos de L , el número de elementos de L, el elemento tercero o los elementos entre las posiciones segunda y quinta de L, se utilizan , respectivamente, los comandos

```
> op(L);
```

```
> nops(L);
> op(3,L);
> op(2..5,L);
```

Dadas dos listas

```
> L2:=[a,b,c,d,d,a]; L3:=[x,y,z,t];
```

éstas pueden concatenarse del siguiente modo

```
> N:=[op(L2),op(L3)];
```

También se pueden construir listas con algunos de sus elementos también listas

```
> Q:=[op(L3),N,op(L2)];
> nops(Q);
```

Arrays.

Los arrays son cajas en las que se introducen objetos de manera ordenada, bien en una única fila (arrays unidimensionales), o en filas y columnas (arrays bidimensionales), o en filas, columnas y pisos,...(arrays multidimensionales). Si un array A tiene dimensión 1 y cuatro elementos, escribiremos

```
> A:=array([2, luis, x^2, 0]);
```

Los elementos del array son 2, *luis*, x^2, 0 y se recuperan escribiendo

```
> A[1]; A[2]; A[3]; A[4];
```

Para recuperar toda la información sobre A ponemos

```
> print(A);
```

Si el array B es de dimensión 2 , con dos filas y tres columnas, se escribe

```
> B:=array([[2, x^2, luis], [j, 23, a]]);
```

Los elementos del array se recuperan escribiendo

```
> B[1,1], B[1,2], B[2,2],B[2,3];
```

Si definimos

```
> R:=array(1..2,1..3,[]);
```

Entonces Maple denota a los elementos de R por

```
> print(R);
```

Podemos definir las entradas de R del siguiente modo:

```
> R[1,2]:=2;R[2,2]:=perro; print(R);
```

Funciones.

La función f que asocia a x el valor $6\,x^3 + 5$ se denota en Maple

```
> f:=x->6*x^3+5;
```

Para calcular la imagen de 7 escribimos

```
> f(7);
```

Si la función tiene varias variables la definición es

```
> g:=(x,y,z)->5*x+3*y^2+z-1;
> g(1,2,0);
> g(a,b,2);
```

Para aplicar una función de una variable a estructuras como listas, conjuntos, arrays, se utiliza el comando "map". Consideremos la lista

```
> S:=[1,3,1,2,4];
```

Para aplicar la función f a la lista S se hace

```
> map(f,S);
```

para obtener otra lista. Lo mismo se hace si en vez de lista tenemos un conjunto o un array unidimensional.

Para aplicar una función de dos variables, g2, a los elementos de dos listas dadas C y E, se utiliza el comando "zip".

```
> g2:=(x,y)->x*y+2;
> C:=[a1,a2,a3];E:=[b1,b2,b3,b4];
> zip(g2,C,E);
```

Las operaciones con funciones suma, resta, producto, cociente y composición se ejecutan con los símbolos +, -, *, /, @ respectivamente. Así, la composición $f@g$ se escribe

```
> (f@g)(x,y,z);
> h:=x->x^2+1;
```

entonces las funciones $h+f$, $h-f$, $h\,f$, $\frac{f}{h}$ se expresan del siguiente modo

```
> (h+f)(x); (h-f)(x); (h*f)(x); (f/h)(x);
```

Para componer una función consigo misma n veces, por ejemplo f al cubo, se escribe

```
> f@@3;
> (f@@3)(x);
> expand(%);
```