1.2 Uso del software como una calculadora. Iniciación a los operadores y almacenamiento de datos.

- **Iniciación al manejo de datos**

Empezaremos a usar el software en modo calculadora, utilizando la ventana de comandos para escribir la instrucción y observando el resultado.

```
>> 2+3

ans =

     5
```

Todo resultado de una operación se debe almacenar en la memoria del computador. Si no se indica explícitamente el nombre de la variable donde se guardará éste, el lenguaje M usa la variable reservada *ans*.

Podemos por tanto interpretar que el resultado de la operación *2+3* se ha almacenado en la variable *ans*.

Cuando el programador use sus propias variables debe utilizar nombres válidos. Como identificador (nombre) se puede utilizar cualquier palabra que empiece por una letra y tenga a continuación cualquier combinación de letras, números o el carácter guión bajo. Es importante conocer que se distinguen mayúsculas de minúsculas, luego, por ejemplo, las variables `A` y `a` son distintas.

Otra cosa importante es saber cómo introducir los valores numéricos en las sentencias de este lenguaje. Los valores numéricos enteros se escriben literalmente, y los valores reales, usando un punto para separar la parte entera de los decimales o también utilizando notación científica. La notación científica es una forma de representar un número utilizando potencias de 10. Por ejemplo, el valor 34.2865 se podría escribir en notación científica como 0.342865×10^2 (se escribirá como 0.342865e2). Veamos un ejemplo en el que asignamos un valor entero a la variable `a`, mientras que a las variables `c` y `d` se les asignan valores reales.

```
>> a=23,  c=45.876,   d= 1.453e-2
```

M tiene variables predefinidas que contienen un valor por defecto. Por ejemplo, entre otras, existe la variable `pi` con el valor correspondiente a la constante matemática π.

Existen algunas variables que representan cantidades que no son consideradas números como tal. Estas cantidades son:

- Infinito, representado por la variable `Inf` (o `inf`)
- Cantidades indefinidas, representadas por la variable `NaN` (o `nan`)

La cantidad infinito se genera por un desborde en las operaciones o por una división entre cero. El valor `NaN`, que es una abreviación de la frase en inglés *'Not a Number'*, es obtenida como resultado de operaciones aritméticas indefinidas tales como $0/0$ ó $\infty - \infty$.

En los ejemplos anteriores aparecen ciertos operadores que conviene analizar con detalle.

- **Operador de asignación**

El operador de asignación (=) se utiliza para dar un valor nuevo a una variable. Su utilización es la siguiente:

```
variable = expresión
```

donde `expresión` es cualquier expresión válida en lenguaje M.

Si la variable no existe, ésta se creará con el valor y el tipo de dato de la expresión situada a la derecha del operador de asignación.

Si la variable existe, cada vez que se le asigne un nuevo valor, ésta pierde su estado anterior (tipo de dato y valor) y toma un nuevo estado correspondiente con el valor y el tipo de la expresión de la derecha.

- **Operadores aritméticos**

Los operadores aritméticos aplicables a escalares (ampliaremos éstos cuando se estudien matrices) son los siguientes:

Suma: +

Resta: -

Producto: *

División derecha: /

División izquierda: \

Potenciación: ^

Resto de una división: no existe un operador. A tal efecto se debe utilizar la función `rem(A,B)` que calcula el resto de dividir `A` entre `B`.

Para desarrollar una sentencia en la que aparezcan varios operadores, las operaciones se deben realizar siguiendo la prioridad de operadores. La siguiente tabla muestra la prioridad de operadores aritméticos y asignación de mayor a menor prioridad.

Operador	Símbolos	A igual prioridad se sigue el orden de
Paréntesis	( )	
Potencia	^	Izquierda a derecha
Multiplicación, división	* , / , \	Izquierda a derecha
Suma y resta, y cambio de signo (-)	+ , -	Izquierda a derecha
Asignación	=	Derecha a izquierda

Ejemplos:

```
>> 2.4*6

        ans=14.4000

>> 2^3

        ans=8

>> c=-1^4

        c=-1

>> c=(-1)^4

        c=1

>> 3/4

       ans= 0.7500

>> 3\4

         ans=1.3333

>> 2/3^2

        ans=0.2222

>> x=2/3*2

        x=1.3333

>> b=rem(6.7,5)

        b=1.7000
```

- **Formatos para el muestreo de resultados en pantalla**

Independientemente de la precisión con la que un dato se ha memorizado en la memoria RAM, podemos elegir entre diferentes formatos de escritura de los datos en pantalla. Existen varios modos de trabajo en MATLAB y Octave. Se indican a continuación, los más significativos:

`format short` punto fijo con cinco cifras significativas[1]. Si el dato no puede mostrarse adecuadamente con este formato opta por un formato en coma flotante.

`format long` punto fijo con quince cifras significativas. Si el dato no puede mostrarse adecuadamente con este formato opta por un formato en coma flotante.

`format short e` formato en coma flotante. Para representar la mantisa se usan cinco cifras significativas.

[1] Cantidad de cifras con las que se muestra un número, sin contabilizar los ceros a la izquierda en números reales y los ceros a la derecha en números enteros.

`format long e` formato en coma flotante. Para representar la mantisa se usan quince cifras significativas.

`format rat` aproximación por formato racional (cociente de enteros)

El modo de trabajo por defecto en lenguaje M es `format short` luego los datos se mostrarán con cinco cifras significativas mientras que el usuario no quiera modificar el formato.

A continuación se utiliza MATLAB, como si de una calculadora se tratara, para calcular los valores 3^{100} y $5+\pi$, cambiando el formato de muestreo de resultados. (Adelantamos aquí el uso del operador potenciación (^)).

```
>>3^100

             ans=5.1538e+047

>>5+pi

             ans=8.1416

>>format short e

>>3^100

             ans=5.1538e+047

>>5+pi

            ans=8.1416 e+000

>>format long

>>3^100

              ans=5.153775207320113e+047

>>5+pi

            ans=8.14159265358979

>>format long e

>>3^100

             ans=5.153775207320113e+047

>>5+pi

             ans=8.141592653589793e+000

>>format rat

>> 5+pi

             ans=920/113
```

Tecleando `format` se vuelve al formato por defecto, es decir, `format short`.

Usemos ahora la ventana de comandos para ver un ejemplo de los errores en el almacenamiento de datos. Escribamos en la ventana de comandos el número 2, y a continuación el resultado de la operación $(\sqrt{2})^2$ (Adelantamos aquí el uso de la función raíz cuadrada (sqrt)). Mostremos 15 cifras significativas.

```
>> 2

ans =

     2

>> format long

>> (sqrt(2))^2

ans =

   2.000000000000000
```

Matemáticamente ambos resultados son iguales y computacionalmente parecen iguales. Vamos a comprobarlo:

```
>> 2-(sqrt(2))^2

ans =

   -4.440892098500626e-016
```

El resultado debería ser cero, pero como el ordenador ha tenido que redondear $\sqrt{2}$ ya que tiene infinitos decimales, arrastra este error y la diferencia resulta distinta de cero. Se observa que hay diferencia a partir del decimal número 16, puede parecer un error pequeño, pero si no se tiene en cuenta, puede ocurrir que:

- si estamos comparando dos cantidades, la diferencia nunca puede ser cero y el programa fallará.

- si estamos realizando un cálculo de ingeniería, se puede producir una propagación y acumulación de errores que dé lugar a un resultado final con un error inaceptable.

Podemos averiguar cuál es la precisión relativa en punto flotante usando el comando `eps`.

```
>> eps
ans =  2.220446049250313e-16
```

Esto significa que si tenemos el valor `1` guardado, el siguiente valor que podemos almacenar es `1+eps`. Es debido a que en coma flotante y doble precisión se tienen 52 bits para la mantisa, siendo `eps` = 2^{-52}.