1. Introducción

GNU Octave es un lenguaje de alto nivel destinado para el cálculo numérico, sólo opera con números. Provee una interfaz sencilla, orientada a la línea de comandos (consola), que permite la resolución de problemas numéricos, lineales y no lineales, además permite la ejecución de scripts y puede ser usado como lenguaje orientado al procesamiento por lotes.

Octave nació alrededor del año 1988, y fue concebido originalmente para ser usado en un curso de diseño de reactores químicos. El desarrollo real de comenzó en 1992. La primera alfa fué publicada en 1993, y en 1997 se publicó la versión 1.0.

2. Iniciar y salir de Octave

Para iniciar octave hay que ejecutar la instrucción *octave* en una consola. Aparecerá la siguiente ventana:

```
GNU Octave, version 3.6.2
Copyright (C) 2012 John W. Eaton and others.
This is free software; see the source code for copying conditions.
There is ABSOLUTELY NO WARRANTY; not even for MERCHANTABILITY or
FITNESS FOR A PARTICULAR PURPOSE.  For details, type ‘warranty’.

Octave was configured for "x86_64-pc-linux-gnu".

Additional information about Octave is available at http://www.octave.org.

Please contribute if you find this software useful.
For more information, visit http://www.octave.org/help-wanted.html

Read http://www.octave.org/bugs.html to learn how to submit bug reports.

For information about changes from previous versions, type ‘news’.

octave:1>
```

Para salir de octave usaremos el comando *quit* o *exit*.

2.1. Instrucciones de utilidad

- >pwd: para mostrar el directorio en el que nos encontramos.
- >ls: para mostrar una lista de los ficheros y los directorios del directorio actual.
- >cd ruta: para cambiar de directorio.
- >>cd ..: para ir al directorio padre.
- >diary: volcar lo mostrado en el transcurso de la ejecución del programa al fichero ‘diary’ de la carpeta por defecto.
- >>diary fichero: el fichero ‘diary’ se guardará en el fichero indicado.

- >>diary off: para el guardado.
- >help comando: muestra la ayuda sobre el comando indicado.
- >history: muestra una lista con los comandos ejecutados.
- >save archivo: guardar la sesión en un archivo.
- >load archivo: cargar la sesión del archivo.

Los comandos pueden recuperarse con la *flecha hacia arriba*, o escribiendo el inicio del comando y pulsar la *flecha hacia arriba*, y navegar por los comandos ejecutados.

2.2. Operaciones básicas

- = *asignacion*
- + *suma*
- - *resta*
- * *multiplicación*
- .* *multiplicación de matrices elemento por elemento (deben coincidir el número de columnas con el número de filas)*
- / *división derecha*
- exp *exponencial*
- log *logaritmo neperiano*
- log10 *logaritmo base 10*
- sin *seno*
- cos *coseno*
- abs *valor absoluto*
- sqrt *raíz cuadrada*
- round *redondeo al entero más cercano*
- floor *redondea por defecto*
- ceil *redondea por exceso*
- ' *transpuesta*

3. Variables

Podemos asignar variables con determinados nombres a las expresiones numéricas (números, constantes). Los nombres son sensibles a mayúsulas y minúsculas, el máximo de caracteres que deben tener es de 31 caracteres, deben empezar or una letra y pueden contener letras, números y el símbolo '_'. Las variables se crean escribiendo el nombre que le queramos dar y asignándole el valor con el operador de asignación. El acceso a la variable se realiza escribiendo el nombre asignado.

```
>a=1/3
```

Cuando se asigna otro valor se machaca en anterior

```
>a=1/3
>a=1/6
>a
a=0.16667
```

Las variables creadas se guardan en una lista de variables.

- ¿whos: obtener la lista de vaiables guardadas.

Las variables guardadas pueden ser borradas.

- >clear: borra todas las varianles
- >>clear a: borra la variable indiaca (en este caso a).

Pueden colocarse varios comandos en una misma línea separándolos con ',', si no se quiere que se muestre el resultado de algún comando se finaliza el comando con ';'.

Se permiten comentarios, de gran ayuda a la hora de realizar scripts para ejecutar.

- % línea de comentario, todo lo que esté a su derecha se considera comentario.

Se puede extender un comando a más de una línea con '...'.

```
>esto\_es\_muy\_largo...
>=17
esto\_es\_muy\_largo = 17
```

3.1. Variables predefinidas

- ans último resultado
- pi = 3.1416
- e = 2.7183
- i, j número imaginario
- Inf infinito
- NaN indeterminado

4. Vectores

Un vector es definido como un conjunto de datos a los cuales se accede por medio de índices. Es una matriz de una dimensión.

La forma en la que octave se definen vectores es utilizando corchete []. Los elementos de una fila se separan con un espacio ' ' o una coma ','.

```
> v= [1,2,3,5,7,11]
v =

    1    2    3    5    7   11
```

Las columnas por su parte se separan mediante puntos y comas ';'.

```
> w= [1;2;3;5;7;11]
w =

    1
    2
    3
    5
    7
   11
```

4.1. Secuencias

En octave podemos crear vectores de secuencas utilizando los dos puntos, p:q:r, donde p sería el valor en el que se iniciaría la secuencia, r el valor final y q el intervalo. El valor q es opcional y si se obvía el intervalo por defecto es 1.

```
> 1:10
ans =

    1    2    3    4    5    6    7    8    9   10
> 1:2:10
ans =

    1    3    5    7    9
```

Además hay otras dos funciones para crear vectores con secuencias separadas x valores, *linspace* y *logspace*. El primero separa los números uniformemente y el segundo logarítmicamente.

```
> linspace(0,15,6)
ans =

    0    3    6    9   12   15
```

Hay dos funciones especiales para crear filas o columnas de unos o ceros, *ones* y *zeros*

```
> ones(1,5)
ans =

   1   1   1   1   1
```

4.2. Funciones sobre vectores

- length *devuelve la longitud del vector*

5. Matrices

Las matrices se introducen por filas, cuyos elementos se separan con ' ' o ',', y que se separan unas de otras con ';'.

```
> M=[1 2 3; 6 2 0; 0 1 -2]
M =

   1   2   3
   6   2   0
   0   1  -2
```

La transpuesta de esta matriz se realizaría fácilmente con el simbolo de trasposición M'.

```
> M'
ans =

   1   6   0
   2   2   1
   3   0  -2
```

Se pueden realizar operaciones entre matrices como su suma de matrices *(M+N)*, resta de matrices *(M-N)*, multiplicación de matrices *(M*N)*[1], multiplicación elemento a elemento *(M.*N)*[2], calcular su matriz inversa *inv(M)*, potenciación o divisiones izquierda y derecha.

5.1. Resolución de ecuaciones

En octave se pueden resolver sistemas de ecuaciones de una forma sencilla utilizando matrices.
Partiendo de las dos ecuaciones:

$$ax + by = c$$
$$dx + ey = f$$

Construimos las matrices siguiente:

```
> a=[a b;d e]
a =

   a   b
   d   e

> b=[f;c]
b =

   c
   f
```

[1] M debe tener el mismo número de colúmnas que N.

[2] Aunque no tiene mucho sentido a la hora de trabajar con matrices en el cálculo numérico si que puede ser útil si se consideran las matrices con datos ordenados en forma matricial.

Donde en la primera matriz colocamos los coeficientes que acompañan a las incógnitas, si en una encuación no aparece una incognita pondremos un 0 en su posición en la matriz. Y en la segunda pondremos los valores de la parte derecha de las ecuaciones. Finalmente el sistema se resolvería:

```
> res=a\b
res =

   X
   Y
```

6. Scripts en octave

Para explicar esta parte voy a suponer que se tienen unos conocimientos básicos de programación, ya que no es este documento el lugar donde explicar IFs, bucles, funciones, etc... por lo que únicamente mostraré la forma de utlizarlos en octave.

Los scripts en Octave son archivos con extensión *.m*. Octave es un programa para el cálculo numérico, por lo que las posibilidades de los scripts son limitadas. Aún así el programa nos da la posibilidad de utilizar operadores lógicos, condiciones, funciones y bucles.

6.1. Operadores lógicos

Los operadores lógicos permitidos son:

$$\&, —, \tilde{}$$

que corresponden a and, or y negación respectivamente.

6.2. Condiciones

6.2.1. IF

```
IF condición1
    cuerpo1
ELSEIF condición2
    cuerpo2
ELSE
    cuerpo3
ENDIF
```

6.2.2. SWITCH

```
SWITCH expresión
CASE etiqueta1
    listadecomandos1
CASE etiqueta2
    listadecomandos2
 ...
OTHERWISE
    listadecomandosFinal
ENDSWITCH
```

6.3. Bucles

6.3.1. FOR

```
FOR var = expresión
    cuerpo
ENDFOR
```

6.3.2. DO-UNITL

```
DO
    cuerpo
UNTIL (condición)
```

6.3.3. WHILE

```
WHILE (condicion)
    cuerpo
ENDWHILE
```

6.4. Funciones

```
FUNCTION [salida1, salida2] = nombre_funcion (argumentos)
  cuerpo
ENDFUNCTION
```

Estas funciones además de en un script se pueden definir en una consola ejecutando octave y utilizarlas posteriormente utilizando su nombre como si de una función predefinida por octave se tratase.

6.5. Recoger información del usuario

Si queremos que el usuario aporte alguna información para el cálculo debemos ejecutar el comando *(*input).

```
kilometros = input('Introduzca los kilometros recorridos: ');
```

6.6. Ejecución

Para ejecutar un programa en octave debemos arrancar el programa en una consola y escribir el nombre del script en la consola de octave.

7. Bibliografía