Breve introducción al OCTAVE

Niurka Rodríguez Quintero,
Correo Electrónico: `niurka@us.es` Página WWW: `http://euler.us.es/~niurka`

Índice

1. Introducción

OCTAVE	$\Longrightarrow$	Lenguaje numérico de programación de libre acceso

1.1. Características principales del OCTAVE

- Programa específico de **Cálculo Numérico**.
  - Sólo opera con **Números**.
  - Se puede considerar como una calculadora programable **muy potente**.
- Programa muy popular entre **estudiantes**, **ingenieros**, **técnicos** e **investigadores** debido a sus características:
  - Programa de **libre acceso**.
  - Programa **interactivo**.
  - **Capacidades Gráficas** sencillas.
  - Posee gran cantidad de **Funciones** de todos los tipos.
  - **Lenguaje de programación** de alto nivel similar a *Fortran, C, Pascal o Basic*, pero **más fácil** de aprender. Su lenguaje de programación es igual al de MATLAB

1.2. Acceso al OCTAVE desde el entorno *Unix*

- Ejecutar la instrucción octave desde cualquier ventana
- Aparece la siguiente ventana del octave:

```
octave:1>
```

- Para salir ejecutamos exit o quit desde la ventana del octave.

1.3. Accesos al OCTAVE desde windows

- Hacer *doble click* sobre el **icono** de **OCTAVE**.
- Al igual que en el entorno *Unix*, aparece la ventana del octave

1.4. Algunas instrucciones de utilidad

> **pwd** nos dice en que directorio nos encontramos. Por ejemplo, una respuesta podría ser:

```
C:\octave\bin
```

> **ls** nos da una lista de los ficheros y los directorios

> **cd** *nombre* nos permite cambiar al directorio *nombre*.

1.5. Operaciones básicas. Funciones elementales.

+	adición
-	sustracción
*	multiplicación
^	potenciación
\	división izquierda
/	división derecha

exp	log	exponencial y logaritmo neperiano
sin	cos	seno y coseno
abs	sqrt	valor absoluto y raíz cuadrada
round	floor	ceil funciones que redondean

Ejemplos

```
> 2 + 3                  > 2 * 2
ans =  5                   ans = 4

> sin(pi/6)              > 2/6
ans = 0.50000        ans =0.33333

> log(5^3)              > round(4.5)
ans = 4.8283           ans = 5

> ceil(4.5)               > floor(4.5)
ans = 5                     ans = 4
```

- Observe que: **los () se reservan sólo para escribir el argumento de las funciones**.

1.6. Ayudas y normas generales del *OCTAVE*

- El comando `help` nos proporciona información sobre las funciones del *OCTAVE*:

```
> help round   % redondea al entero m'as cercano
> help floor   % redondea por defecto
> help ceil    % redondea por exceso
```

- Las **flechas:** ↑ y ↓ permiten **recuperar comandos anteriores**.

- Las **flechas:** ← y → permiten movernos a lo largo de una línea de instrucciones y **corregirla**.

- *OCTAVE* distingue entre mayúsculas y minúsculas:

```
> ceil(2.3)
ans =
    3
```

NO es lo mismo que:

```
> Ceil(2.3)
error: 'Ceil' undefined near line 22 column 1
```

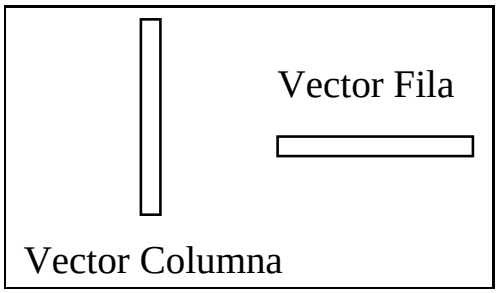
- Podemos asignar variables con determinados nombres a las expresiones numéricas (*números*, *constantes*).

```
> m = 9.11e-31;  q = -1.6e-19;
> r = abs(q)/m
r =  1.7563e+11
> 3e+8
ans = 300000000
> m*(ans^2)
ans = 8.1990e-014
```

- Los nombres de estas variables pueden formarse utilizando **letras**, **dígitos**, etc.
- Las variables se pueden borrar con el comando `clear` *nombre*.
- Asignación por defecto: si a una expresión numérica no le asignamos un nombre, *OCTAVE* crea la variable `ans`.
- El comando who nos permite conocer los nombres de las variables asignadas. Ejecute **who**.

2. Vectores

vector: conjunto de números $a_1, a_2, ..., a_n$



2.1. Vectores fila y vectores columnas.

- Para definir *vectores* utilizamos los **corchetes** [ ].
- Los elementos de una fila se separan mediante **espacios en blanco** o **comas**.
- Los elementos de una columna se separan por **puntos y comas** o por **nuevas líneas**.

```
> A=[ 1 2 3 4 5 6 7 8 9];     % vector fila

> vecf=[1,2,3,4,5,6,7,8,9];   % vector fila


> B=[ 1
> 2
> 3
> 4 ];           % vector columna

> vecc=[1;2;3;4];    % vector columna
```

- El % se utiliza en *OCTAVE* para escribir comentarios.

2.2. Utilización de los dos puntos :

- *1er elemento del vector* : *incremento* : *último elemento*
- *1er elemento del vector* : *último elemento* $\Longrightarrow$ *OCTAVE* toma el *incremento* = 1

```
> A=1:2:5
A =  1    3    5

> B=[5:-1:3]'
B =
  5
  4
  3

>  x = [0:0.1:2*pi]';
>  y = sin(x);
>  [x y]
```

2.3. Funciones sobre los vectores

- **length** calcula el número de elementos de un vector (longitud de un vector). Su argumento es el propio vector.

- **sin** si el argumento de la función seno es un vector entonces, esta función calcula el seno de cada elemento del vector. El argumento de las funciones trigonométricas debe de estar expresado en radianes.

```
> v=[0.1:0.1:0.6]
v = 0.1000 0.2000 0.3000 0.4000 0.5000 0.6000

> sin(v)
ans = 0.0998 0.1987 0.2955 0.3894 0.4794 0.5646

> length(v)
ans =  6
```

2.4. Operaciones vectoriales. Operaciones puntuales

Operaciones		Operaciones puntuales	
$+$	suma	$+$	suma
$-$	resta	$-$	resta
$*$	multiplicación	$.*$	multiplicación
$/$	división derecha	$./$	división derecha
$\hat{}$	potenciación	$.\hat{}$	potenciación

- **Las operaciones puntuales** se utilizan para realizar operaciones entre vectores y matrices. Por ejemplo si queremos multiplicar cada elemento del vector x por el correspondiente elemento del vector y, siendo $x = (1, -2, 4, 2)$ e $y = (3, -5, 4, 0)$, escribimos

```
% definimos los elementos de los vectores
> x = [1 -2 4 2]; y = [3 -5 4 0];
 % utilizamos la multiplicaci\'on puntual
> x.*y
ans=
    3 10 16 0
```

3. Editor del OCTAVE. Programación.

Para editar un programa nuevo, desde la misma ventana del *OCTAVE*, escibir el comando **edit**.

3.1. Tipos de m-files

• Archivos de instrucciones (estos archivos se ejecutan directamente desde la ventana del *OCTAVE*).
prac.m

```
% Este es el programa prac.m y se guardan los valores
% de la intensidad (I) y del voltage (V)
I=[0.01;0.02;0.03;0.036;0.032;0.028;0.024;0.018;0.012;0.008]; V=[3.04;6.41;9.84;11.73;10.61;9.02;7.65;5.71;3.79;2.55;
```

Para ejecutarlo y realizar operaciones con las variables guardadas:

```
> clear
> help prac
% En el programa prac.m se guardan los valores
% de la intensidad (I) y el voltage (V)
> prac              % ejecutamos el programa
> plot(I,V)         % dibuja V(I)
```

Observaciones importantes:
⋆ Los ficheros deben ser escritos en *ASCII*. La extensión del programa es **.m**
⋆ El programa debe de ejecutarse desde el directorio donde se encuentre.

• Véanse los programas ejemplo1.m y ejemplo2.m, donde se calculan los valores medios de n medidas y sus correspondientes errores absolutos. Pare ejecutarlos, solo hay que escribir en la ventana del octave > **ejemplo1** o > **ejemplo2** sin la extensión **.m** del programa.

4. Gráficos

4.1. Gráficos en 2 dimensiones

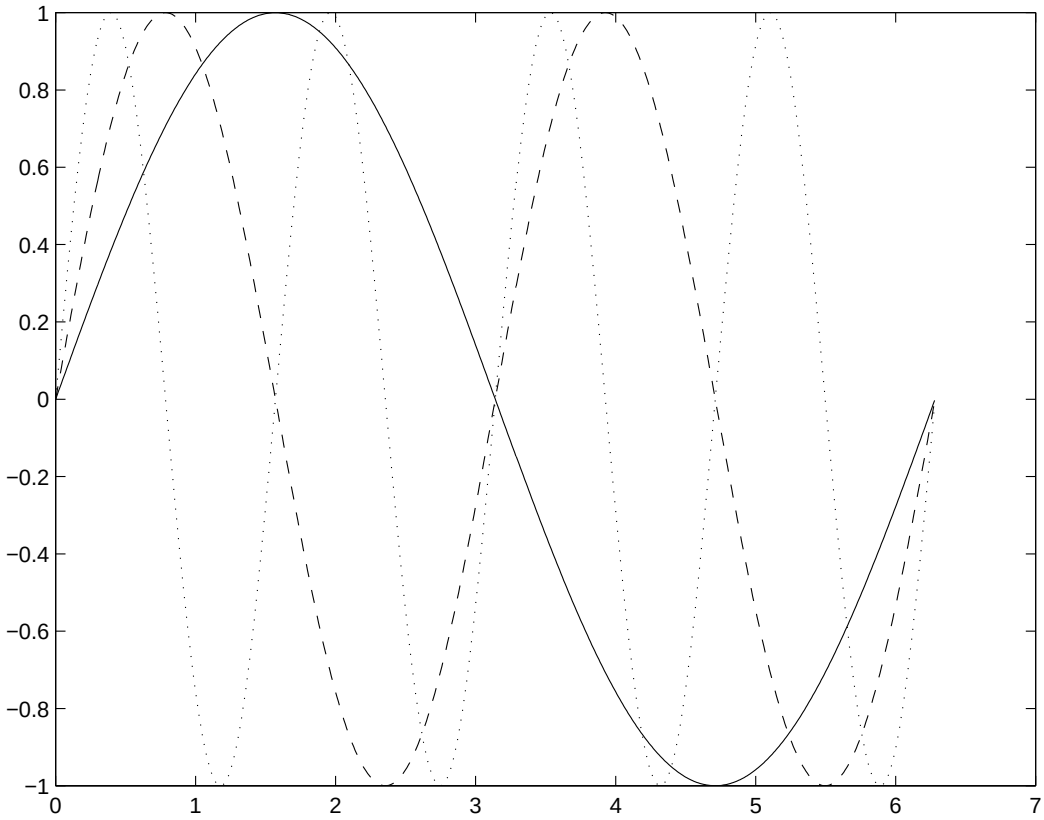
• Dada un conjunto de n puntos (x_i, y_i), $i = 1, 2, ..., n$, definidos previamente en los vectores x e y del *OCTAVE*; la instrucción **plot(x,y)** nos dibuja los pares de puntos (x_i, y_i) unidos por líneas. Ejecuta **help plot**.

• **plot(x,y,'cts')**, donde **c** es el color de las líneas, **t** el tipo de línea y **s** el símbolo que usa *OCTAVE* para dibujar los puntos.

Color	Tipos de líneas	Símbolos
y yellow	. point	- solid
m magenta	o circle	: dotted
c cyan	x x-mark	-. dashdot
r rojo	+ plus	– dashed
g green	* star	
b blue	s square	
w white	d diamond	
k black		

Ejemplo: Gráficos múltiples. Varias curvas en el mismo gráfico.

```
> x=0:.01:2*pi;
> y1=sin(x);y2=sin(2*x);y3=sin(3*x);
> plot(x,y1,x,y2,'--',x,y3,'.')
```



- Utilización del **hold on**, **hold off** y el **clf**.

```
> clf
> x=0:.01:2*pi;
> y1=sin(x);y2=sin(2*x);y3=sin(3*x);
> plot(x,y1)
> hold on
> plot(x,y2,'--'); plot(x,y3,'.')
> hold off
```

Funciones gráficas

- `clf` borra la pantalla gráfica.
- `hold on` permite añadir al último gráfico una nueva figura.
- `hold off` desactiva el hold on.
- `axis([` *xmin,xmax,ymin,ymax* `])` escala la ventana gráfica.
- `grid` dibuja una retícula cuadrada.
- `xlabel('`*nombre_del_eje_x*`')`, `ylabel('`*nombre_eje_y*`')`, `title('`*título*`')`.

5. Grabar y leer datos en ficheros. Impresión de las gráficas

• La instrucción `save fname.mat x y z` graba las variables `a` b `c` en el fichero `fname.mat` (*archivos mat o MAT-files*).

• La instrucción `load fname.mat` recupera las variables guardadas en el archivo `fname.mat`.

Ejemplo

```
> clear; clf
> x = [0:pi/60:2*pi]; y = sin(x.^2);
> save datos.mat x y
> clear
> who
> load datos.mat
> who
> x
> plot(x,y)
```

• Para imprimir la figura en un archivo postscript utilizamos el comando `print -dps nfile.ps`. Por ejemplo, **print -dps fig.ps** crea el archivo postscript, fig.ps, de la figura que este en la ventana gráfica del *OCTAVE*.