

Guía de laboratorio

Profesor: Leonardo Hernández R.
Semestre académico: I semestre académico de 2018

Laboratorio	2 – Introducción a Octave
Estudiantes por grupo	Hasta 3 estudiantes
Tiempo promedio por estudiante estimado	5h

1 Objetivo

El objetivo del laboratorio es:

Aprender fundamentos de lenguaje Octave necesarios para experimentar con algoritmos de aprendizaje de máquina.

Octave es un lenguaje interpretado de alto nivel, pensando principalmente para cálculos y experimentos numéricos. También provee capacidades gráficas extensivas para visualización y manipulación de datos.

El lenguaje Octave es muy similar a MATLAB, de manera que la mayoría de los programas son fácilmente portables.

Octave es distribuido bajo los términos de GNU General Public License.

2 Procedimiento

Para la elaboración del laboratorio se suministra la carpeta *lab2*, que contiene los siguientes archivos:

- Esta guía
- La plantilla del laboratorio
- Código en lenguaje Octave

El procedimiento para realizar el laboratorio consiste en:

- Estudiar esta guía
- Realizar en la carpeta *lab2* los ejercicios que en algunas de las secciones de la guía se solicitan, 9 ejercicios en total.
- Diligenciar la plantilla de laboratorio

3 Introducción a Octave y ejercicios del laboratorio

En las siguientes secciones se explican fundamentos del lenguaje Octave, además se solicita la realización de unos ejercicios que se deben entregar como resultado del laboratorio. Todos estos ejercicios deben realizarse en la carpeta *lab2*.

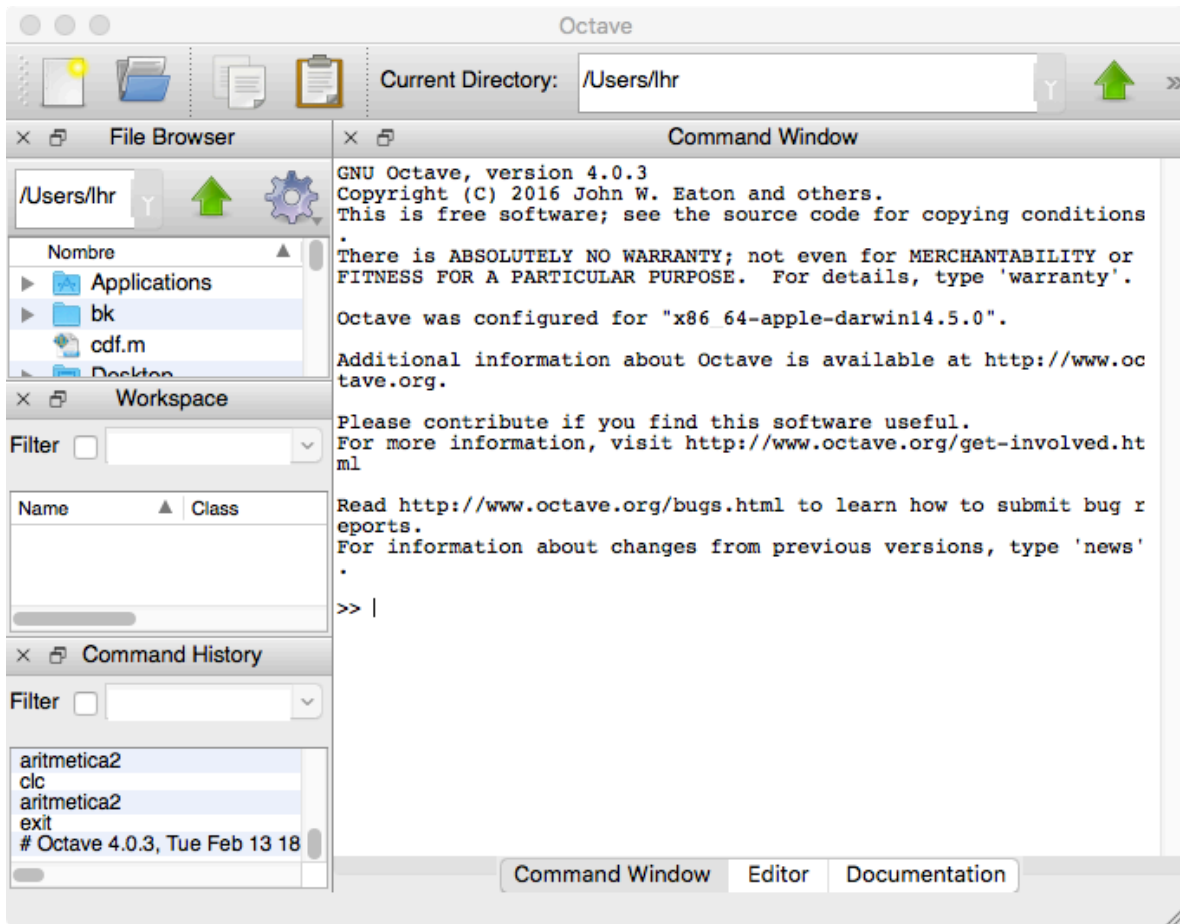
Debe tener instalado Octave en su computador, de no ser así, puede descargar el instalador mediante el enlace correspondiente en la plataforma Moodle.

Nota: En los computadores del laboratorio de Ingeniería de Sistemas y Computación ya se encuentra instalada esta herramienta.

3.1 Ejecución de Octave

Para ingresar la aplicación, basta con hacer doble clic en el ícono del escritorio 'Octave 4.2.1 (GUI)'. Véase la Figura 1

Figura 1. GUI de Octave



3.2 Cálculos elementales

Es posible realizar cálculos elementales fácilmente en Octave. Inicialmente se requiere hacer clic en la pestaña *Command Window*, para abrir la consola. Véase la Figura 1. Luego simplemente ingrese las expresiones seguidas de la tecla 'enter'; inmediatamente se mostrará el resultado.

A continuación, se presentan tres ejemplos de cálculos elementales realizados en la ventana de comandos:

```

==> a=3+4*2-20/4
a = 6

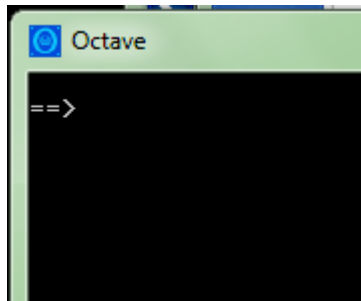
==> b=10^2
b = 100

==> c=sqrt(36)
c = 6

==>

```

Para dejar la consola en blanco, use el comando `clc`, clear console. Así queda la consola, después de ejecutarse el comando `clc`



Para ayuda sobre alguna función utilice el comando `help`, por ejemplo:

```

==> help log
'log' is a built-in function

-- Mapping Function: log (X)
  Compute the natural logarithm, 'ln (X)', for each element of X.
  To compute the matrix logarithm, see *note Linear Algebra::.

  See also: exp, log1p, log2, log10, logspace

Additional help for built-in functions and operators is
available in the on-line version of the manual. Use the command
'doc <topic>' to search the manual index.

Help and information about Octave is also available on the WWW
at http://www.octave.org and via the help@octave.org
mailing list.

```

3.3 Creación de matrices

En la siguiente imagen se observa cómo se inicializa un vector fila:

```
==> A=[ 2 3 ?]  
A =  
  
    2    3    ?
```

Usando puntos y comas para separar los elementos, se inicializa un vector columna en lugar de un vector fila.

```
==> A=[2; 3; ?]  
A =  
  
    2  
    3  
    ?
```

También es sencillo inicializar una matriz.

```
==> A=[4 5 9; 3 4 ?; 14 8 0]  
A =  
  
    4    5    9  
    3    4    ?  
   14    8    0
```

La función *magic(n)* crea un cuadrado mágico de n filas por n columnas. En un cuadrado mágico, la suma de cada fila, de cada columna y de cada diagonal es igual al mismo valor. En el ejemplo de la imagen, cada fila, cada columna y cada diagonal suma 34.

```
==> A=magic(4)  
A =  
  
   16    2    3   13  
    5   11   10    8  
    9    7    6   12  
    4   14   15    1
```

La utilidad de la función *magic* es generar matrices rápidamente para hacer pruebas, o experimentar instrucciones.

La referencia a elementos de un arreglo se hace mediante subíndices encerrados entre paréntesis. Los elementos de los vectores, y las filas y las columnas de las matrices se numeran desde 1. Siguen algunos ejemplos:

```
==> a=[5 7 8 2]
a =
     5     7     8     2

==> b=a(2)+a(4)
b = 9
```

```
==> a=[8; 3; 2; 4]
a =
     8
     3
     2
     4

==> b=a(2)+a(4)
b = 7
```

```
==> A=magic(4)
A =
    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1

==> a=A(2,3)+A(3,1)
a = 19
```

3.4 Aritmética de matrices

Para ilustrar la aritmética de matrices, se usarán las matrices A y B , inicializadas de la siguiente manera:

```
==> A=[3 4; 5 6]
A =
     3     4
     5     6

==> B=[2 -7; 8 -9]
B =
     2    -7
     8    -9
```

En la imagen se muestran ejemplos de instrucciones para sumar, restar y la multiplicar las dos matrices anteriores:

```
==> C=A+B
C =
     5    -3
    13    -3

==> C=A-B
C =
     1    11
    -3    15

==> C=A*B
C =
    38   -57
    58   -89
```

La transposición de matrices también es muy sencilla en Octave:

```
==> C=A'
C =
     3     5
     4     6
```

También se cuenta en Octave con operadores elemento a elemento. Para ilustrar estos operadores, se utilizará la matriz *A*, inicializada de la siguiente manera:

```
==> A=[7 4; 2 8]
A =
     7     4
     2     8
```

Con la siguiente instrucción se suma 4 a cada uno de los elementos de la matriz *A*.

```
==> B=A+4
B =
    11     8
     6    12
```

Ahora se presenta una instrucción que resta 5 a cada uno de los elementos de *A*.

```
==> C=A-5
C =
     2    -1
    -3     3
```

Igualmente, se pueden multiplicar o dividir por un valor cada uno de los elementos de una matriz.

```
==> E=A*3
E =
    21    12
     6    24
```

```
==> F=A/4
F =
    1.75000    1.00000
    0.50000    2.00000
```

Para elevar al cuadrado cada uno de los elementos de una matriz, hay que utilizar un punto, para diferenciar la operación de la definida en Álgebra Lineal.

```
==> A
A =
     7     4
     2     8

==> B=A.^2
B =
    49    16
     4    64

==> B=A^2
B =
    57    60
    30    72

==> C=A*A
C =
    57    60
    30    72
```

También se cuenta con la multiplicación y la división elemento a elemento entre dos matrices, como se puede observar en las figuras. Note el punto antes de los operadores, para diferenciar los operadores de los definidos en Álgebra Lineal.


```

==> A=[4 7; 2 8]
A =
     4     7
     2     8

==> B=[5 3; 1 6]
B =
     5     3
     1     6

```

```

==> C=A.*B
C =
    20    21
     2    48

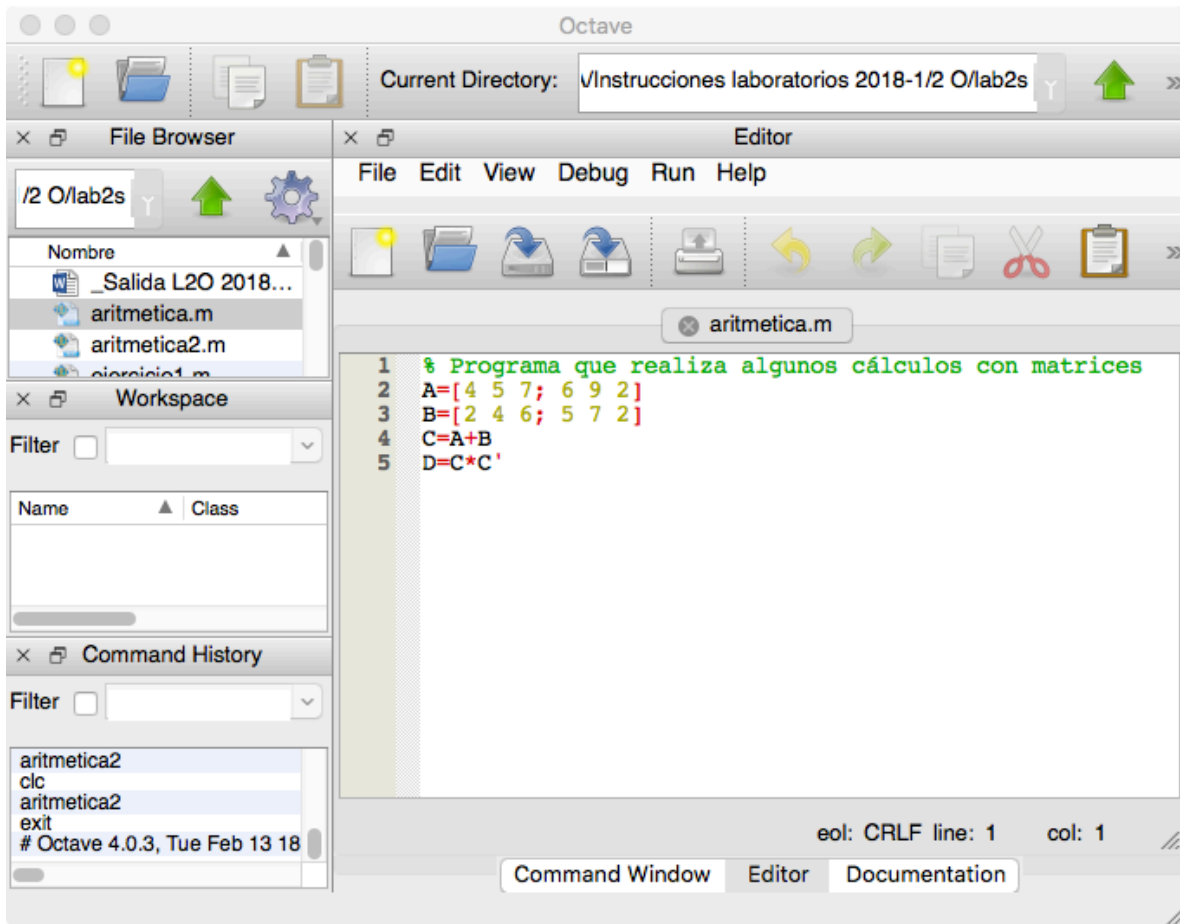
==> C=A./B
C =
    0.800000    2.333333
    2.000000    1.333333

```

3.5 Programas

Además de permitir ejecutar instrucciones una a una, en su ventana de comandos, el IDE (Entorno de desarrollo integrado) de Octave permite ejecutar programas. Como ejemplo, se presenta en la Figura 2, un programa que realiza algunas operaciones con matrices.

Figura 2. Programa de ejemplo



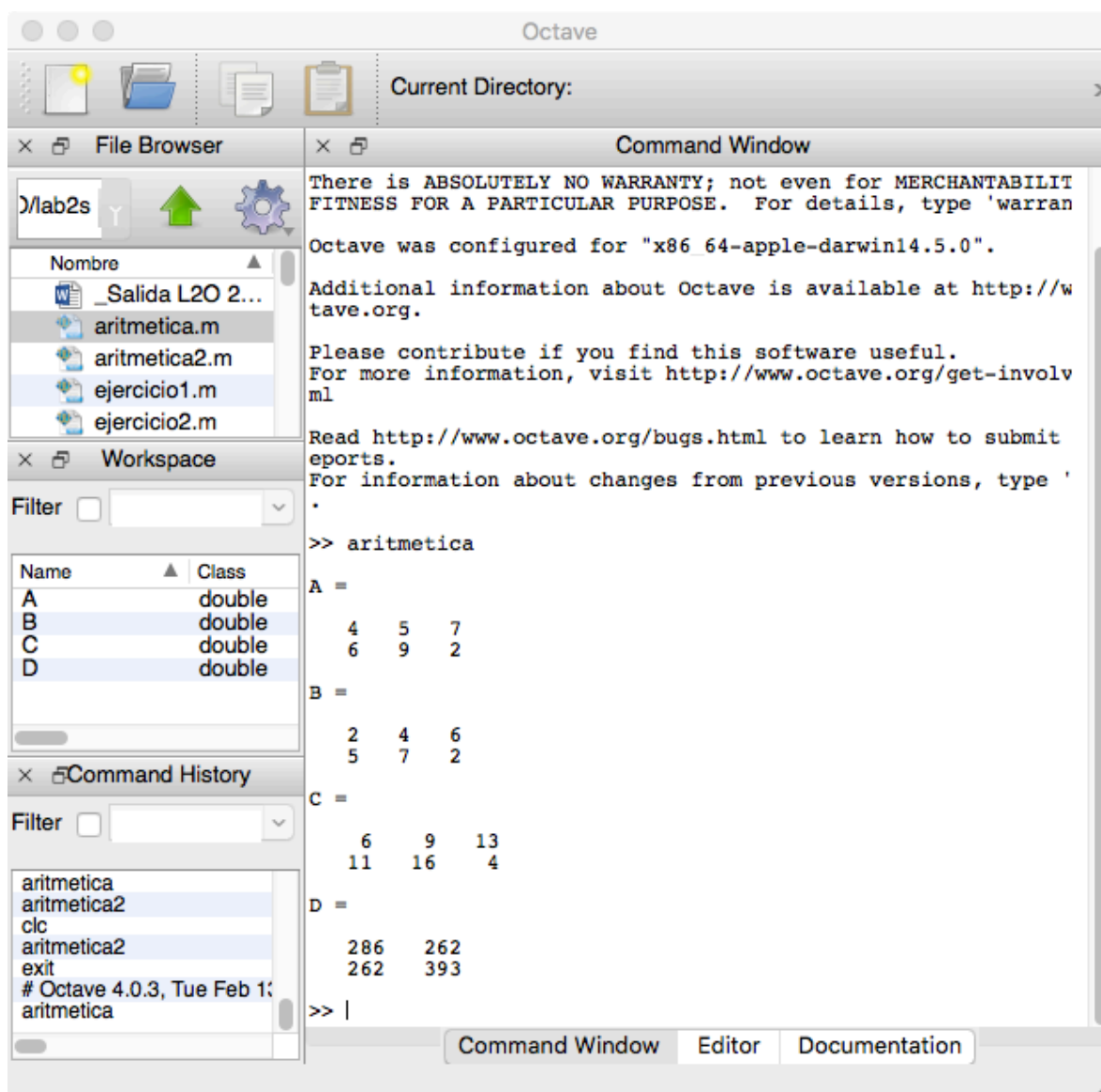
Nótese que se está usando la pestaña 'Editor'.

Los programas en Octave deben tener extensión `.m`.

El símbolo de porcentaje se usa para incluir comentarios dentro del código.

Para ejecutar el programa, la ruta-nombre de la carpeta que lo contiene debe encontrarse en el cuadro de texto 'Current directory', luego simplemente se hace clic en el ícono 'Save File and Run' y luego en la pestaña 'Command Window'. Véase la Figura 3.

Figura 3. Ejemplo de salida de un programa



Si no se desea que ciertos resultados se muestren en la consola, basta con agregar punto y coma al final de las correspondientes instrucciones. En la Figura 4 y en la Figura 5, se puede observar cómo se reduce la salida, utilizando esta técnica.

Figura 4. Instrucciones finalizadas con punto y coma

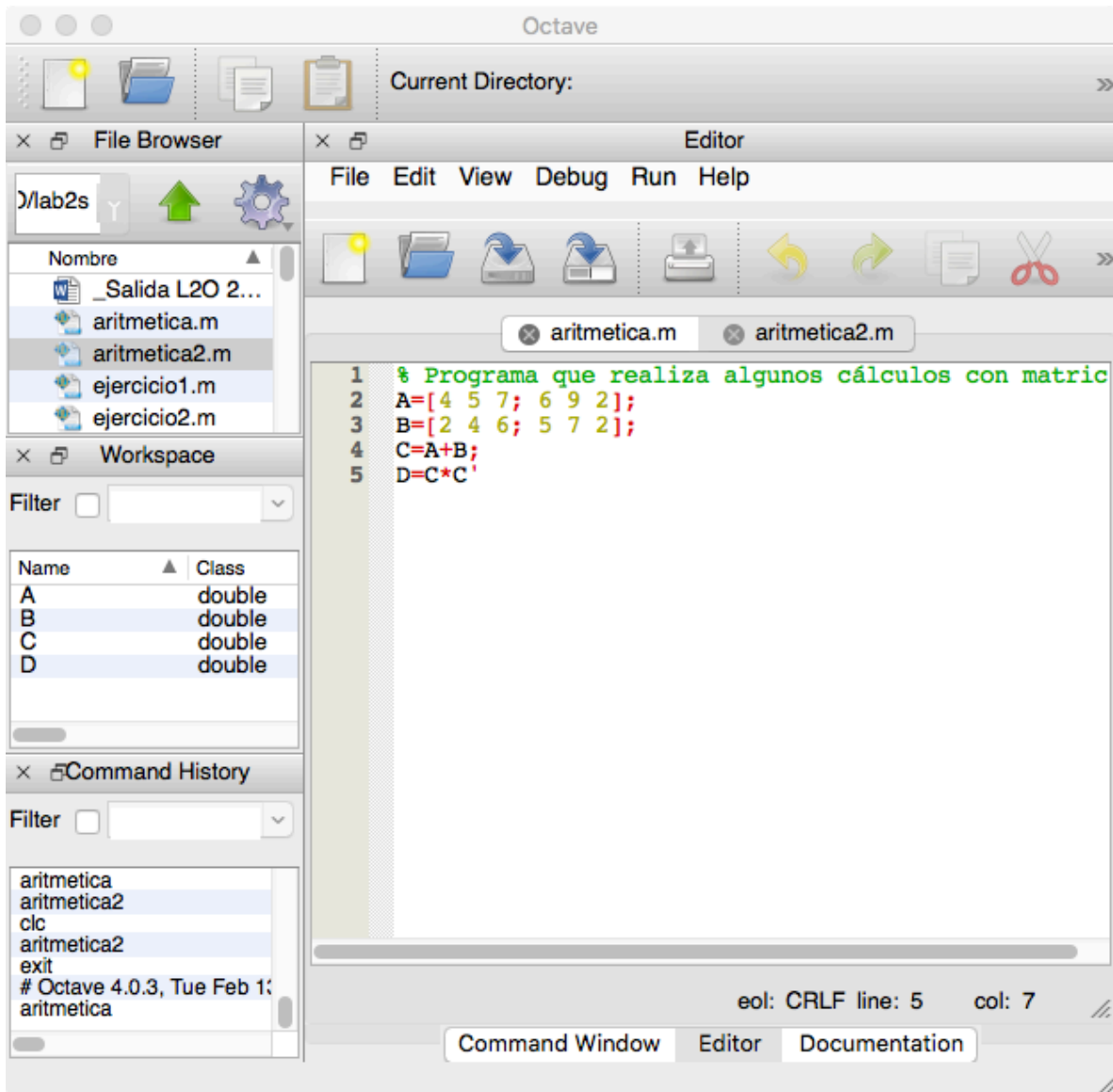
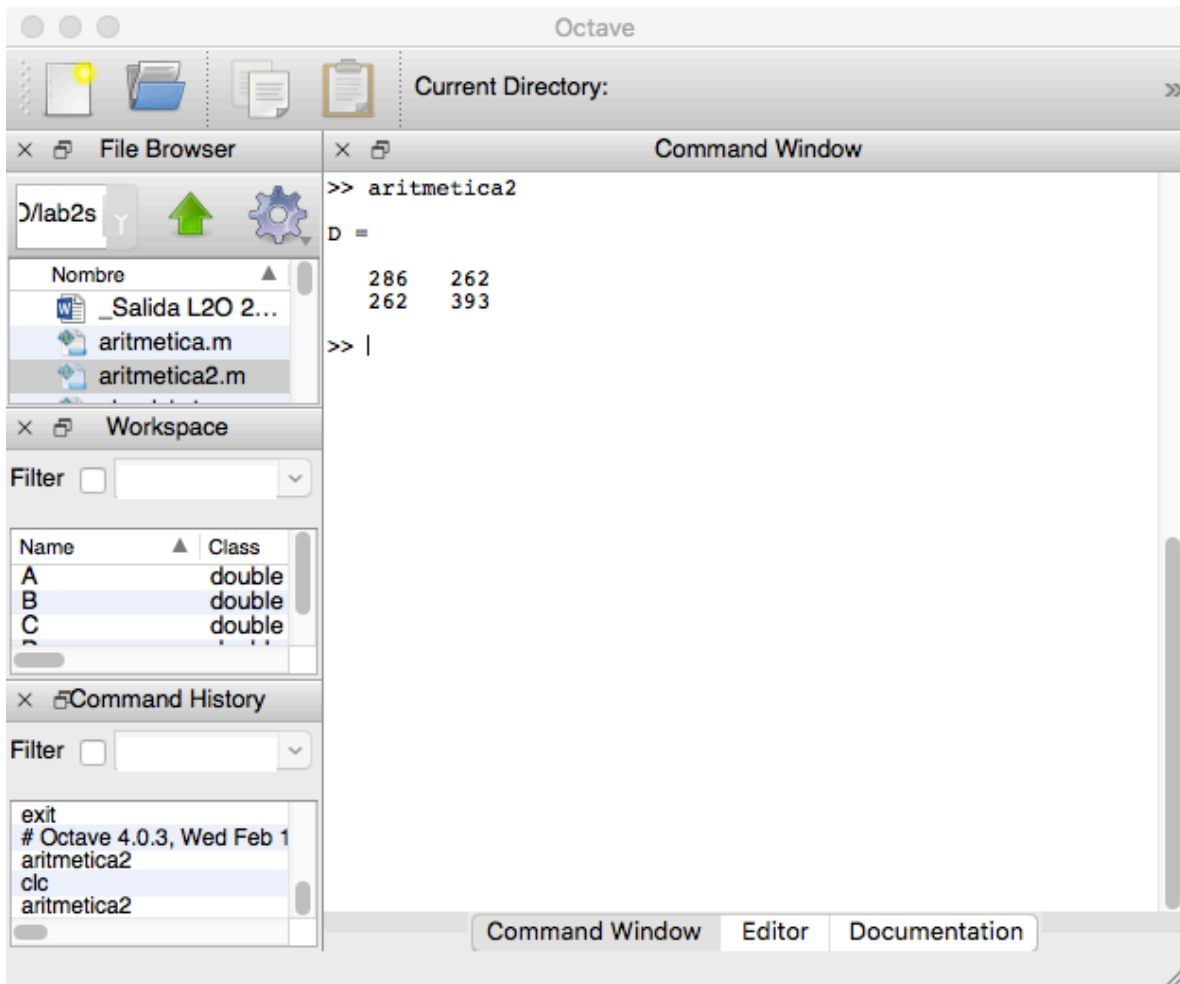


Figura 5. GUI de Octave con la salida de un programa



3.5.1 Ejercicio 1

Elabore en la carpeta `lab2` un programa que realice las siguientes inicializaciones y cálculos:

$$\text{i) } A = \begin{bmatrix} 4 & 6 & 9 & 2 \\ 1 & 2 & 3 & 4 \\ 4 & 3 & 5 & 8 \end{bmatrix}$$

$$\text{ii) } B = \begin{bmatrix} 2 & 6 & 1 & 3 \\ 4 & 3 & 1 & 9 \\ 6 & -3 & 6 & 5 \\ 2 & 6 & 8 & 8 \end{bmatrix} f$$

iii) Hallar la matriz $C = A + 1$

iv) Halle la matriz D , de 4×4 donde $D_{ij} = (B_{ij})^2$

v) Hallar la matriz $E = CD$

vi) Finalmente, halle $F = E^T$

Si el programa queda bien implementado, debe obtener la matriz F de 4×3 elementos, cuyo menor elemento es 220 y su mayor elemento es 1095

3.5.2 Ejercicio 2

Elabore en la carpeta *lab2* un programa que realice las siguientes inicializaciones y cálculos:

i) $A = \begin{bmatrix} 6 & 4 & 8 \\ 3 & 6 & 8 \end{bmatrix}$

ii) $B = \begin{bmatrix} 8 & 1 & 3 \\ 6 & -2 & 4 \\ 1 & -6 & 4 \end{bmatrix}$

iii) $C = AB$

iv) $D = A^T C$

v) $E = B + D$

vi) $F = \frac{E}{2}$

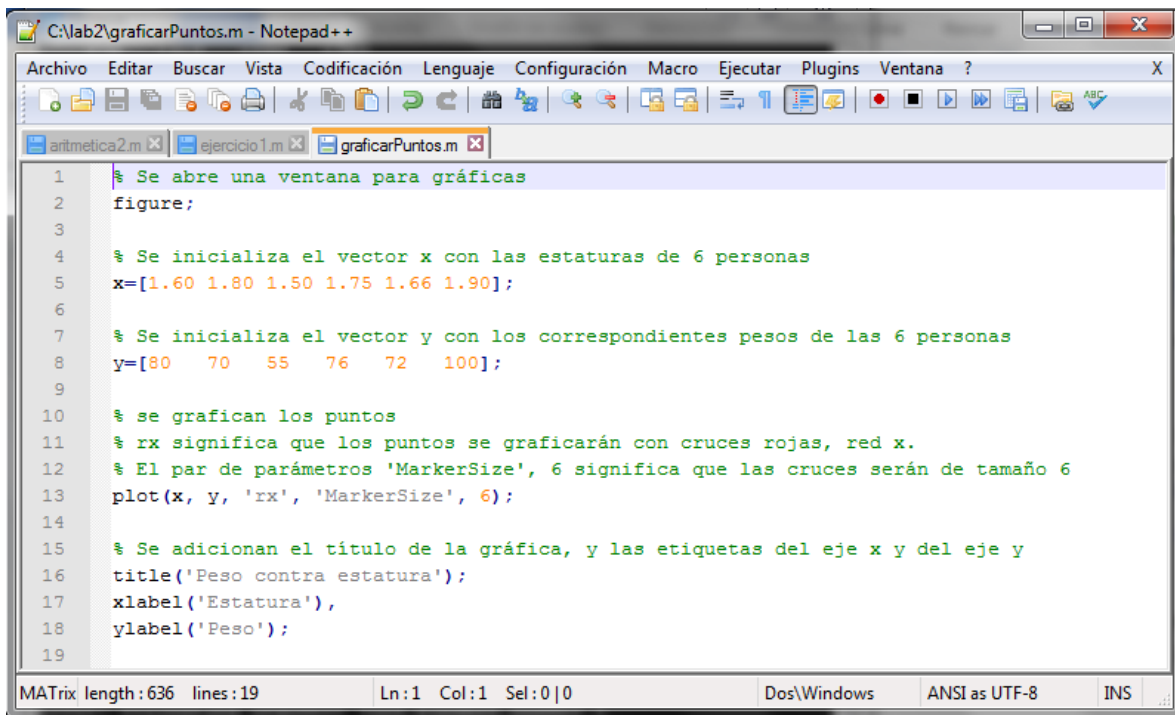
vii) Inicialice G con la matriz que resulta de multiplicar cada elemento de la matriz F por su correspondiente elemento de la matriz B

Si el programa queda bien implementado, deberá poder observar en la consola la matriz G de 3×3 elementos, cuyo menor elemento es -235 y su mayor elemento es 2768

3.6 Gráficas en el plano cartesiano

3.6.1 Graficar puntos

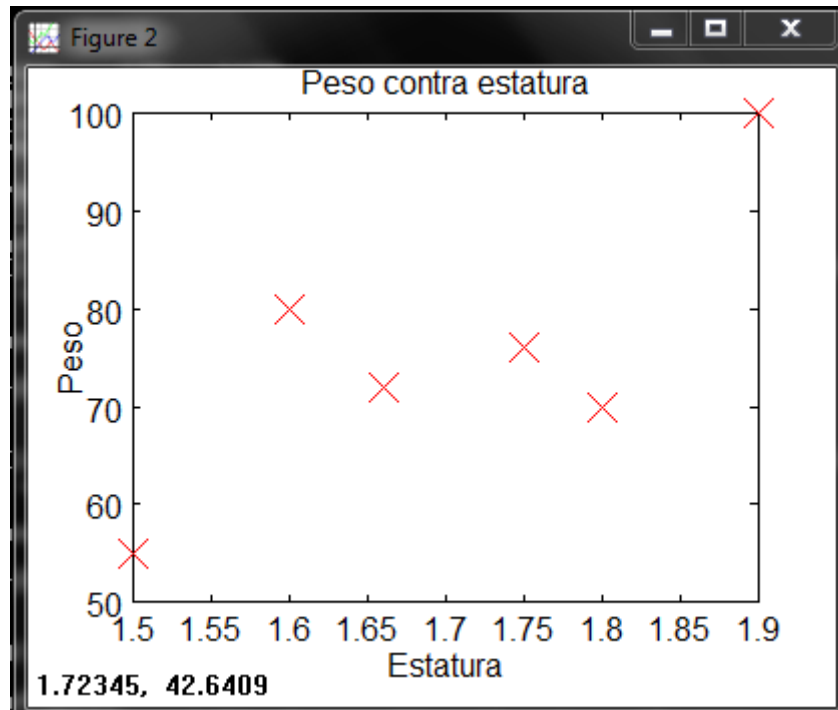
El siguiente ejemplo, el cual se encuentra incluido en la carpeta *lab2* ilustra cómo se grafican puntos en el plano cartesiano. El programa grafica el peso contra la estatura de varias personas. Las instrucciones se encuentran explicadas en los comentarios del programa.



```
1 % Se abre una ventana para gráficas
2 figure;
3
4 % Se inicializa el vector x con las estaturas de 6 personas
5 x=[1.60 1.80 1.50 1.75 1.66 1.90];
6
7 % Se inicializa el vector y con los correspondientes pesos de las 6 personas
8 y=[80 70 55 76 72 100];
9
10 % se grafican los puntos
11 % rx significa que los puntos se graficarán con cruces rojas, red x.
12 % El par de parámetros 'MarkerSize', 6 significa que las cruces serán de tamaño 6
13 plot(x, y, 'rx', 'MarkerSize', 6);
14
15 % Se adicionan el título de la gráfica, y las etiquetas del eje x y del eje y
16 title('Peso contra estatura');
17 xlabel('Estatura'),
18 ylabel('Peso');
19
```

MATrix length: 636 lines: 19 Ln: 1 Col: 1 Sel: 0 | 0 Dos\Windows ANSI as UTF-8 INS

La gráfica resultante es la siguiente.



3.6.2 Gráfica de funciones

Antes de presentar el ejemplo para graficar una función en el plano cartesiano, se explicará la función `linspace()`, de gran utilidad para asignarle los valores a la variable del eje horizontal.

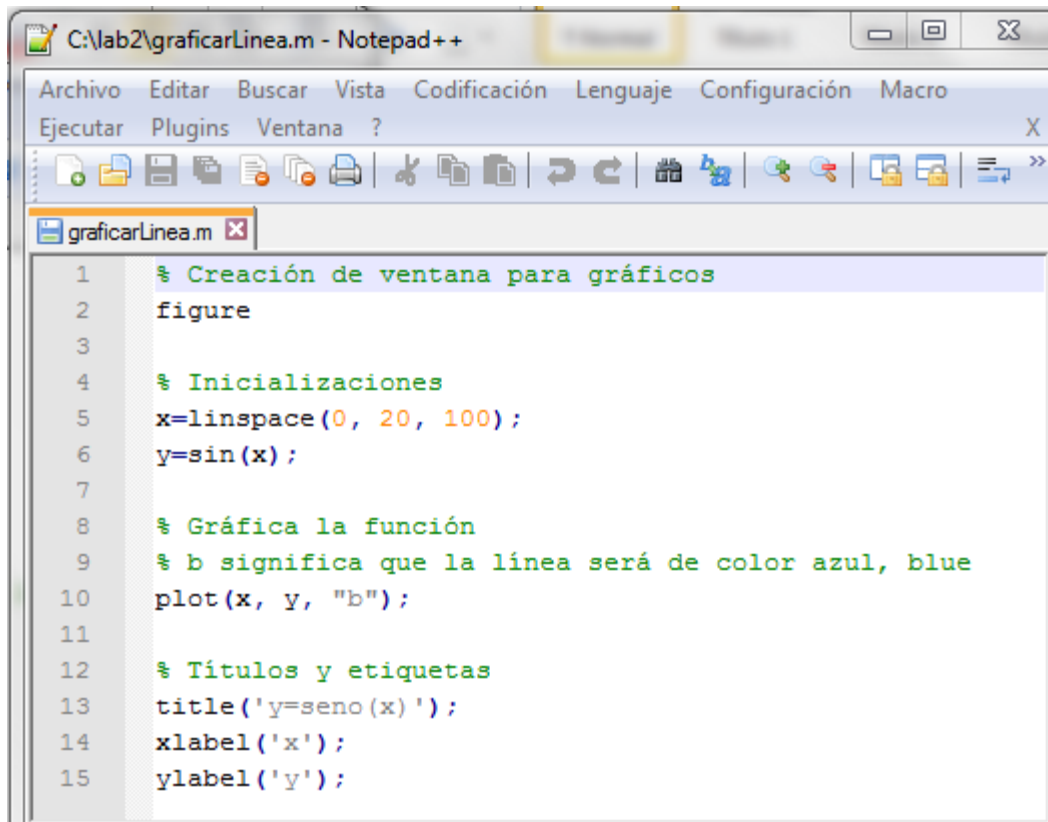
La función `linspace(xi, xf, num)` devuelve un vector con `num` elementos igualmente espaciados entre `xi` y `xf`, por ejemplo:

```
==> x=linspace(0, 2, 5)
x =
    0.00000    0.50000    1.00000    1.50000    2.00000
```

También, previamente a la presentación del ejemplo para elaborar la gráfica en el plano cartesiano, se debe mencionar que en Octave es posible aplicar una función a un arreglo. El resultado es que la función se aplica a cada uno de los elementos del arreglo, como en el siguiente ejemplo:

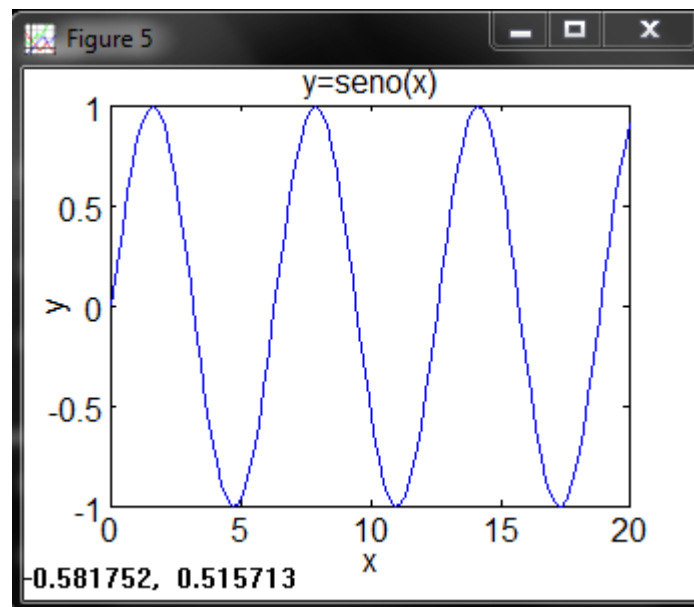

```
==> x=[16, 36, 100];  
==> y=sqrt(x)  
y =  
     4     6    10
```

Ya con los dos elementos anteriores, se procede a presentar un ejemplo de programa que elabora la gráfica de la función *seno* en el plano cartesiano. Se incluyen explicaciones adicionales en los comentarios del código.



```
C:\lab2\graficarLinea.m - Notepad++  
Archivo  Editar  Buscar  Vista  Codificación  Lenguaje  Configuración  Macro  
Ejecutar  Plugins  Ventana  ?  
graficarLinea.m  
1  % Creación de ventana para gráficos  
2  figure  
3  
4  % Inicializaciones  
5  x=linspace(0, 20, 100);  
6  y=sin(x);  
7  
8  % Gráfica la función  
9  % b significa que la línea será de color azul, blue  
10 plot(x, y, "b");  
11  
12 % Títulos y etiquetas  
13 title('y=seno(x)');  
14 xlabel('x');  
15 ylabel('y');
```

La gráfica resultante del anterior programa es la siguiente.



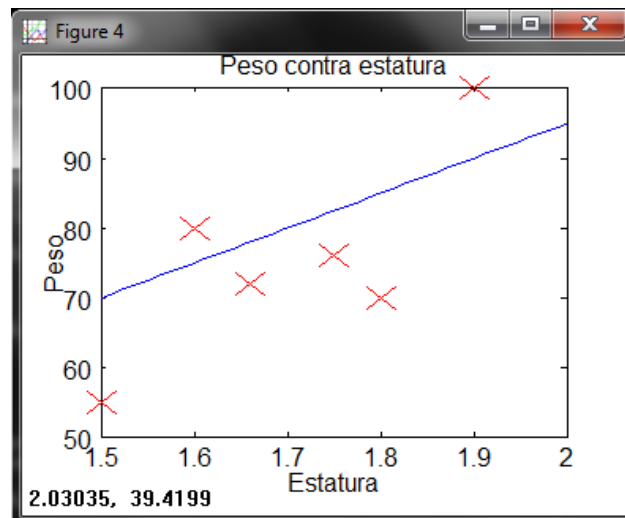
3.6.3 Gráficas superpuestas

También se pueden superponer 2 gráficas en la misma figura, como se explica en los comentarios del siguiente programa:

```
C:\lab2\graficarSuperpuesta.m - Notepad++
Archivo  Editar  Buscar  Vista  Codificación  Lenguaje  Configuración  Macro
Ejecutar  Plugins  Ventana  ?
graficarLinea.m  graficarSuperpuesta.m

1  % Gráfica de peso contra estatura
2  figure;
3  x=[1.60 1.80 1.50 1.75 1.66 1.90];
4  y=[80 70 55 76 72 100];
5  plot(x, y, 'rx', 'MarkerSize', 6);
6  title('Peso contra estatura');
7  xlabel('Estatura'),
8  ylabel('Peso');
9
10 % Superponer la siguiente gráfica a la anterior
11 hold on
12
13 % Gráfica de una recta
14 x=linspace(1.5, 2, 100);
15 y=50*x-5;
16 plot(x, y, "b");
17
18 % No superponer la siguiente gráfica a las anteriores
19 hold off
20
```

La grafica resultante es la siguiente:



3.6.4 Ejercicio 3

Elabore en la carpeta *lab2* un programa que grafique las dos siguientes funciones en un mismo plano cartesiano, con x variando de 0 a 50

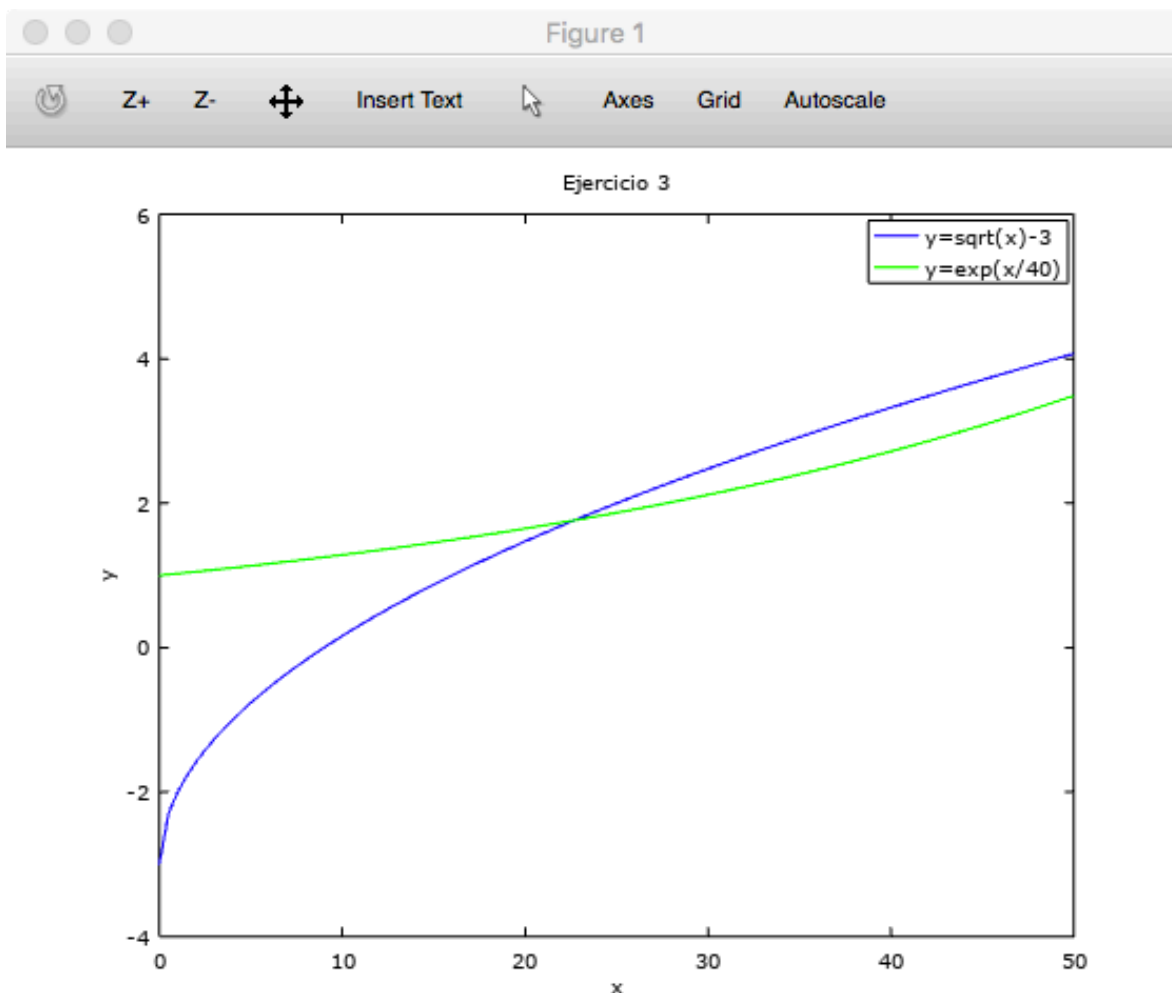
$$y = \sqrt{x} - 3$$

$$y = e^{x/40}$$

La gráfica debe incluir el título de la figura, el nombre de los ejes y las convenciones. Además, la curva debe ser suave.

Debe obtener una gráfica como la de la Figura 6.

Figura 6. Gráfica en el plano cartesiano



3.6.5 Ejercicio 4

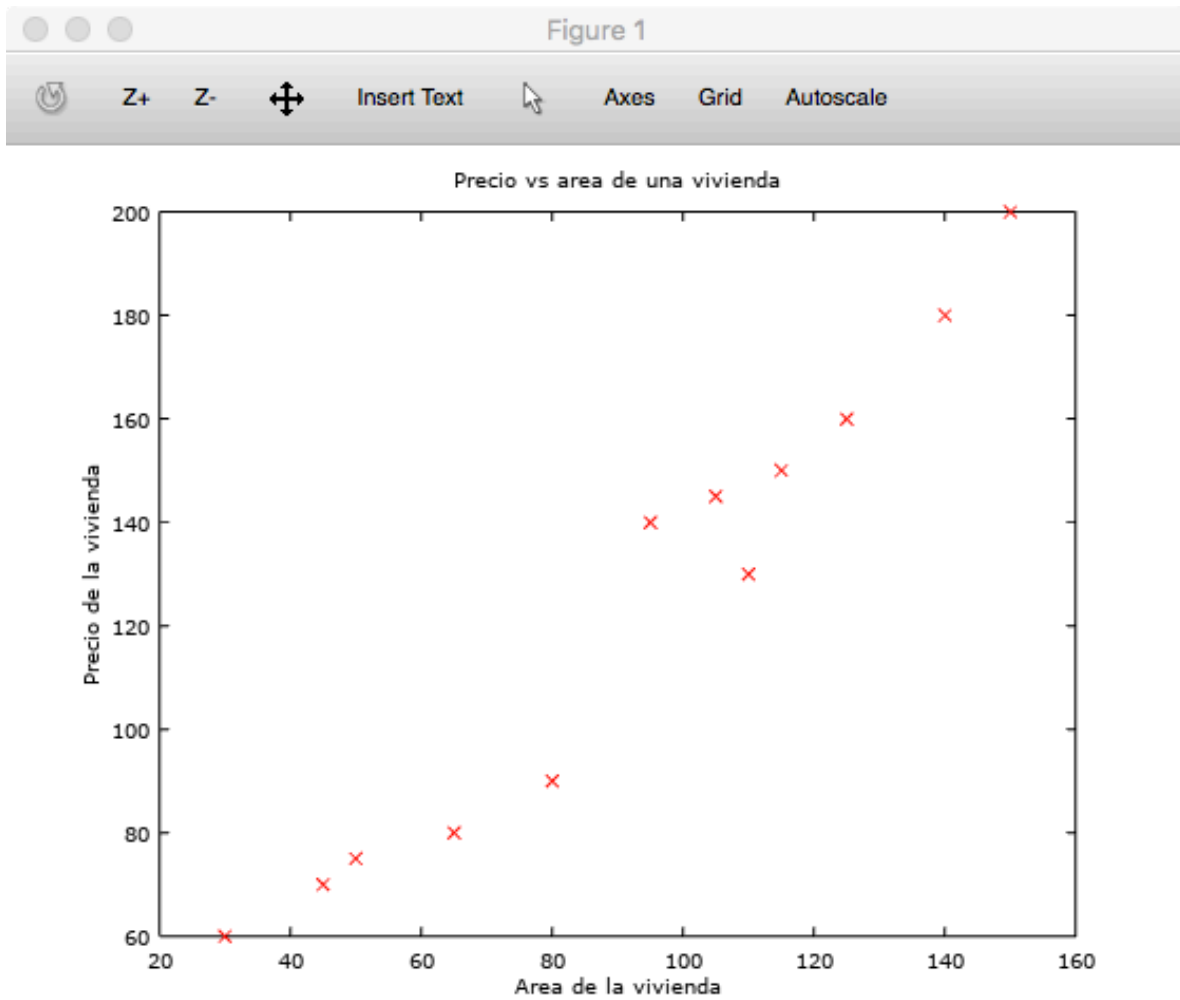
Elabore en la carpeta *lab2* un programa que grafique los siguientes datos

Área de una vivienda	Precio de una vivienda
30	60
45	70
50	75
65	80
80	90
95	140
105	145
110	130
115	150
125	160
140	180
150	200

La gráfica debe incluir el título de la figura y el nombre de los ejes

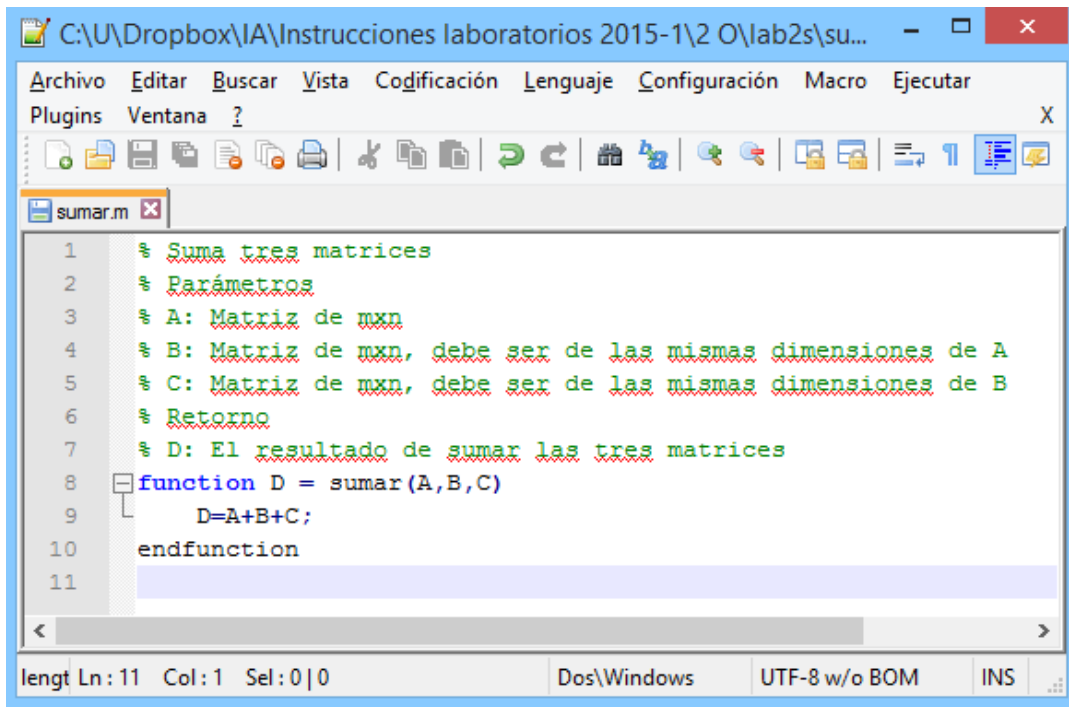
Debe obtener un gráfica como la de la Figura 7

Figura 7. Gráfica de puntos



3.7 Definición de funciones

En el siguiente ejemplo se puede apreciar la sintaxis para la definición de una función. Los comentarios dentro del código brindan explicaciones sobre el ejemplo.



El nombre de una función debe ser igual al nombre del archivo que la contiene sin la extensión *.m*. En la figura , el nombre de la función es *sumar* y el del archivo es *sumar.m*.

A continuación se incluye una prueba de la anterior función, desde la línea de comandos.

```
==> A=[2 3 4; 6 8 3];
==> B=[5 6 7; 2 8 5];
==> C=[8 4 5; 8 4 0];
==> D=sumar(A, B, C)
D =
    15    13    16
    16    20     8
```

3.7.1 Ejercicio 5

Elabore en la carpeta *lab2* una función con las siguientes especificaciones:

Parámetros

A: una primera matriz de $m \times n$

Valor retornado

E : Una matriz calculada mediante los siguientes pasos:

- i) $B = A^T + 5$
- ii) $C = A * B$
- iii) $D = C^2$
- iv) Halle E , una matriz de $m \times n$ donde $E_{ij} = C_{ij}D_{ij}$

A continuación se presenta un ejemplo de una prueba a esta función.

```
>> A=[3 4 5; 6 4 3; 7 9 4; 3 5 2]

A =

     3     4     5
     6     4     3
     7     9     4
     3     5     2

>> E=ejercicio5(A)

E =

    6334460    6655540    10678191    5476779
    7300560    8567748    13453070    6712438
    17727966    20202510    33404586    16764726
    4475454     5017438     8454196     4257176

>>
```


3.7.2 Ejercicio 6

Elabore en la carpeta *lab2* una función con las siguientes especificaciones:

Parámetros

A : una matriz de $m \times n$

B : una matriz de $m \times p$.

Valor retornado

Z : Una matriz calculada mediante los siguientes pasos:

- i. $D = B^T A$
- ii. $E = D D^T$
- iii. $F = B^T B$
- iv. $Z = E - F$

El siguiente es un ejemplo de una prueba a esta función.

```
>> A=[5 2 4; 1 3 2; 2 -1 -2]
```

```
A =
```

```
5 2 4
```

```
1 3 2
```

```
2 -1 -2
```

```
>> B=[3 1; -2 4; -1 6]
```

```
B =
```

```
3 1
```

```
-2 4
```

```
-1 6
```

```
>> z=ejercicio6(A,B)
```

```
z =
```

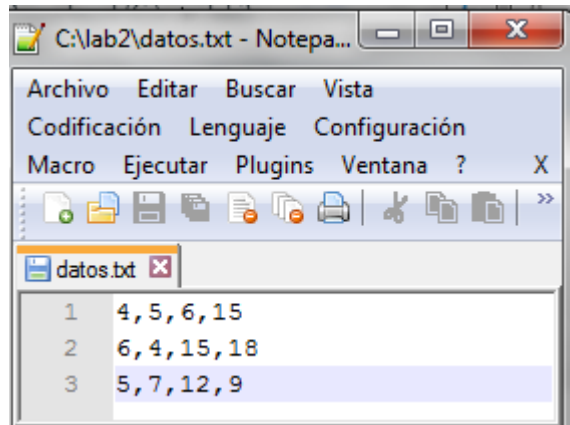
```
208 250
```

```
250 452
```

3.8 Manipulación de matrices

3.8.1 Lectura de una matriz almacenada en un archivo tipo txt

La función *load()* permite inicializar una matriz con valores almacenados en un archivo tipo txt. Para ilustrar el uso de esta función, se ha incluido en la carpeta *lab2* el siguiente archivo:



Para leer el archivo de datos, primero debe utilizar el comando `cd` para que el directorio actual sea el directorio donde se encuentra el archivo de datos:

```
==> cd "c:/lab2"
```

Posteriormente, se puede utilizar la función `load()` para inicializar una matriz con los datos del archivo de texto.

```
==> A=load("datos.txt")
A =
    4    5    6   15
    6    4   15   18
    5    7   12    9
```

3.8.2 Combinación de matrices

Se puede combinar dos matrices en una sola. Para ilustrar esta característica, se utilizarán las siguientes matrices:

```
==> A=[ 2 4; 5 6]
A =
    2    4
    5    6

==> B=[3 7; 9 8]
B =
    3    7
    9    8
```

Una forma de combinar las matrices es colocando una a la derecha de la otra.

```
==> C=[A B]
C =
     2     4     3     7
     5     6     9     8
```

También podemos colocar una matriz debajo de otra:

```
==> D=[A; B]
D =
     2     4
     5     6
     3     7
     9     8
```

3.8.3 Extracción de partes de una matriz a otra matriz

Para ilustrar esta característica, usaremos la siguiente matriz A.

```
==> A=magic(6)
A =
    35     1     6    26    19    24
     3    32     7    21    23    25
    31     9     2    22    27    20
     8    28    33    17    10    15
    30     5    34    12    14    16
     4    36    29    13    18    11
```

Como un primer ejemplo, se muestra en la siguiente figura cómo extraer las posiciones 3 a 5 de las filas 2 a 4.

```
==> B=A(2:4,3:5)
B =
     7    21    23
     2    22    27
    33    17    10
```

En el siguiente ejemplo, se muestra como extraer de la matriz A todas las posiciones de las filas 2 a 4. Los dos puntos en el segundo subíndice se interpretan como *todas las posiciones*.

```
==> B=A(2:4, : )
B =
     3     32     7     21     23     25
    31     9     2     22     27     20
     8     28    33    17    10     15
```

En este otro ejemplo se extrae de la matriz *A* las posiciones 1 a 2, de las filas 4 hasta la fila final. La palabra reservada *end* se interpreta en este ejemplo como el subíndice de la última fila.

```
==> B=A(4:end, 1:2)
B =
     8     28
    30     5
     4    36
```

Y un último ejemplo mostrando cómo se extraen todos los elementos de las filas 2 y 4 de una matriz.

```
==> B=A([2,4], : )
B =
     3     32     7     21     23     25
     8     28    33    17    10     15
```

3.8.4 Conformación de vectores con partes de una matriz

También es posible conformar vectores con partes de una matriz. Para ilustrar esta característica se utilizará la misma matriz *A* del ejemplo anterior:

```
==> A=magic(6)
A =
    35     1     6     26    19     24
     3     32     7     21    23     25
    31     9     2     22    27     20
     8     28    33    17    10     15
    30     5    34    12    14     16
     4    36    29    13    18     11
```

En un primer ejemplo se forma un vector con las posiciones 2 a 5 la fila 3 de la matriz *A*.

```
==> b=A(3, 2:5)
b =
     9     2    22    27
```

En este otro ejemplo se conforma un vector con las posiciones 2 a 5 de la última fila.

```
==> b=A(end, 2:5)
b =
    36    29    13    18
```

En forma similar, se pueden conformar vectores columna con partes de un matriz.

3.9 Funciones con matrices

En los párrafos siguientes se comentarán algunas funciones útiles en la manipulación de matrices.

La función *zeros* permite inicializar una matriz en ceros:

```
==> A=zeros(3,2)
A =
     0     0
     0     0
     0     0
```

En forma similar, la función *ones* () permite inicializar una matriz en unos.

```
==> A=ones(3,2)
A =
     1     1
     1     1
     1     1
```

La función *size* () permite hallar el número de filas y el número de columnas de una matriz, o ambas dimensiones a la vez. Sigue un ejemplo:

```

==> A=[6 5 2; 2 8 6; 9 6 2; 10 6 7]
A =

     6     5     2
     2     8     6
     9     6     2
    10     6     7

==> n=size(A,1)
n = 4

==> n=size(A,2)
n = 3

==> n=size(A)
n =

     4     3

```

La función *length* () permite hallar el número de elementos de un vector fila o de un vector columna. Se incluye un ejemplo de cada caso:

```

==> A=[5 6 7]
A =

     5     6     7

==> n=length(A)
n = 3

==> A=[5; 6; 7]
A =

     5
     6
     7

==> n=length(A)
n = 3

```

La función *sum* () retorna la suma de los elementos de un vector:

```

==> A=[4; 6; 7];
==> s=sum(A)
s = 17

```

3.9.1 Ejercicio 7

Elabore en la carpeta *lab2* una función con las siguientes especificaciones:

Parámetro:

A : Una matriz de $n \times n$, con $n \geq 2$

Valor retornado:

Z : Una matriz hallada mediante los siguientes cálculos:

- i. Inicialice la matriz B con un cuadrado mágico de las mismas dimensiones de A
- ii. $C = A + B$
- iii. Inicialice la matriz D con las dos primeras columnas de la matriz C
- iv. Inicialice la matriz E con las dos últimas filas de la matriz C
- v. $F = D + E^T$
- vi. Inicialice el vector columna G con el resultado de colocar la segunda columna de la matriz F debajo de la primera columna de la matriz F
- vii. $Z = G^T$

El siguiente es un ejemplo de una prueba de esta función:


```

>> A=[1 5 7; 7 4 3; 9 6 3]

A =

     1     5     7
     7     4     3
     9     6     3

>> z=ejercicio7(A)

z =

    19    19    23    19    24    20

>>

```

3.9.2 Ejercicio 8

Elabore en la carpeta *lab2* una función con las siguientes especificaciones:

Parámetros:

A : Una matriz de $m \times n$, con $n \geq 2$

B : Una matriz de $p \times n$, con $n \geq 2$

Valor retornado:

F : Una matriz hallada mediante los siguientes cálculos:

- i. Inicialice C con el resultado de colocar la matriz B debajo de la matriz A
- ii. Inicialice la matriz D con las dos primeras columnas de la matriz C
- iii. $E = D + 1$
- iv. Inicialice la matriz F con el resultado de colocar la matriz E a la derecha de la matriz D

El siguiente es un ejemplo de una prueba de esta función:

```
>> A=[15 18 21; 24 27 30]

A =

    15    18    21
    24    27    30

>> B=[50 52 -1; 59 63 -3; -1 2 -3]

B =

    50    52    -1
    59    63    -3
    -1     2    -3

>> F=ejercicio8(A,B)

F =

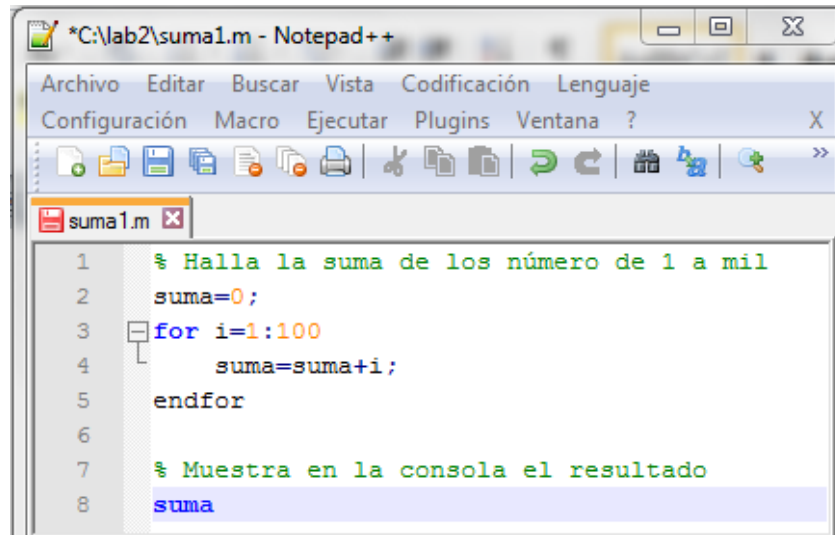
    15    18    16    19
    24    27    25    28
    50    52    51    53
    59    63    60    64
    -1     2     0     3

>>
```

3.10 Sentencias de control

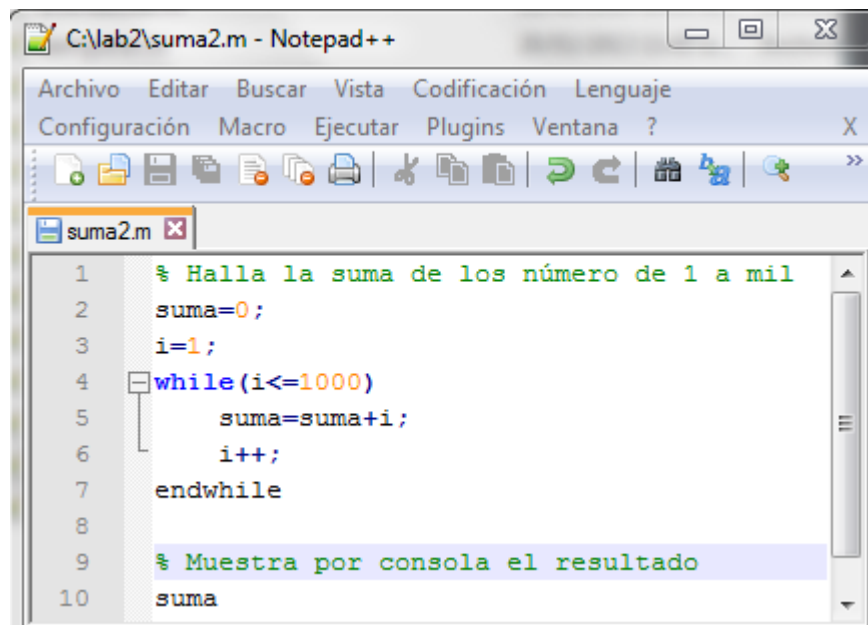
3.10.1 Bucles

Para repetir instrucciones, se cuenta con las sentencias *for ... endfor*, *while ... endwhile*, y *do ... until*. En los siguientes programas se puede observar el uso de cada una de estas sentencias. Se ha incluido la salida correspondiente a cada uno de los programas.



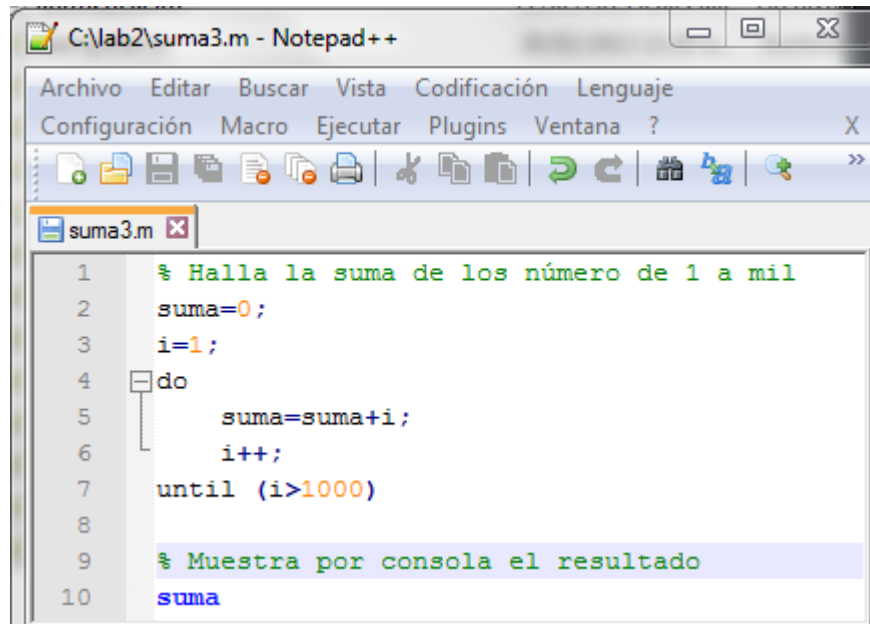
```
*C:\lab2\suma1.m - Notepad++
Archivo  Editar  Buscar  Vista  Codificación  Lenguaje
Configuración  Macro  Ejecutar  Plugins  Ventana  ?
suma1.m
1  % Halla la suma de los número de 1 a mil
2  suma=0;
3  for i=1:100
4      suma=suma+i;
5  endfor
6
7  % Muestra en la consola el resultado
8  suma
```

```
==> suma1
suma = 5050
```



```
C:\lab2\suma2.m - Notepad++
Archivo  Editar  Buscar  Vista  Codificación  Lenguaje
Configuración  Macro  Ejecutar  Plugins  Ventana  ?
suma2.m
1  % Halla la suma de los número de 1 a mil
2  suma=0;
3  i=1;
4  while(i<=1000)
5      suma=suma+i;
6      i++;
7  endwhile
8
9  % Muestra por consola el resultado
10 suma
```

```
==> suma2  
suma = 500500
```



```
==> suma3  
suma = 500500
```

Nota: Los operadores relacionales son los siguientes:

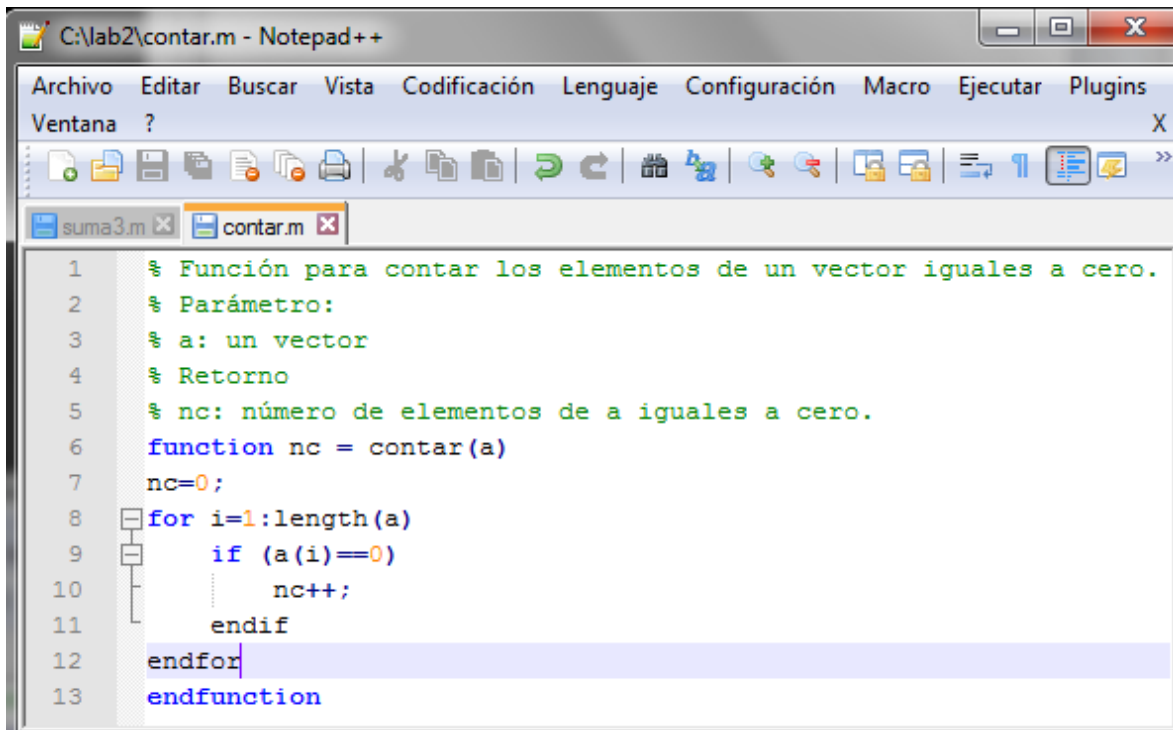
< > == <= >= != ~=

Los dos últimos operadores de la lista representan '*diferente de*'.

Los operadores lógicos son &, |, y !, los cuales corresponden al 'Y lógico', 'O lógico' y 'Negación lógica'

3.10.2 Decisiones

Para hacer decisiones, se dispone en Octave de la sentencia `if ...else ... endif`, cuya sintaxis se puede apreciar en la siguiente imagen. Como es usual, la parte *else* de esta sentencia es opcional.



La siguiente imagen muestra una prueba de la anterior función.

```
==> a=[4 6 0 6 4 8 0 1 0 0];
==> nc=contar(a)
nc = 4
```

3.11 Impresión en consola

3.11.1 Función fprintf ()

La función *fprintf* () permite imprimir en la consola, usando una plantilla. Vemos los siguientes ejemplos:

- i) Imprimir por consola el valor de una variable entera.

```
==> n=12*12;
==> fprintf('El resultado es: %d\n',n)
El resultado es: 144
```

La cadena `"El resultado es: %d\n"` se denomina una plantilla. El sistema reemplaza en esta plantilla los caracteres `%d` por los caracteres `144`, quedando la salida `"El resultado es: 144"` seguido de un salto de línea

ii) Imprimir una variable entera especificando el ancho

```
==> n=12*12;
==> fprintf("El resultado es: %10d\n",n)
El resultado es:      144
```

Al especificar un ancho de 10, aparecen espacios en blanco a la izquierda del número 144.

iii) Imprimir en consola el valor de una variable real.

```
==> a=200/3;
==> fprintf("El resultado es: %f\n",a)
El resultado es: 66.666667
```

El sistema reemplaza en la plantilla los caracteres `%f` por los caracteres `66.666667`, quedando la salida `"El resultado es: 66.666667"`.

iv) También se puede especificar ancho para las variables reales.

```
==> a=200/3;
==> fprintf("El resultado es: %15f\n",a)
El resultado es:      66.666667
```

v) Adicionalmente, se puede especificar para variables reales el número de decimales que se desean en la salida:

```
==> a=200/3;
==> fprintf("El resultado es: %15.4f\n",a)
El resultado es:      66.6667
```

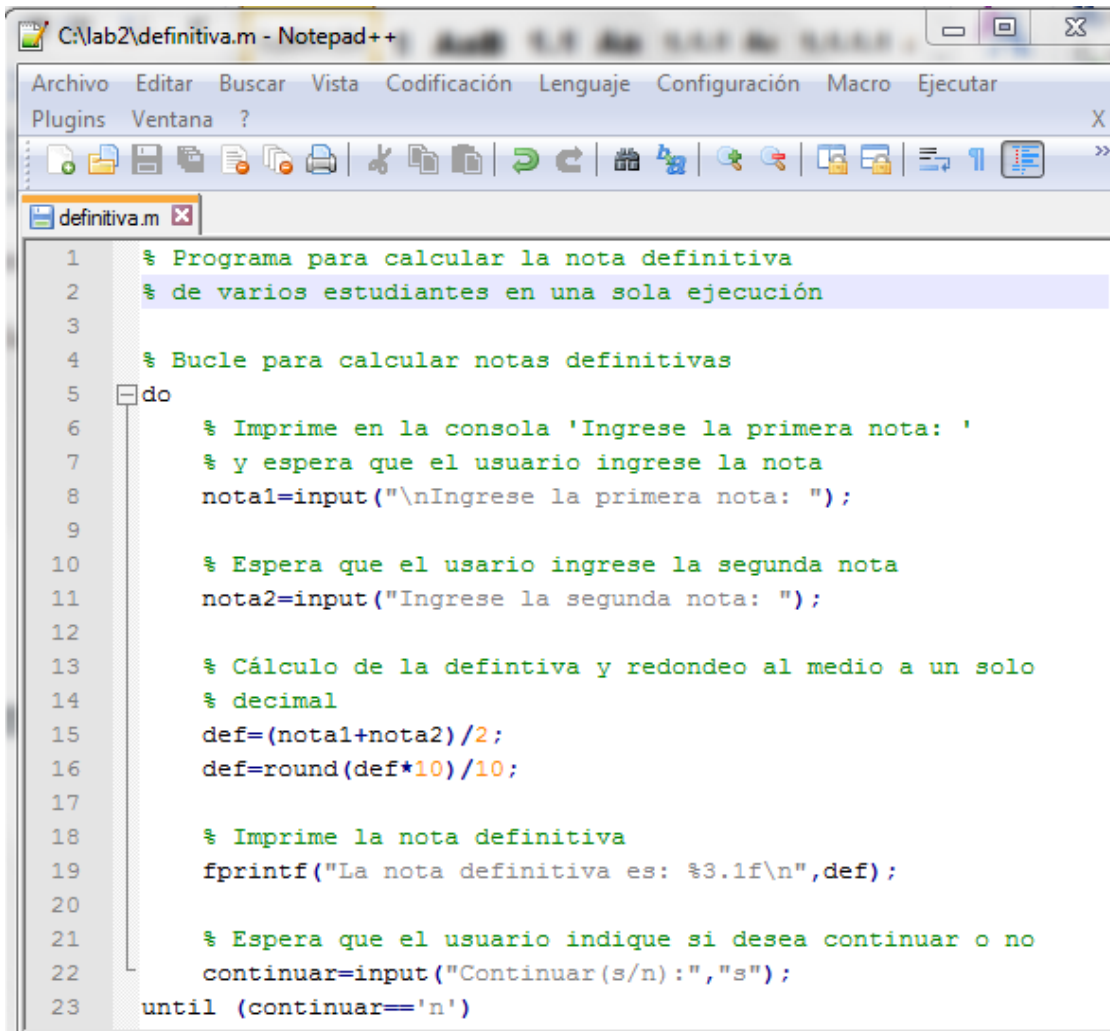
Se puede observar que en la salida solo figuran 4 decimales.

3.11.2 Ejemplo de programa de entrada, cálculo y salida

El siguiente programa ilustra el uso de instrucciones de entrada y de salida por consola.

Como ejemplo, se considerara el problema de calcular la nota definitiva de una asignatura, que es el promedio de las calificaciones de dos parciales.

El programa de la figura permite calcular varias notas definitivas, en una sola ejecución:



```
1 % Programa para calcular la nota definitiva
2 % de varios estudiantes en una sola ejecución
3
4 % Bucle para calcular notas definitivas
5 do
6     % Imprime en la consola 'Ingrese la primera nota: '
7     % y espera que el usuario ingrese la nota
8     nota1=input("\nIngrese la primera nota: ");
9
10    % Espera que el usuario ingrese la segunda nota
11    nota2=input("Ingrese la segunda nota: ");
12
13    % Cálculo de la definitiva y redondeo al medio a un solo
14    % decimal
15    def=(nota1+nota2)/2;
16    def=round(def*10)/10;
17
18    % Imprime la nota definitiva
19    fprintf("La nota definitiva es: %3.1f\n",def);
20
21    % Espera que el usuario indique si desea continuar o no
22    continuar=input("Continuar(s/n):","s");
23 until (continuar=='n')
```

La siguiente es una prueba del anterior programa:

```
==> definitiva

Ingrese la primera nota: 3
Ingrese la segunda nota: 5
La nota definitiva es: 4.0
Continuar(s/n):s

Ingrese la primera nota: 3
Ingrese la segunda nota: 2.9
La nota definitiva es: 3.0
Continuar(s/n):n
```

3.12 Programa para probar programas

3.12.1 Ejercicio 9

Entre los archivos suministrados para la realización del laboratorio, en la carpeta *lab2*, se encuentra el programa *ejercicio9.m*, que prueba todos los programas y las funciones solicitados en las anteriores secciones. En el caso de las funciones, el programa *ejercicio9.m* les suministra argumentos adecuados.

Todo lo que se requiere hacer en este punto es cambiar en *ejercicio9.m* los nombres de los archivos por los nombres que fueron utilizados por ustedes en el desarrollo del laboratorio.

Al ejecutar *ejercicio9.m*, debe poderse observar la salida, una después de otra, de los ocho ejercicios solicitados, ya sea texto en la consola o gráficas en nuevas ventanas.

Tenga en cuenta que las funciones (ejercicios 5, 6, 7 y 8) se prueban con parámetros diferentes a los usados anteriormente, y así mismo arrojarán resultados diferentes.

Los restantes ejercicios, por ser programas y no tener parámetros, producirán siempre la misma salida.

4 Diligenciamiento de la plantilla

Deben diligenciar la información solicitada en la plantilla, en formato *doc*, que se incluye entre los archivos del laboratorio, carpeta *lab2*.

Entre lo solicitado por la plantilla, se encuentra el tiempo promedio invertido por estudiante para realizar este laboratorio. Por ejemplo, si el grupo consta de tres estudiantes, dos de los cuales invirtieron en elaborar el laboratorio 4 horas cada uno, y el otro, 2 horas, el tiempo promedio es $\frac{4+4+2}{3} = 3.33h = 3h\ 20m$.

5 Forma de entrega

El procedimiento de entrega consiste en:

- Comprimir en un solo archivo de extensión *rar* o *zip* la carpeta *lab2*, que debe incluir la plantilla diligenciada y el código solicitado.
- Subir mediante el correspondiente enlace el archivo comprimido a la plataforma Moodle. Solo es necesario que uno de los integrantes del grupo suba el archivo, ya que en la plantilla se encontrarán incluidos los nombres de todos los integrantes.

La plantilla debe estar en formato Word para poderle hacer comentarios.

Para informes entregados con posterioridad a la fecha-hora de plazo establecida habrá las siguientes penalizaciones:

- Hasta 24 horas de retraso, 1 unidad
- De 24 a 48 horas de retraso, 2 unidades
- De 48 a 72 horas de retraso, 3 unidades
- De 72 a 96 horas de retraso, 4 unidades
- De 96 horas de retraso en adelante, 5 unidades

[Revisado LHR 2018-02-14]