

Tarea 07: Capitulo 1 - Octave

Alejandra Magalí López Miranda, 201600085^{1,*}

¹*Escuela de Ingeniería Mecánica Eléctrica, Facultad de Ingeniería, Universidad de San Carlos, Guatemala.*
(Dated: 15 de septiembre de 2025)

I. INTRODUCCIÓN

Esta tarea en Octave permite explorar y aplicar conceptos fundamentales de programación científica. Incluye el uso de números, matrices, cadenas, estructuras, operadores, funciones, gráficos y conexión a bases de datos, así como estructuras de control de flujo como if, for, while y try-catch. El objetivo es familiarizarse con la sintaxis de Octave y aprender a manipular datos, realizar cálculos y generar visualizaciones.

II. OBJETIVOS

A. General

- Familiarizarse con Octave y desarrollar habilidades para manipular datos numéricos y estructuras, crear funciones, generar gráficos y conectar con bases de datos, aplicando la programación científica de manera eficiente.

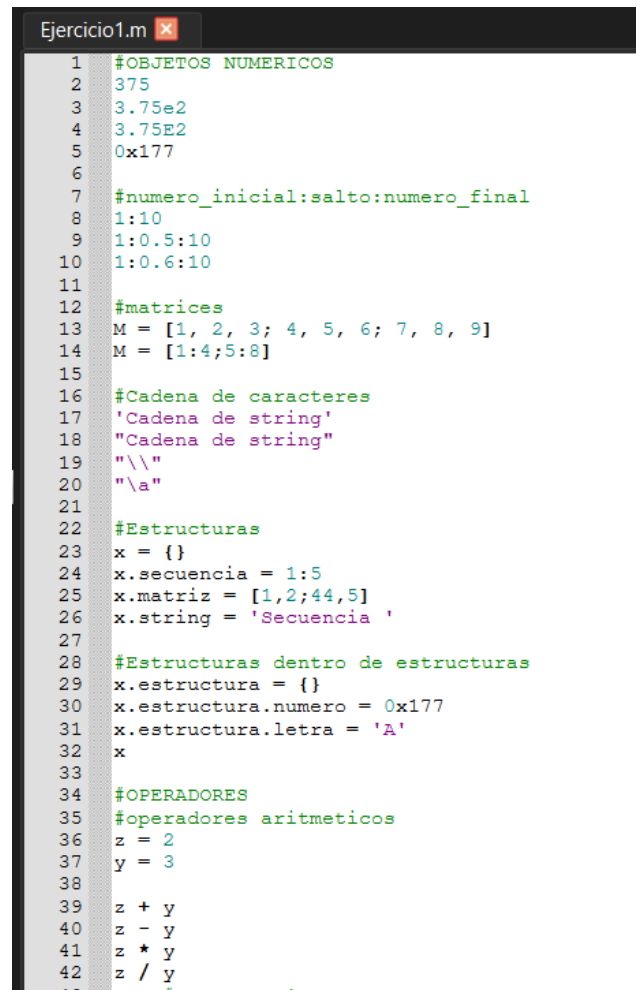
B. Específicos

- * Aplicar operaciones aritméticas, comparaciones y lógicas sobre números, matrices y estructuras.
- * Implementar estructuras de control de flujo (if, for, while, try-catch) para automatizar procesos y tomar decisiones condicionales.
- * Generar funciones personalizadas y representaciones gráficas de datos, incluyendo múltiples funciones y personalización de estilo.

III. RESULTADOS

A. Código en Octave

Figura 1



```
Ejercicio1.m x
1  #OBJETOS NUMERICOS
2  375
3  3.75e2
4  3.75E2
5  0x177
6
7  #numero_inicial:salto:numero_final
8  1:10
9  1:0.5:10
10 1:0.6:10
11
12 #matrices
13 M = [1, 2, 3; 4, 5, 6; 7, 8, 9]
14 M = [1:4;5:8]
15
16 #Cadena de caracteres
17 'Cadena de string'
18 "Cadena de string"
19 "\\\"
20 "\a"
21
22 #Estructuras
23 x = {}
24 x.secuencia = 1:5
25 x.matriz = [1,2;44,5]
26 x.string = 'Secuencia '
27
28 #Estructuras dentro de estructuras
29 x.estructura = {}
30 x.estructura.numero = 0x177
31 x.estructura.letra = 'A'
32 x
33
34 #OPERADORES
35 #operadores aritmeticos
36 z = 2
37 y = 3
38
39 z + y
40 z - y
41 z * y
42 z / y
43
```

* 2816733570108@ingenieria.usac.edu.gt

Figura 2

```
Ejercicio2.m
1 #Estructuras de control de flujo
2 x = 1
3 y = 0
4
5 if(x > y)
6     'X es mayor a Y'
7 elseif (x == y)
8     'X y Y son iguales'
9 else
10    'Y es mayor a X'
11 endif
12
13 #ejemplo 2
14 x = 1
15 y = 0
16 z = -5
17
18 if(x > y & z < 0)
19     'X es mayor a Y y Z es menor a 0'
20 elseif (x == y | z < 0)
21     'X y Y son iguales'
22 else
23     'Y es mayor a X'
24 endif
25
26 #Estructuras de control while
27
28 x = 1
29 y = 0
30 z = -5
31
32 while(z < y )
33     'Valor de z'
34     z
35     ++z
36 endwhile
37
38 #Estructuras de control for
39 fib = ones(1, 10);
40
41 for i = 3:10
42     fib(i) = fib(i-1) + fib(i-2);
43 end
```

Figura 3

```
Ejercicio3.m hipotenusa.m
1 %% Copyright (C) 2025 Alejandra
2 %%
3 %% This program is free software: you can redistribute it and/or modify
4 %% it under the terms of the GNU General Public License as published by
5 %% the Free Software Foundation, either version 3 of the License, or
6 %% (at your option) any later version.
7 %%
8 %% This program is distributed in the hope that it will be useful,
9 %% but WITHOUT ANY WARRANTY; without even the implied warranty of
10 %% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
11 %% GNU General Public License for more details.
12 %%
13 %% You should have received a copy of the GNU General Public License
14 %% along with this program. If not, see <https://www.gnu.org/licenses/>.
15 %%
16 %% -- texinfo --
17 %% @deftypefn {} (@var{retval}) hipotenusa (@var{input1}, @var{input2})
18 %%
19 %% @seealso{}
20 %% @end deftypefn
21 %%
22 %% Author: Alejandra <Alejandra@LAPTOP-HLVV3OUK>
23 %% Created: 2025-09-14
24
25 function hipo = hipotenusa (a, b)
26     hipo = sqrt(a.^2 + b.^2);
27 endfunction
```

Figura 4

```
Ejercicio4.m
1 pkg load database
2 conn = pg_connect(setdbopts('dbname','postgres','host','localhost','port','54
3 N=pg_exec_params(conn, "INSERT INTO test VALUES ('12', 'Texto Prueba', 1005 )
4 N=pg_exec_params(conn, "SELECT * FROM test;");
5 disp(N)
6 N.data
```

B. Objetos numericos y operadores

Figura 5

```
Command Window
>> Ejercicio1

ans = 375
ans = 375
ans = 375
ans = 375
>>
```

Figura 6

ans =	1	2	3	4	5	6	7	8	9	10
ans =	Columns 1 through 8:									
	1.0000	1.5000	2.0000	2.5000	3.0000	3.5000	4.0000	4.5000		
	Columns 9 through 16:									
	5.0000	5.5000	6.0000	6.5000	7.0000	7.5000	8.0000	8.5000		
	Columns 17 through 19:									
	9.0000	9.5000	10.0000							
ans =	Columns 1 through 8:									
	1.0000	1.6000	2.2000	2.8000	3.4000	4.0000	4.6000	5.2000		
	Columns 9 through 16:									
	5.8000	6.4000	7.0000	7.6000	8.2000	8.8000	9.4000	10.0000		

Figura 7

```

M =

    1     2     3
    4     5     6
    7     8     9

M =

    1     2     3     4
    5     6     7     8

```

Figura 8

```

ans = Cadena de string
ans = Cadena de string
ans = \
ans =

```

Figura 9

```

scalar structure containing the fields:

    secuencia =

         1     2     3     4     5

    matriz =

         1     2
        44     5

    string = Secuencia
    estructura =

    scalar structure containing the fields:

        numero = 375
        letra = A

x =

    scalar structure containing the fields:

        secuencia =

             1     2     3     4     5

        matriz =

             1     2
            44     5

        string = Secuencia
        estructura =

        scalar structure containing the fields:

            numero = 375
            letra = A

```

Figura 10

```

z = 2
y = 3
ans = 5
ans = -1
ans = 6
ans = 0.6667
ans = 3
ans = 2
a = 2
b = 3
ans = 1
ans = 1
ans = 0
ans = 0
ans = 0
ans = 1
j = 1
k = 0
ans = 0
ans = 1
ans = 0

```

C. Estructuras de control de flujo, Matrices

Figura 11

```
Command Window
>> Ejercicio2

x = 1
y = 0
ans = X es mayor a Y
>> |
```

Figura 12

```
x = 1
y = 0
z = -5
ans = X es mayor a Y y Z es menor a 0
>> |
```

Figura 13

```
>> Ejercicio2

x = 1
y = 0
z = -5
ans = Valor de z
z = -5
ans = -4
ans = Valor de z
z = -4
ans = -3
ans = Valor de z
z = -3
ans = -2
ans = Valor de z
z = -2
ans = -1
ans = Valor de z
z = -1
ans = 0
>>
```

Figura 14

```
fib =
    1     1     2     3     5     8    13    21    34    55
>>
```

Figura 15

```
ans = NO SE PUDO EJERCUTAR, se continua con la ejecucion normal del codigo
```

Figura 16

```
0.1000 0.2000 0.3000
N =
     0     1     2
     8    10    12
    375    374    293
ans =
     1.0000     3.0000     9.0000
    12.0000    15.0000    23.0000
   375.1000   374.2000   293.3000
ans =
     1.0000     1.0000     5.0000
    -4.0000    -5.0000    -1.0000
   -374.9000  -373.8000  -292.7000
ans =
    2641.000    2639.000    2077.000
    4165.000    4168.000    3291.000
    114.100     114.300     90.500
ans =
    1.4992e+03    1.8680e+03    3.2194e+03
   -3.7500e+02   -7.4780e+02   -2.0504e+03
    8.0000e+00    1.5000e+01    6.2000e+01
ans =
     69.500    126.800    233.900
ans =
     1.0000     4.0000     0.1000
     2.0000     5.0000     0.2000
     7.0000    11.0000     0.3000
```

D. Funciones en Octave y Graficas

Figura 17

```

Command Window

>> Ejercicio3

x = 2.2361
>> Ejercicio3

x = 2.2361
b = 1
>> |

```

Figura 18

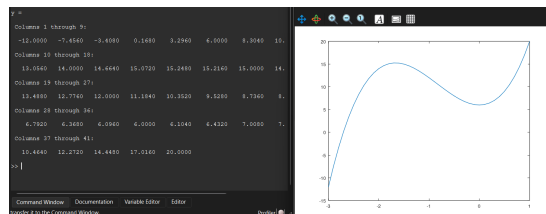


Figura 19

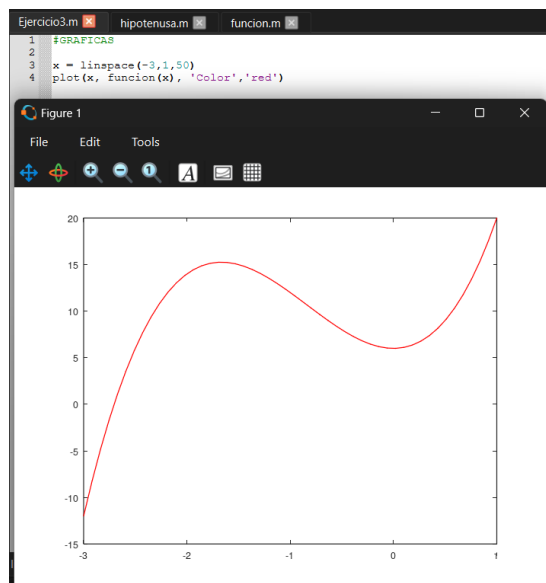


Figura 20

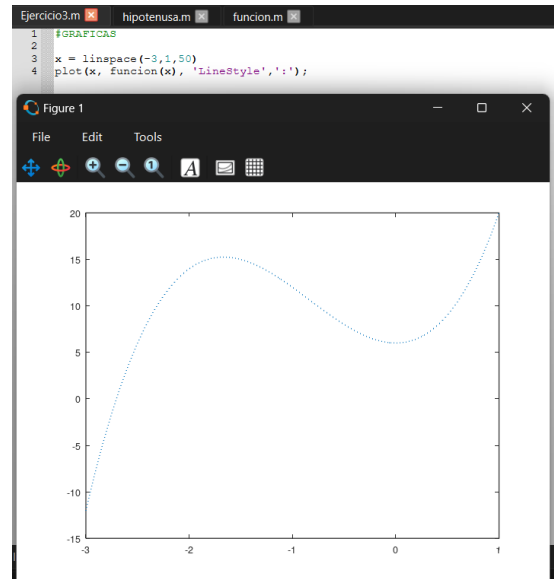


Figura 21

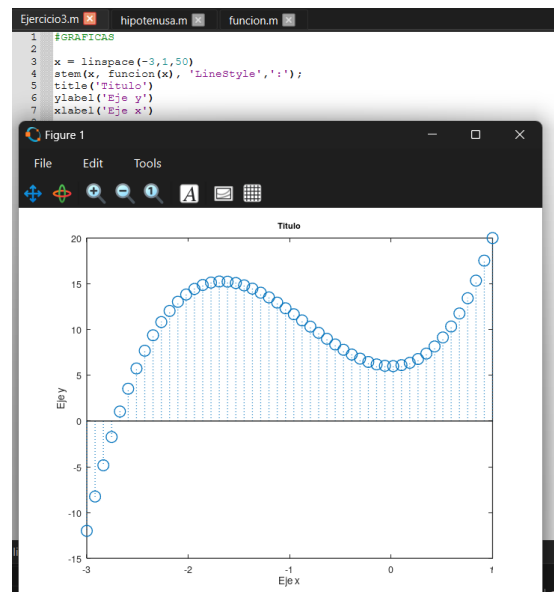
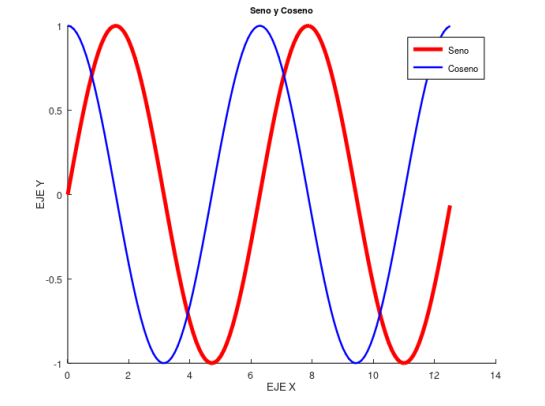


Figura 22



IV. CONEXION ENTRE OCTAVE Y POSTGRESQL

Figura 23

```
>> Ejercicio4
N = 1
data =
{
  [1,1] = 12
  [1,2] = Texto Prueba
  [1,3] = 1005
}

columns =
{
  [1,1] = num
  [1,2] = text
  [1,3] = num2
}

types =

1x3 struct array containing the fields:

    name
    is_array
    is_composite
    is_enum
    elements
>>
```

Figura 24

Data Output				Messages	Notifications
	num integer	text character varying (50)	num2 integer	SQL	
1	12	Texto Prueba	1005		

Figura 25

```
>> N.data
ans =
{
  [1,1] = 12
  [1,2] = Texto Prueba
  [1,3] = 1005
}
```

V. CONCLUSIONES

- * Octave permite realizar operaciones matemáticas y algebraicas de manera rápida y eficiente.
- * Las estructuras de control y funciones permiten automatizar tareas complejas, mejorar la legibilidad del código y reutilizar cálculos.
- * La conexión con bases de datos y la capacidad de graficar datos permiten que los programas sean útiles para aplicaciones reales en ingeniería, ciencia y análisis de datos.

VI. ANEXOS

Link de repositorio: <https://github.com/AlejandraMLM/980-Proyectos.git>