

Universidad Autónoma de Querétaro

Facultad de Informática

Algoritmos Avanzados

Dra. Diana Córdova Esparza

Práctica #1

Vectores y MATLAB

Saúl Axel López Gómez

Agosto 2, 2025

Índice

1. Introducción	3
2. Fundamentos teóricos	3
3. Material	3
4. Metodología	4
4.1. Ejercicio 1.	4
4.2. Ejercicio 2.	5
4.3. Ejercicio 3.	5
4.4. Ejercicio 4.	5
4.5. Ejercicio 5.	5
4.6. Ejercicio 6.	6
4.7. Ejercicio 7.	6
4.8. Ejercicio 8.	6
5. Resultados	7
5.1. Ejercicio 1.	7
5.2. Ejercicio 2.	10
5.3. Ejercicio 3.	10
5.4. Ejercicio 4.	11
5.5. Ejercicio 5.	11
5.6. Ejercicio 6.	12
5.7. Ejercicio 7.	12
5.8. Ejercicio 8.	13
6. Conclusiones	14
Referencias	15

Índice de figuras

1.	Comando para un vector columna	7
2.	Comando para un vector fila	7
3.	Comando para un vector columna de números impares	8
4.	Comando para un vector columna de números impares	8
5.	Comando para un vector fila de números pares	9
6.	Comando para un vector fila de números pares	10
7.	Comando para la raíz cuadrada de todos los elementos de un vector	10
8.	Comando para elevar al 2/3 los elementos del vector	11
9.	Comando para obtener el coseno de un vector	11
10.	Comando para obtener el arcoseno de un vector	12
11.	Comando para conocer el valor de una posición en específico	12
12.	Comando para conocer el valor de una posición en específico	13

Índice de Códigos

1.	Ejercicio 1: 1) Vector columna	4
2.	Ejercicio 1: 2) Vector fila	4
3.	Ejercicio 1: 3) Vector columna de números impares	4
4.	Ejercicio 1: 4) Vector columna de números impares	4
5.	Ejercicio 2: Elementos del vector al cuadrado con el operador	5
6.	Ejercicio 3: Raíz cuadrada de los elementos del vector con el operador	5
7.	Ejercicio 4: Elementos del vector al 2/3 con el operador	5
8.	Ejercicio 5: Coseno de los elementos del vector	5
9.	Ejercicio 6: ArcoSeno de los elementos del vector	6
10.	Ejercicio 7: linspace de un vector	6
11.	Ejercicio 8: Elemento de un vector	6

1. Introducción

En el área de las ciencias computacionales, el mayor objetivo es el formular algoritmos que puedan servir para solucionar problemas con una solida base en la teoría fundamental, por lo cual, se pretende, tener las bases de un lenguaje de programación de bajo nivel, para poder interpretar de manera fácil y eficaz las matemáticas con el ámbito computacional, sin entrar en detalles a cosas mas técnicas como el uso de memoria, tipos de datos, optimizaciones, etc. Por lo tanto, en este trabajo se busca poner en practica los conceptos y operaciones entre vectores de 'n' dimensiones. Es fundamental la practica de estos conceptos base para poder hacer uno de operaciones matemáticas mas complejas.

2. Fundamentos teóricos

Para representar los fenómenos en una forma matemática, no basta con solo decir cantidades, es requido ser mas puntual, especificar de mejor manera, como ejemplo básico de la vida cotidiana, se encuentra un auto conduciendo por la ciudad, no basta con solo saber la rapidez con la que se desplaza, es necesario conocer hacia que dirección esta yendo; matemáticamente se le conoce a esto como magnitud y dirección, que interpretándolo, es el significado de un vector.

Para la correcta interpretación del carro moviéndose por la ciudad, es necesario conocer todas las variables que están influyendo, si se piensa en un plano cartesiano, se puede obtener las coordenadas del carro cuando este se encuentre viajando, la representación matemática según (Joag, 2016) de un vector contiene la forma:

$$\vec{V} = i\hat{x} + j\hat{y} + k\hat{z} \quad (1)$$

Donde nos dice las direcciones a varios ejes: (x, y, z).

Esta forma vectorial puede ser representada en forma computacional para hacer cálculos complejos o difíciles para el humano, como herramienta auxiliar, se usa los lenguajes de programación, los cuales traducen los conceptos matemáticos entendibles por el humano a la representación de 1 y 0, donde la maquina es capaz de entender y realizar cálculos, donde (Sebesta, 2012) esta de acuerdo.

Existen lenguajes de programación robustos los cuales son perfectos para aplicaciones donde se requiera mucho control sobre lo que pasa al rededor y donde las características del computador son limitadas. Sin embargo, en el caso de ámbito matemático, se busca que la programación sea lo mas amigable y parecida a resolver un problema a mano, por lo tanto (Linge y Langtangen, 2016) sugiere el uso de MATLAB, donde la curva de aprendizaje es pequeña, pero da bases solidas para la migración a lenguajes robustos como C++, C, Fortran, etc.

3. Material

- Computadora con sistema operativo Windows 11
- Documentación oficial de MATLAB/Octave
- Documentación proporcionada por la Dra. Diana
- Lenguaje de programación GNU Octave

4. Metodología

En cada ejercicio siguiente, se pretende poner en practica los conocimientos adquiridos en la primer sesión de introducción al lenguaje de programación Octave, en cada ejercicio se propone una posible solución o comando, donde en la sección de resultados (5) se verifica la veracidad de cada comando propuesto en esta sección.

4.1. Ejercicio 1.

Escribe el comando MATLAB que genera cada uno de los siguientes vectores.

1. $\begin{pmatrix} 1 \\ 2 \\ -3 \end{pmatrix}$

Dado que es una matriz 3x1, donde contiene 3 renglones, se propone un vector columna, donde se usara el simbolo ";" para separar cada elemento como:

```
1 Eje1_1 = [1; 2; -3]
2
```

Código 1: Ejercicio 1: 1) Vector columna

2. $(1, 2, -1, 3)$

En este caso, es una matriz 1x4, donde contiene 4 renglones, se propone un vector fila como:

```
1 Eje1_2 = [1, 2, -1, 3]
2
```

Código 2: Ejercicio 1: 2) Vector fila

3. Un vector columna que contenga los números impares entre 1 y 1000.

Dado que se requieren los números impares entre [1, 1000], solo se propone un vector fila con el rango establecido y en pasos de 2 como:

```
1 Eje1_3 = [1:2:1000]
2
```

Código 3: Ejercicio 1: 3) Vector columna de números impares

4. Un vector fila que contenga los números pares entre 2 y 1000.

Dado que se requieren los números pares entre [2, 1000], se propone un vector fila con sus transpuesta para asi obtener un vector fila:

```
1 Eje1_4 = [2:2:1000] '
2
```

Código 4: Ejercicio 1: 4) Vector columna de números impares

4.2. Ejercicio 2.

Si $x=0:2:20$, escribe el comando MATLAB que eleva al cuadrado cada componente de x

Como ya esta definido $x=0:2:20$, se puede manipular la variable de x , donde se toma cada elemento para elevarlo al cuadrado con el operador `.'^`, con el siguiente comando se debe de poder elevar al cuadrado cada elemento de x

```
1 Eje2 = x.^2
2
```

Código 5: Ejercicio 2: Elementos del vector al cuadrado con el operador `.`

4.3. Ejercicio 3.

Si $x=[0,1,4,9,16,25]$, escribe el comando MATLAB que calcula la raíz cuadrada de cada componente de x

Como ya esta definido el vector x , se ocupa el mismo operador `.'^`, con el siguiente comando se debe de poder obtener la raíz cuadrada de cada elemento de x

```
1 Eje3 = x.^(1/2)
2
```

Código 6: Ejercicio 3: Raiz cuadrada de los elementos del vector con el operador `.`

4.4. Ejercicio 4.

Si $x=0:.1:1$, escribe el comando de MATLAB que eleva cada componente de x a $2/3$

Como ya esta definido el vector x , se ocupa el mismo operador `.'^`, con el siguiente comando se debe de poder elevar a $2/3$ de cada elemento de x

```
1 Eje4 = x.^(2/3)
2
```

Código 7: Ejercicio 4: Elementos del vector al $2/3$ con el operador `.`

4.5. Ejercicio 5.

Si $x=0:\pi/2:2\pi$, escribe el comando MATLAB que calcula el coseno de cada componente de x

Como ya esta definido el vector x , no es necesario usar el operador `.'^` dado que es mayormente usado para operaciones algebraicas en vectores. Como el coseno es una función, en octave esta diseñado para calcular el coseno de cada elemento sin usar el operador, con el siguiente comando se debe de poder calcular el coseno de cada elemento de x

```
1 Eje5 = cos(x)
2
```

Código 8: Ejercicio 5: Coseno de los elementos del vector

4.6. Ejercicio 6.

Si $x = -1:1:1$, escribe el comando MATLAB que calcula el arcoseno de cada componente de x

Como ya esta definido el vector x , con el siguiente comando se debe de poder calcular el arcoseno de cada elemento de x

```
1 Eje6 = asin(x)
2
```

Código 9: Ejercicio 6: ArcoSeno de los elementos del vector

4.7. Ejercicio 7.

Si $x = \text{linspace}(0, 2\pi, 1000)$, ¿cuál es la entrada 50 de x ? ¿Cuál es la longitud de x ?

La funcion **linspace** ayuda a partir n veces un rango fijo. Dado que linspace toma como argumentos el inicio, final, numero de puntos, entonces la longitud de x sera de 1000 elementos. Con la siguiente linea de comando se puede acceder a un elemento especifico del vector x

```
1 Eje7 = x(50)
2
```

Código 10: Ejercicio 7: linspace de un vector

4.8. Ejercicio 8.

Si $k = 0:100$, ¿cuál es la entrada número 12 de $y = 0.5.^k$?

Con la siguiente linea de comando se puede acceder al elemento 12 de y

```
1 Eje8 = y(12)
2
```

Código 11: Ejercicio 8: Elemento de un vector

5. Resultados

A partir de cada ejercicio se probaran todos los comandos propuestos en la sección de metodología (4), con su respectivo análisis y captura de pantalla

5.1. Ejercicio 1.

1. $\begin{pmatrix} 1 \\ 2 \\ -3 \end{pmatrix}$

Escribiendo el comando propuesto en Octave, se obtiene un nuevo vector columna Eje1_1, en la imagen 1, del lado izquierdo se ve en el editor de variables que efectivamente cada elemento del vector se encuentra en su respectiva posición mientras que del lado derecho se observa la línea de comando donde se escribió el comando.

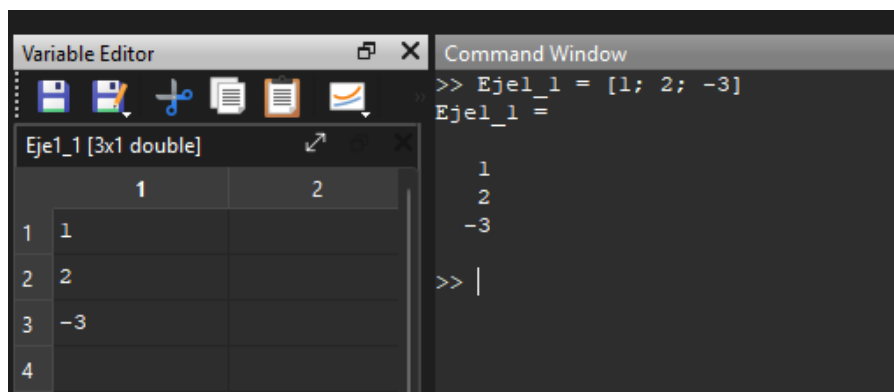


Figura 1: Comando para un vector columna

2. $(1, 2, -1, 3)$

El comando propuesto obtuvo una respuesta positiva, igual a la esperada, ya que en la imagen 2, del lado izquierdo se encuentra el editor de variables, el cual cada valor corresponde a cada elemento del vector, mientras que del lado derecho se encuentra la línea de comandos, el cual muestra la misma salida del vector esperado

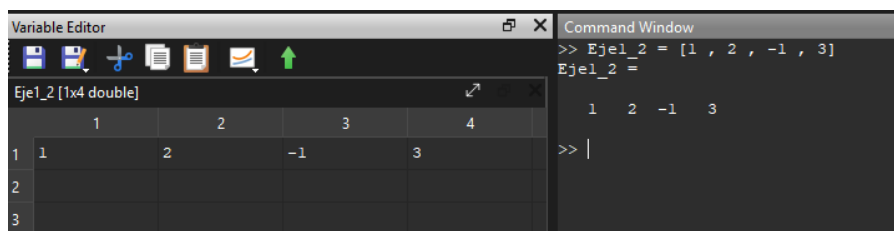


Figura 2: Comando para un vector fila

3. Un vector columna que contenga los números impares entre 1 y 1000.

El comando propuesto en la sección 4 de este ejercicio, sirve para obtener números impares, dado que este vector es largo, visualmente no se puede apreciar los primeros elementos del vector

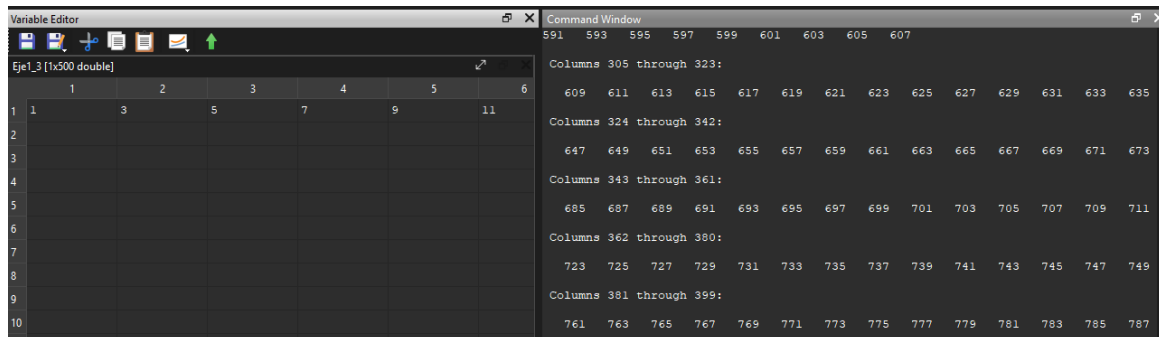


Figura 3: Comando para un vector columna de números impares

En este lenguaje no cuenta con un método parecido a `.head()` como en python para desplegar visualmente los primeros dígitos, por lo tanto, se procede a escribir el siguiente comando para desplegar los primeros 20 dígitos del vector columna de 1000 elementos.

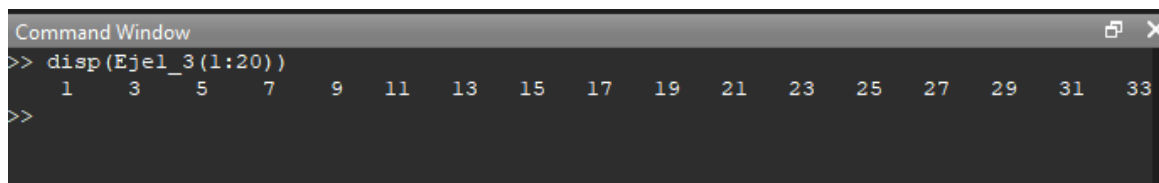
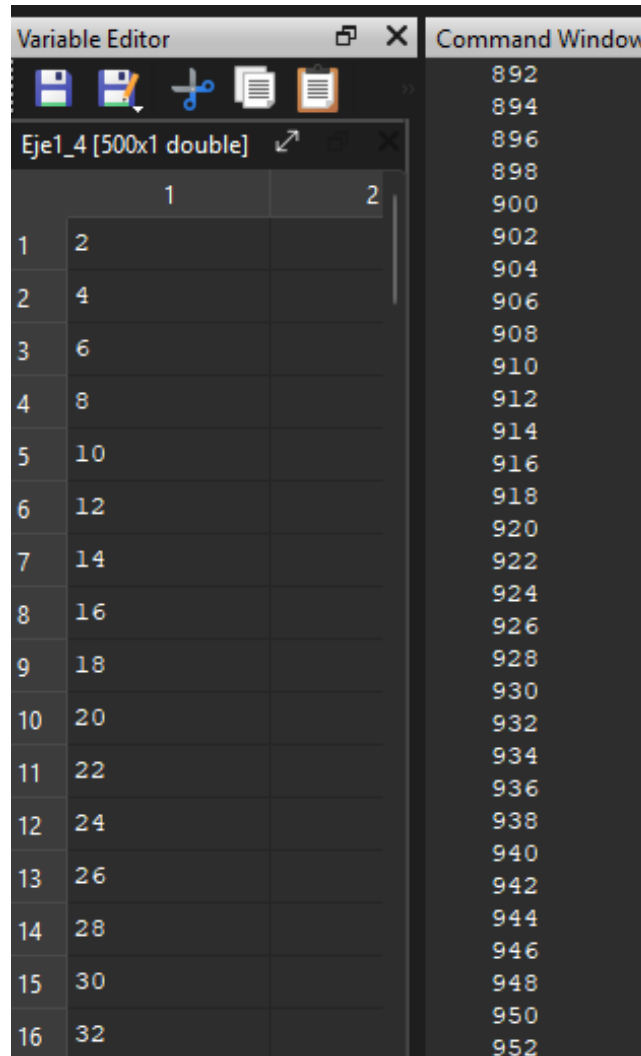


Figura 4: Comando para un vector columna de números impares

4. Un vector fila que contenga los números pares entre 2 y 1000.

El comando de la sección 4 funciona para obtener números impares en un vector fila, si se quiere visualizar nuevamente los primeros dígitos, se puede escribir el mismo comando del ejercicio anterior.



The image shows a MATLAB interface with two windows. The 'Variable Editor' window on the left displays a variable named 'Eje1_4' of type 'double' with dimensions '500x1'. It shows a list of 16 rows, each with a single column of even numbers from 2 to 32. The 'Command Window' on the right shows a list of 16 even numbers from 892 to 952, which are the first 16 elements of the vector.

	1	2
1	2	
2	4	
3	6	
4	8	
5	10	
6	12	
7	14	
8	16	
9	18	
10	20	
11	22	
12	24	
13	26	
14	28	
15	30	
16	32	

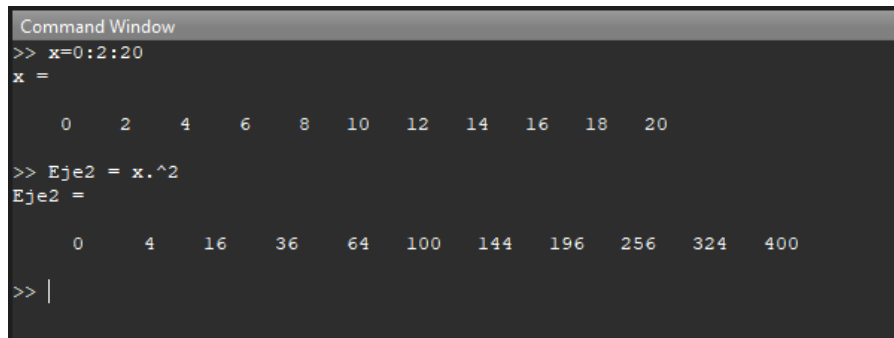
892
894
896
898
900
902
904
906
908
910
912
914
916
918
920
922
924
926
928
930
932
934
936
938
940
942
944
946
948
950
952

Figura 5: Comando para un vector fila de números pares

5.2. Ejercicio 2.

Si $x=0:2:20$, escribe el comando MATLAB que eleva al cuadrado cada componente de x

El nuevo vector generado en la variable **Eje2**, contiene todos los elementos del vector x elevados al cuadrado, esto para no afectar al vector original.



```
Command Window
>> x=0:2:20
x =
    0    2    4    6    8   10   12   14   16   18   20

>> Eje2 = x.^2
Eje2 =
    0    4   16   36   64   100   144   196   256   324   400

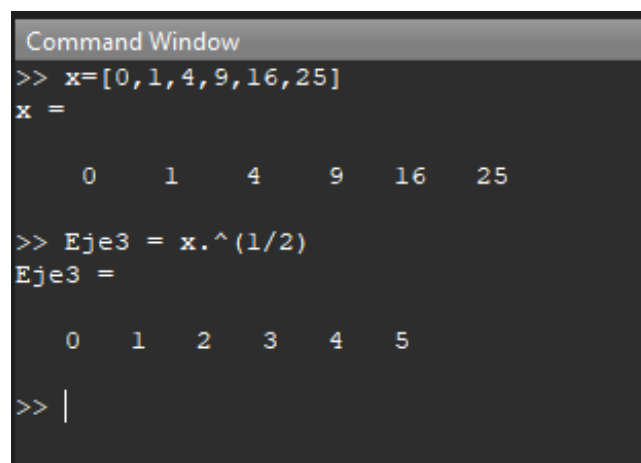
>> |
```

Figura 6: Comando para un vector fila de números pares

5.3. Ejercicio 3.

Si $x=[0,1,4,9,16,25]$, escribe el comando MATLAB que calcula la raíz cuadrada de cada componente de x

Dado que matemáticamente la raíz cuadrada se puede expresar en su forma de fracción, es posible elevar cada elemento del vector x al $(1/2)$ en un nuevo vector Eje3.



```
Command Window
>> x=[0,1,4,9,16,25]
x =
    0    1    4    9   16   25

>> Eje3 = x.^(1/2)
Eje3 =
    0    1    2    3    4    5

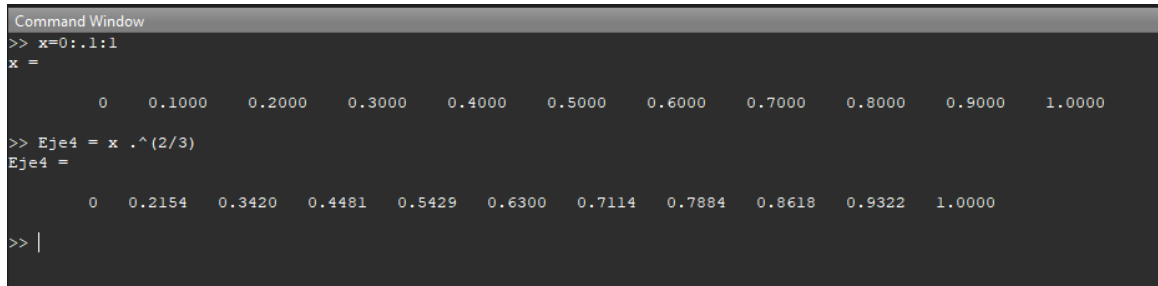
>> |
```

Figura 7: Comando para la raíz cuadrada de todos los elementos de un vector

5.4. Ejercicio 4.

Si $x=0:0.1:1$, escribe el comando de MATLAB que eleva cada componente de x a $2/3$

Al igual que el ejercicio anterior, se puede elevar todos los elementos del vector x al $2/3$ donde se guarda en un nuevo vector Eje4



```

Command Window
>> x=0:0.1:1
x =
    0    0.1000    0.2000    0.3000    0.4000    0.5000    0.6000    0.7000    0.8000    0.9000    1.0000

>> Eje4 = x .^(2/3)
Eje4 =
    0    0.2154    0.3420    0.4481    0.5429    0.6300    0.7114    0.7884    0.8618    0.9322    1.0000

>> |

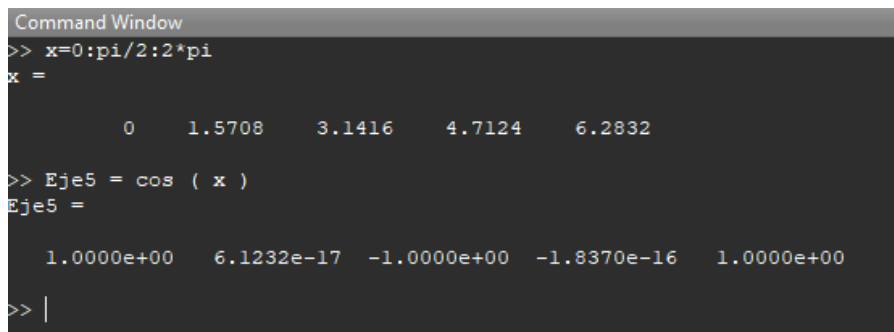
```

Figura 8: Comando para elevar al $2/3$ los elementos del vector

5.5. Ejercicio 5.

Si $x=0:\pi/2:2\pi$, escribe el comando MATLAB que calcula el coseno de cada componente de x

Dado que Octave contiene funciones trigonométricas que reciben un vector o un escalar, no es necesario iterar en todos los elementos del vector, solo basta con asignarle como argumento el vector que se quiere calcular el coseno.



```

Command Window
>> x=0:pi/2:2*pi
x =
    0    1.5708    3.1416    4.7124    6.2832

>> Eje5 = cos ( x )
Eje5 =
    1.0000e+00    6.1232e-17   -1.0000e+00   -1.8370e-16    1.0000e+00

>> |

```

Figura 9: Comando para obtener el coseno de un vector

5.6. Ejercicio 6.

Si $x = -1:0.1:1$, escribe el comando MATLAB que calcula el arcoseno de cada componente de x

Al igual que el ejercicio anterior, solo basta con pasar como argumento el vector al arcoseno, para así, obtener un vector con todos los elementos calculados del arcoseno.

```

Command Window
>> x=-1:0.1:1
x =

Columns 1 through 20:
-1.0000 -0.9000 -0.8000 -0.7000 -0.6000 -0.5000 -0.4000 -0.3000 -0.2000 -0.1000    0    0.1000 0.2000 0.3000 0.4000 0.5000 0.6000 0.7000 0.8000 0.9000 1.0000

Column 21:
1.0000

>> Eje6 = asin( x )
Eje6 =

-1.5708 -1.1198 -0.9273 -0.7754 -0.6435 -0.5236 -0.4115 -0.3047 -0.2014 -0.1002    0    0.1002 0.2014 0.3047 0.4115 0.5236 0.6435 0.7754 0.9273 1.1198 1.5708

>> |

```

Figura 10: Comando para obtener el arcoseno de un vector

5.7. Ejercicio 7.

Si $x = \text{linspace}(0, 2\pi, 1000)$, ¿cuál es la entrada 50 de x ? ¿Cuál es la longitud de x ?

El comando `linspace`, permite dividir un intervalo dado en un número n de elementos, donde la posición de los argumentos de la función es: `linspace(inicio, final, n_elementos)`, por lo tanto, para conocer el valor de la posición 50, se puede acceder mediante el siguiente comando

```

Columns 985 through 1000:
    6.1888    6.1951    6.2014    6.2077    6.2140    6.2203    6.2266    6.2329

>> Eje7 = x (50)
Eje7 = 0.3082
>>

```

Figura 11: Comando para conocer el valor de una posición en específico

5.8. Ejercicio 8.

Si $k=0:100$, ¿cuál es la entrada número 12 de $y=0.5.^k$?

Al igual que el ejercicio anterior, solo basta con declarar los vectores previos y acceder a ellos con el siguiente comando.

```
Command Window
>> k=0:100
k =

Columns 1 through 36:

    0     1     2     3     4     5     6     7     8     9    10    11    12

Columns 37 through 72:

    36    37    38    39    40    41    42    43    44    45    46    47    48

Columns 73 through 101:

    72    73    74    75    76    77    78    79    80    81    82    83    84

>> y=0.5.^k
y =

Columns 1 through 16:

    1.0000e+00    5.0000e-01    2.5000e-01    1.2500e-01    6.2500e-02    3.1250e-02

Columns 17 through 32:

    1.5259e-05    7.6294e-06    3.8147e-06    1.9073e-06    9.5367e-07    4.7684e-07

Columns 33 through 48:

    2.3283e-10    1.1642e-10    5.8208e-11    2.9104e-11    1.4552e-11    7.2760e-12

Columns 49 through 64:

    3.5527e-15    1.7764e-15    8.8818e-16    4.4409e-16    2.2204e-16    1.1102e-16

Columns 65 through 80:

    5.4210e-20    2.7105e-20    1.3553e-20    6.7763e-21    3.3881e-21    1.6941e-21

Columns 81 through 96:

    8.2718e-25    4.1359e-25    2.0680e-25    1.0340e-25    5.1699e-26    2.5849e-26

Columns 97 through 101:

    1.2622e-29    6.3109e-30    3.1554e-30    1.5777e-30    7.8886e-31

>> Eje8 = y(12)
Eje8 = 4.8828e-04
>> |
```

Figura 12: Comando para conocer el valor de una posición en específico

6. Conclusiones

Usar MATLAB como lenguaje puede ser muy útil en entornos donde se quiere obtener resultados rápidos con poco esfuerzo, sin importar la cantidad de memoria usada, los tipos de datos a usar, etc. Sin embargo, MATLAB requiere de una licencia para su correcto y legal uso, lamentablemente la universidad no cuenta con licencias disponibles. Como solución, se busca el uso de tecnologías o frameworks similares pero de código abierto, como es el caso del lenguaje Octave. Donde es ampliamente recomendado el uso de frameworks de código abierto para evitar problemas de derechos de autor en investigaciones o en la industria, inclusive, el uso de software no autorizado pone en riesgo la integridad de la maquina a usar.

Los primeros pasos usando este lenguaje resultaron amigable al usuario, ya que se aprecia que es muy enfocado a las operaciones matriciales, ademas, otra ventaja es el uso de un interpretador en lugar de un compilador, porque facilitara el desarrollo de funciones mas complejas, facilitando igual la depuración de código para detectar errores.

Con una experiencia previa a python, puede que a Octave le falten algunos métodos importantes para el uso y manipulación de elementos en matrices como es el caso de la librería de pandas en python, donde con métodos como `.tail()`, `.head()`, `.loc[]`, `.iloc[]`, etc facilitan muchísimo mas la manipulación.

En practicas posteriores se continua trabajando en este lenguaje dado que se necesita familiarizar un poco mas para un correcto desarrollo e implementacion de algoritmos.

Referencias

- Joag, P. S. (2016). *An introduction to vectors, vector operators and vector analysis*. Daryaganj, Delhi, India: Cambridge University Press.
- Linge, S., y Langtangen, H. P. (2016). *Programming for computations – matlab/octave: A gentle introduction to numerical simulations with matlab/octave*. Heidelberg: Springer. (Open Access under CC BY-NC 4.0 license) doi: 10.1007/978-3-319-32452-4
- Sebesta, R. W. (2012). *Concepts of programming languages* (10th ed.). Upper Saddle River, New Jersey: Pearson Education, Inc. (Published as Addison-Wesley)