

Universidad Autónoma de Querétaro

Facultad de Informática

Algoritmos Avanzados

Dra. Diana Córdova Esparza

Práctica #2

Matrices y MATLAB

Saúl Axel López Gómez

Agosto 4, 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.	6
4.5. Ejercicio 5.	6
4.6. Ejercicio 6.	7
4.7. Ejercicio 7.	7
4.8. Ejercicio 8.	7
4.9. Ejercicio 9.	8
5. Resultados	9
5.1. Ejercicio 1.	9
5.2. Ejercicio 2.	10
5.3. Ejercicio 3.	10
5.4. Ejercicio 4.	11
5.5. Ejercicio 5.	13
5.6. Ejercicio 6.	14
5.7. Ejercicio 7.	15
5.8. Ejercicio 8.	15
5.9. Ejercicio 9.	17
6. Conclusiones	18
Referencias	19

Índice de figuras

1.	Comando para generar una matriz	9
2.	Comando para una matriz	9
3.	Matriz de elementos igual a -3	10
4.	Matriz de Hilbert	10
5.	Diferencia de comandos para redondear valores	12
6.	Matriz de numeros aleatorios	13
7.	Diferencias entre operadores ', .' para matrices complejas	14
8.	Comando para conocer el valor de una posición en específico	15
9.	Matriz de filas pares	15
10.	Matriz de filas impares	16
11.	Matriz de una fila	16
12.	Matriz a partir de vectores	17

Índice de Códigos

1.	Ejercicio 1: 1) Matriz 3x4	4
2.	Ejercicio 1: 2) Matriz 4x3	4
3.	Ejercicio 2: Matriz 3x5	5
4.	Ejercicio 3: Matriz de Hilbert	5
5.	Ejercicio 4: Metodo floor	6
6.	Ejercicio 4: Metodo floor	6
7.	Ejercicio 4: Metodo floor	6
8.	Ejercicio 4: Metodo floor	6
9.	Ejercicio 5: Coseno de los elementos del vector	6
10.	Ejercicio 6: Matriz compleja	7
11.	Ejercicio 7: linspace de un vector	7
12.	Ejercicio 8: Filas pares de una matriz	7
13.	Ejercicio 8: Filas impares de una matriz	7
14.	Ejercicio 8: Ultima fila de una matriz	8
15.	Ejercicio 9: Contruccion de una matriz con vectores	8

1. Introducción

La manipulación de datos a nivel multidimensional resulta ser una tarea compleja para solucionar un problema analíticamente, es por esto, que se ha propuesto la implementación de matrices en sistemas computacionales para reducir el error humano, por lo tanto, el dominio teórico y práctico resulta ser de vital utilidad para seguir desarrollando algoritmos eficientes, robustos que puedan resolver problemáticas existentes.

MATLAB contiene un sistema fácil para el manejo de matrices, al igual que las operaciones básicas, como lo son la multiplicación por un escalar, transposición de matrices, conjugadas, multiplicación, etc. Por lo tanto, en este documento, se propone la realización de ejercicios clave para la familiarización del uso del lenguaje Octave como alternativa a MATLAB, todo dentro del ámbito matricial.

2. Fundamentos teóricos

Según ([Hartman, 2011](#)) una matriz es un arreglo rectangular de números donde los números horizontales representan las filas y los números verticales representan las columnas, en este sentido si se cuentan con demasiada cantidad de números, esta representación matricial puede crecer a más dimensiones, donde se dice que es una matriz $m \times n$, donde la representación es de la forma:

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}$$

Cuando se tiene una matriz cuadrada, $n \times n$, se puede representar directamente la matriz de Hilbert, donde viene dada por:

$$H_{ij} = \frac{1}{i + j - 1} \quad (1)$$

En esta matriz es de suma importancia, dado que pequeños errores en los datos o un mal redondeo por parte de los métodos floor, ceil, round y fix, pueden generar grandes errores en la solución de un problema ([Montes-Rodríguez y Virtanen, 2024](#))

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 segunda 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 donde se especializara en operaciones matriciales.

4.1. Ejercicio 1.

Escribe los comandos de MATLAB que generan las siguientes matrices.

1.
$$\begin{pmatrix} 1 & 1 & 3 & 0 \\ -3 & -1 & 6 & 8 \\ 4 & 4 & 10 & -7 \end{pmatrix}$$

Dado que es una matriz 3x4, se propone el siguiente comando para generar la matriz:

```
1 Eje1_1 = [1, 1, 3, 0; -3, -1, 6, 8; 4, 4, 10, -7]
2
```

Código 1: Ejercicio 1: 1) Matriz 3x4

2.
$$\begin{pmatrix} 10 & 5 & -6 \\ -5 & -8 & -4 \\ -5 & -10 & 3 \\ 8 & 8 & -5 \end{pmatrix}$$

En este caso, es una matriz 4x3, se propone el siguiente comando:

```
1 Eje1_2 = [10, 5, -6; -5, -8, -4; -5, -10, 3; 8, 8, -5]
2
```

Código 2: Ejercicio 1: 2) Matriz 4x3

4.2. Ejercicio 2.

Escribe un solo comando que cree una matriz 3x5 con cada entrada igual a -3.

La forma mas sencilla es usando la multiplicación de un escalar por una matriz, donde se puede generar primeramente una matriz de tamaño 3x5, seguidamente multiplicar por el escalar -3, como en el siguiente comando

```
1 Eje2 = -3 * ones(3, 5)
2
```

Código 3: Ejercicio 2: Matriz 3x5

4.3. Ejercicio 3.

Crea una matriz de Hilbert con los siguientes comandos, ademas, escribe una formula para la posición (i,j) de la matriz H. Nota: **Format rat** no funciona en Octave

```
1 format rat
2 H=hilb(5)
3 format
4
```

Código 4: Ejercicio 3: Matriz de Hilbert

Las indicaciones muestran que no funciona Format rat, sin embargo, en versiones superiores ($\geq 5.x$) del lenguaje Octave, Format rat es incluido propiamente como un comando similar a lo que se conoce en MATLAB.

La ecuación para la posición (i,j) de la matriz de Hilbert, viene dada con:

$$H_{ij} = \frac{1}{i + j - 1} \quad (2)$$

4.4. Ejercicio 4.

Explica la diferencia entre los comandos `floor`, `ceil`, `round` y `fix`. Apoya tu explicación con ejemplos en MATLAB.

Los 4 métodos sirven para redondear valores flotantes a un entero, supongamos el siguiente vector de flotantes:

Eje4 = [0.4, 0.8, -1.2, 0.1, -4.9]

■ floor

```
1 Eje4_floor = floor(Eje4)
2
```

Código 5: Ejercicio 4: Metodo floor

■ ceil

```
1 Eje4_ceil = ceil(Eje4)
2
```

Código 6: Ejercicio 4: Metodo floor

■ round

```
1 Eje4_round = round(Eje4)
2
```

Código 7: Ejercicio 4: Metodo floor

■ fix

```
1 Eje4_fix = fix(Eje4)
2
```

Código 8: Ejercicio 4: Metodo floor

4.5. Ejercicio 5.

Escribe un comando MATLAB que genere una matriz 4x4 con valores aleatorios enteros entre -5 y 5.

El siguiente comando genera numeros enteros aleatorios dentro del rango [-5, 5]

```
1 Eje5 = randi([-5, 5], 4, 4)
2
```

Código 9: Ejercicio 5: Coseno de los elementos del vector

4.6. Ejercicio 6.

El operador `'` genera realmente la conjugada traspuesta de una matriz. Para verlo, introduzca la matriz `A=[1,2+i,sqrt(2);3,3-i,sqrt(-1)]`, teclea entonces entonces `A'`. Describe el resultado. ¿Qué ocurre si ponemos `A.'`? Explica la diferencia entre los operadores de transposición `'` y `.`.

El siguiente comando define la matriz compleja, al igual, se obtienen la transpuesta conjugada y una reorganización de filas y columnas, los resultados y el análisis se encuentran en la sección de resultados [5](#)

```
1 A=[1,2+i,sqrt(2);3,3-i,sqrt(-1)]
2 A'
3 A.'
4
```

Código 10: Ejercicio 6: Matriz compleja

4.7. Ejercicio 7.

¿Cuál es la entrada en fila 5 y columna 7 de la matriz `pascal(10)`?

Dado que solo se quiere acceder a un elemento en específico, se puede usar:

```
1 Eje7 = pascal(10)
2 Eje7_ = Eje7(5,7)
3
```

Código 11: Ejercicio 7: linspace de un vector

4.8. Ejercicio 8.

Sea `T=toeplitz(1:7)`. Escribe un comando de MATLAB que:

1. Coloca las filas pares de la matriz `T` en una matriz `B`.

```
1 Eje8_even = T([2:2:7],[1:7])
2
```

Código 12: Ejercicio 8: Filas pares de una matriz

2. Coloca las filas impares de la matriz `T` en una matriz `C`.

```
1 Eje8_odd = T([1:2:7],[1:7])
2
```

Código 13: Ejercicio 8: Filas impares de una matriz

3. Coloca la última columna de la matriz T en un vector b.

```
1 b = T([end],[1:7])  
2
```

Código 14: Ejercicio 8: Ultima fila de una matriz

4.9. Ejercicio 9.

Sea $x=[0,\pi/2,2\pi]$. Construye una matriz, con comandos de MATLAB, cuya primera fila es x , la segunda fila está formada por el seno de cada entrada de x , y la tercera fila es el coseno de cada entrada de x .

Primero se necesita declarar los 3 vectores a usar para después construir la matriz

```
1 x=[0,pi/2,2*pi]  
2 sen_x = sin(x)  
3 cos_x = cos(x)  
4 Eje9 = [x; sen_x; cos_x]  
5
```

Código 15: Ejercicio 9: Contruccion de una matriz con vectores

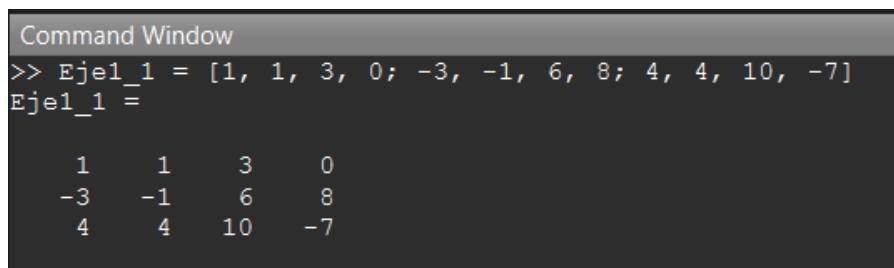
5. Resultados

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.

$$1. \begin{pmatrix} 1 & 1 & 3 & 0 \\ -3 & -1 & 6 & 8 \\ 4 & 4 & 10 & -7 \end{pmatrix}$$

Escribiendo el comando propuesto en Octave, se obtiene una matriz de la forma de la imagen 1



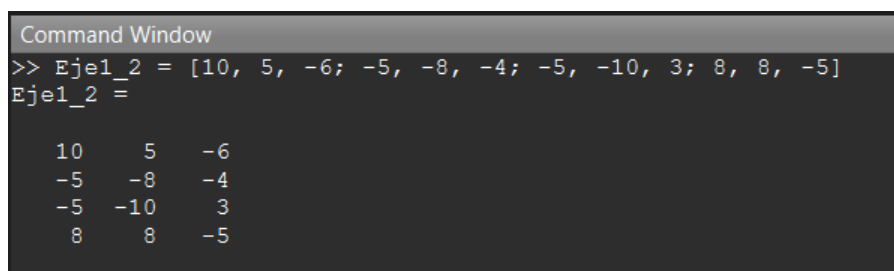
```
Command Window
>> Eje1_1 = [1, 1, 3, 0; -3, -1, 6, 8; 4, 4, 10, -7]
Eje1_1 =

     1     1     3     0
    -3    -1     6     8
     4     4    10    -7
```

Figura 1: Comando para generar una matriz

$$2. \begin{pmatrix} 10 & 5 & -6 \\ -5 & -8 & -4 \\ -5 & -10 & 3 \\ 8 & 8 & -5 \end{pmatrix}$$

El comando propuesto obtuvo una respuesta positiva, igual a la esperada, ya que en la imagen 2, muestra correctamente el la matriz esperada



```
Command Window
>> Eje1_2 = [10, 5, -6; -5, -8, -4; -5, -10, 3; 8, 8, -5]
Eje1_2 =

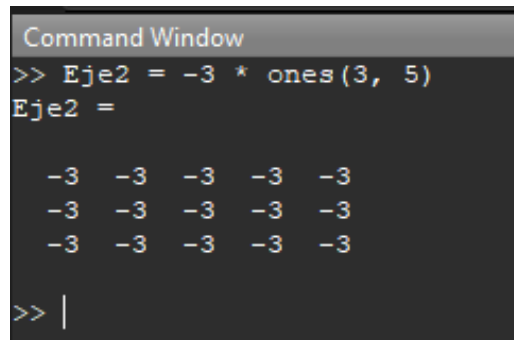
    10     5    -6
    -5    -8    -4
    -5   -10     3
     8     8    -5
```

Figura 2: Comando para una matriz

5.2. Ejercicio 2.

Escribe un solo comando que cree una matriz 3x5 con cada entrada igual a -3.

Dado que todos los elementos de la matriz se espera que sean del mismo valor, se puede generar una matriz de unos de las dimensiones 3x5 para luego, multiplicar cada elemento por un escalar igual a -3



```
Command Window
>> Eje2 = -3 * ones(3, 5)
Eje2 =

    -3    -3    -3    -3    -3
    -3    -3    -3    -3    -3
    -3    -3    -3    -3    -3

>> |
```

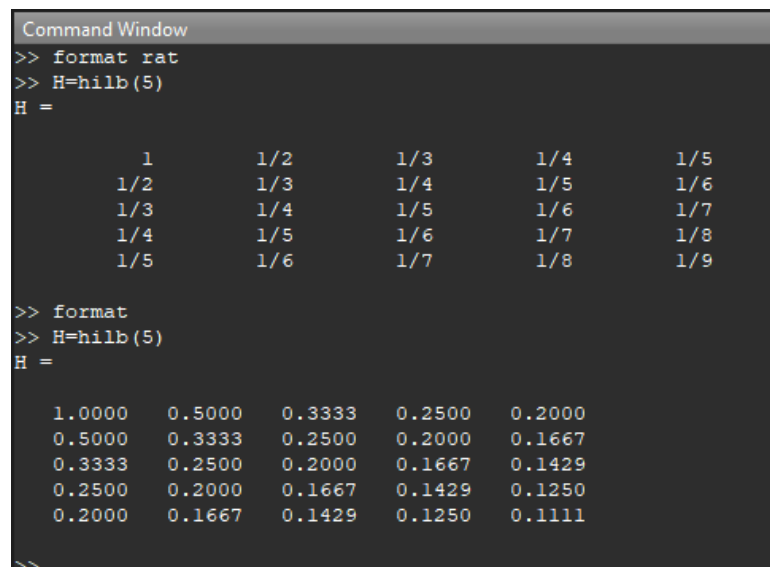
Figura 3: Matriz de elementos igual a -3

5.3. Ejercicio 3.

Crea una matriz de Hilbert con los siguientes comandos, además, escribe una fórmula para la posición (i,j) de la matriz H. Nota: **Format rat** no funciona en Octave

A partir de versiones recientes de Octave, se implementó la funcionalidad **Format rat** que muestra de forma visual una aproximación de los números racionales que representa cada elemento de la matriz en forma fraccionaria.

El comando **format** nos permite restablecer la forma visual de la matriz en formato racional.



```
Command Window
>> format rat
>> H=hilb(5)
H =

      1      1/2      1/3      1/4      1/5
    1/2      1/3      1/4      1/5      1/6
    1/3      1/4      1/5      1/6      1/7
    1/4      1/5      1/6      1/7      1/8
    1/5      1/6      1/7      1/8      1/9

>> format
>> H=hilb(5)
H =

    1.0000    0.5000    0.3333    0.2500    0.2000
    0.5000    0.3333    0.2500    0.2000    0.1667
    0.3333    0.2500    0.2000    0.1667    0.1429
    0.2500    0.2000    0.1667    0.1429    0.1250
    0.2000    0.1667    0.1429    0.1250    0.1111

>>
```

Figura 4: Matriz de Hilbert

5.4. Ejercicio 4.

Explica la diferencia entre los comandos `floor`, `ceil`, `round` y `fix`. Apoya tu explicación con ejemplos en MATLAB.

Los cuatro comandos hacen la misma función, redondear, pero para enfoques diferentes, a pesar de que se propuso un vector, el cual se usa como entrada a los 4 métodos, se obtienen diferentes resultados, por lo tanto cada metodo hace:

- **floor**: Redondea el numero flotante hacia los negativos como:

1. $\text{floor}(0.4) = 0$
2. $\text{floor}(0.8) = 0$
3. $\text{floor}(-1.2) = -2$
4. $\text{floor}(0.1) = 0$
5. $\text{floor}(-4.9) = -5$

- **ceil**: Redondea el numero flotante hacia los positivos como:

1. $\text{ceil}(0.4) = 1$
2. $\text{ceil}(0.8) = 1$
3. $\text{ceil}(-1.2) = -1$
4. $\text{ceil}(0.1) = 1$
5. $\text{ceil}(-4.9) = -4$

- **round**: Redondea el numero flotante hacia el entero mas cercano como:

1. $\text{round}(0.4) = 0$
2. $\text{round}(0.8) = 1$
3. $\text{round}(-1.2) = -1$
4. $\text{round}(0.1) = 0$
5. $\text{round}(-4.9) = -5$

- **fix**: Trunca el numero decimal como:

1. $\text{round}(0.4) = 0$
2. $\text{round}(0.8) = 0$
3. $\text{round}(-1.2) = -1$
4. $\text{round}(0.1) = 0$
5. $\text{round}(-4.9) = -4$

```
Command Window
>> Eje4 = [0.4, 0.8, -1.2, 0.1, -4.9]
Eje4 =

    0.4000    0.8000   -1.2000    0.1000   -4.9000

>> Eje4_floor = floor(Eje4)
Eje4_floor =

     0     0    -2     0    -5

>> Eje4_ceil = ceil(Eje4)
Eje4_ceil =

     1     1    -1     1    -4

>> Eje4_round = round(Eje4)
Eje4_round =

     0     1    -1     0    -5

>> Eje4_fix = fix(Eje4)
Eje4_fix =

     0     0    -1     0    -4
```

Figura 5: Diferencia de comandos para redondear valores

5.5. Ejercicio 5.

Escribe un comando MATLAB que genere una matriz 4x4 con valores aleatorios enteros entre -5 y 5 .

Con el siguiente comando es posible generar matrices de cualquier tamaño con numero aleatorios enteros:

```
>> A = randi([-5, 5], 4, 4)
A =

     2     -3     -5      1
     3      1     -5      1
     0      1      5      5
     4     -2      1     -5
```

Figura 6: Matriz de numeros aleatorios

5.6. Ejercicio 6.

El operador `'` genera realmente la conjugada traspuesta de una matriz. Para verlo, introduzca la matriz $A = [1, 2+i, \sqrt{2}; 3, 3-i, \sqrt{-1}]$, teclea entonces entonces A' . Describe el resultado. ¿Qué ocurre si ponemos $A.'$? Explica la diferencia entre los operadores de transposición `'` y `.`.

En matrices complejas, los operadores `'` y `.` tienen un propósito distinto:

El operador `'` : Conjuga cada elemento complejo, o sea, cambia de signo de la parte imaginaria

El operador `.` : Transpone la matriz, intercambiando filas por columnas.

En matrices no complejas, se obtiene el mismo resultado con ambos operadores. En la imagen siguiente, se observa en el cuadrado rojo, como los signos fueron cambiados, mientras que en el recuadro azul, se mantuvieron los signos originales.

```

Command Window
>> A=[1,2+i,sqrt(2);3,3-i,sqrt(-1)]
A =

    1.0000 + 0i    2.0000 + 1.0000i    1.4142 + 0i
    3.0000 + 0i    3.0000 - 1.0000i    0 + 1.0000i

>> A'
ans =

    1.0000 - 0i    3.0000 - 0i
    2.0000 - 1.0000i    3.0000 + 1.0000i
    1.4142 - 0i    0 - 1.0000i

>> A.'
ans =

    1.0000 + 0i    3.0000 + 0i
    2.0000 + 1.0000i    3.0000 - 1.0000i
    1.4142 + 0i    0 + 1.0000i
  
```

Figura 7: Diferencias entre operadores `'`, `.` para matrices complejas

5.7. Ejercicio 7.

¿Cuál es la entrada en fila 5 y columna 7 de la matriz `pascal(10)`?

Primero se tiene que definir la matriz de pascal, una vez que se define, se puede acceder a un elemento en específico, con la ubicación de la fila y la columna

```
>> Eje7 = pascal(10)
Eje7 =

     1     1     1     1     1     1     1     1     1     1
     1     2     3     4     5     6     7     8     9    10
     1     3     6    10    15    21    28    36    45    55
     1     4    10    20    35    56    84   120   165   220
     1     5    15    35    70   126   210   330   495   715
     1     6    21    56   126   252   462   792  1287  2002
     1     7    28    84   210   462   924  1716  3003  5005
     1     8    36   120   330   792  1716  3432  6435 11440
     1     9    45   165   495  1287  3003  6435 12870 24310
     1    10    55   220   715  2002  5005 11440 24310 48620

>> Eje7(5,7)
ans = 210
>> |
```

Figura 8: Comando para conocer el valor de una posición en específico

5.8. Ejercicio 8.

Sea `T=toeplitz(1:7)`. Escribe un comando de MATLAB que:

1. Coloca las filas pares de la matriz `T` en una matriz `B`.

En la siguiente imagen, cada rectángulo rojo denota la fila que se necesita asignar al nuevo vector, usando el comando propuesto, efectivamente podemos apreciar que se obtiene los renglones esperados

```
>> T=toeplitz(1:7)
T =

     1     2     3     4     5     6     7
     2     1     2     3     4     5     6
     3     2     1     2     3     4     5
     4     3     2     1     2     3     4
     5     4     3     2     1     2     3
     6     5     4     3     2     1     2
     7     6     5     4     3     2     1

>> Eje8_even = T([2:2:7],[1:7])
Eje8_even =

     2     1     2     3     4     5     6
     4     3     2     1     2     3     4
     6     5     4     3     2     1     2

>> |
```

Figura 9: Matriz de filas pares

2. Coloca las filas impares de la matriz T en una matriz C.

En este ejercicio, se pide las filas impares denotadas por rectángulos de color amarillo, los cuales, despues de ingresar el comando, efectivamente se obtienen al nuevo vector generado

```
>> T=toeplitz(1:7)
T =
1 2 3 4 5 6 7
2 1 2 3 4 5 6
3 2 1 2 3 4 5
4 3 2 1 2 3 4
5 4 3 2 1 2 3
6 5 4 3 2 1 2
7 6 5 4 3 2 1

>> Eje8_odd = T([1:2:7],[1:7])
Eje8_odd =
1 2 3 4 5 6 7
3 2 1 2 3 4 5
5 4 3 2 1 2 3
7 6 5 4 3 2 1
```

Figura 10: Matriz de filas impares

3. Coloca la última columna de la matriz T en un vector b.

Para acceder a la ultima fila, solo es necesario el método end para indicar que se necesita la ultima fila

```
>> T=toeplitz(1:7)
T =
1 2 3 4 5 6 7
2 1 2 3 4 5 6
3 2 1 2 3 4 5
4 3 2 1 2 3 4
5 4 3 2 1 2 3
6 5 4 3 2 1 2
7 6 5 4 3 2 1

>> b = T([end],[1:7])
b =
7 6 5 4 3 2 1

>>
```

Figura 11: Matriz de una fila

5.9. Ejercicio 9.

Sea $x=[0,\pi/2,2\pi]$. Construye una matriz, con comandos de MATLAB, cuya primera fila es x , la segunda fila está formada por el seno de cada entrada de x , y la tercera fila es el coseno de cada entrada de x .

Primero se debe de definir los 3 vectores necesarios para construir la matriz. luego se puede hacer la llamada de cada uno de ellos para generar la matriz

```
>> x=[0,pi/2,2*pi]
x =
    0    1.5708    6.2832

>> sen_x = sin(x)
sen_x =
    0    1.0000   -0.0000

>> cos_x = cos(x)
cos_x =
    1.0000e+00    6.1232e-17    1.0000e+00

>> Eje9 = [x; sen_x; cos_x]
Eje9 =
    0    1.5708    6.2832
    0    1.0000   -0.0000
    1.0000    0.0000    1.0000
```

Figura 12: Matriz a partir de vectores

6. Conclusiones

La ventaja de usar MATLAB o un lenguaje basado en el, como Octave, radica en la facilidad de generar matrices, acceder a los elementos, hacer operaciones matriciales, ya que, en comparación con otros lenguajes de programación como C++ es necesario definir los tipos de datos, crear vectores para almacenar los elementos y finalmente iterar entre ellos con ciclos For anidados para poder recorrer toda la matriz y encontrar el elemento que se necesita. A veces, realizar todos los pasos mencionados previamente, se vuelve un poco tedioso e inclusive, a veces se pueden cometer errores.

Con un lenguaje basado en un interpretador, se vuelve una tarea muy fácil que toma segundos al realizar operaciones matriciales, siempre y cuando, el uso de memoria y los tipos de datos no sean de interés, dado que Octave se puede usar solo para modelar soluciones rápidas. En sistemas mas complejos, en algoritmos que estarán en algún producto, es estrictamente necesario hacer la migración a lenguaje de programación mas robustos.

Faltan muchos métodos por descubrir en Octave, los cuales, ayudaran a optimizar tiempo de programación, pero en la practica y conforme se vayan resolviendo problemas puntuales, se pueden ir descubriendo. Sin embargo, como una introducción a las matrices, esta practica resulta muy adecuada para conocer como se pueden definir y operar en matrices.

Referencias

- Hartman, G. (2011). *Fundamentals of matrix algebra* (Third Edition, Version 3.1110 ed.). Department of Mathematics and Computer Science, Virginia Military Institute: Autor.
- Montes-Rodríguez, A., y Virtanen, J. A. (2024). The hilbert matrix done right. *arXiv preprint*. doi: 10.48550/arXiv.2411.07324