



UNIVERSIDAD AUTÓNOMA DE QUERÉTARO
FACULTAD DE INGENIERÍA

Universidad Autónoma de Querétaro

FACULTAD DE INGENIERÍA

ELIMINACIÓN GAUSSIANA Y GAUSS-JORDAN

Análisis numérico

Autor:

David Gómez Torres

1 Septiembre del 2021

1. Introducción	1
1.1. Solución de sistemas pequeños de ecuaciones	1
1.1.1. Método gráfico	1
1.1.2. Determinantes y la regla de Cramer	2
1.1.3. La eliminación de incógnitas	3
1.2. Eliminación de Gauss simple	4
1.3. Técnicas para mejorar las soluciones	5
1.3.1. Pivoteo	5
1.4. Sistema de ecuaciones no lineales	6
1.5. Gauss-Jordan	6
2. Metodología	7
2.1. Multiplicación de matrices	7
2.2. Sistema de ecuaciones lineales (1)	9
2.3. Sistema de ecuaciones lineales (2)	11
2.4. Sistema de ecuaciones lineales (3)	14
2.5. Sistema de ecuaciones lineales (4)	16
2.6. Masas atadas por resortes	19
3. Anexos	21
3.1. Anexo A: Evidencia del funcionamiento de los código reportados	21
4. Bibliografía	25

CAPÍTULO 1

INTRODUCCIÓN

En esta sección se presenta la parte teórica de los métodos empleados para resolver los ejercicios, haciendo un resumen de las páginas 191-216 correspondientes al libro [1].

1.1. Solución de sistemas pequeños de ecuaciones

1.1.1. Método gráfico

Para dos ecuaciones se puede obtener una solución al graficarlas en coordenadas cartesianas con un eje que corresponda a x_1 y el otro a x_2 . Debido a que en estos sistemas lineales, cada ecuación se relaciona con una línea recta, lo cual se ilustra fácilmente mediante las ecuaciones generales

$$a_{11}x_1 + a_{12}x_2 = b_1$$

$$a_{21}x_1 + a_{22}x_2 = b_2$$

En ambas ecuaciones se puede despejar x_2 :

$$x_2 = -\left(\frac{a_{11}}{a_{12}}\right)x_1 + \frac{b_1}{a_{12}}$$

$$x_2 = -\left(\frac{a_{21}}{a_{22}}\right)x_1 + \frac{b_2}{a_{22}}$$

De esta manera, las ecuaciones ahora están en la forma de líneas rectas; es decir, $x_2 = (\text{pendiente}) x_1 + \text{intercepto}$. Tales líneas se grafican en coordenadas cartesianas con x_2 como la ordenada y x_1 como la abscisa. Los valores de x_1 y x_2 en la **intersección** de las líneas representa la **solución**.

Para tres ecuaciones simultáneas, cada ecuación se representa como un plano en un sistema de coordenadas tridimensional. El punto en donde se intersecan los tres planos representa la solución. Para más de tres incógnitas, los métodos gráficos no funcionan y, por consiguiente, tienen poco valor práctico para resolver ecuaciones simultáneas.

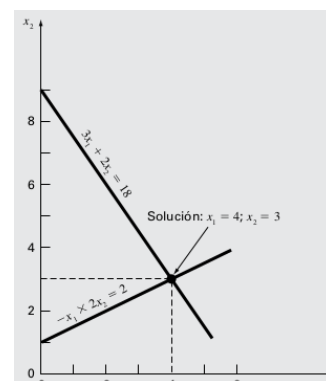


Figura 1.1: Ejemplo de intersección de las rectas

1.1.2. Determinantes y la regla de Cramer

La regla de Cramer es otra técnica de solución adecuada para un sistema pequeño de ecuaciones. Antes de hacer una descripción de tal método, se mencionará en forma breve el concepto de determinante que se utiliza en la regla de Cramer. Además, el determinante tiene relevancia en la evaluación del mal condicionamiento de una matriz.

Determinantes

El determinante se puede ilustrar para un sistema de tres ecuaciones simultáneas:

$$[A]\{X\} = \{B\}$$

donde $[A]$ es la matriz de coeficientes:

$$[A] = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

El determinante D de este sistema se forma, a partir de los coeficientes de la ecuación, de la siguiente manera:

$$D = \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} \quad (1.1)$$

Aunque el determinante D y la matriz de coeficientes $[A]$ se componen de los mismos elementos, son conceptos matemáticos completamente diferentes. Por esto, para distinguirlos visualmente se emplean corchetes para encerrar la matriz y líneas rectas verticales para el determinante. En contraste con una matriz, el determinante es un simple número.

Por ejemplo, el valor del determinante de segundo orden

$$D = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix}$$

se calcula como

$$D = a_{11}a_{22} - a_{21}a_{12} \quad (1.2)$$

En el caso del determinante de tercer orden el determinante, que es un simple valor numérico, se calcula así

$$D = a_{11} \begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} - a_{12} \begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix} + a_{13} \begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix} \quad (1.3)$$

donde a los determinantes de 2 por 2 se les llama **menores**.

Regla de Cramer

Esta regla establece que cada incógnita de un sistema de ecuaciones lineales algebraicas puede expresarse como una fracción de dos determinantes con denominador D y con el numerador obtenido a partir de D , al reemplazar la columna de coeficientes de la incógnita en cuestión por las constantes b_1, b_2, \dots, b_n .

Por ejemplo, x_1 se calcula como

$$x_1 = \frac{\begin{vmatrix} b_1 & a_{12} & a_{13} \\ b_2 & a_{22} & a_{23} \\ b_3 & a_{23} & a_{33} \end{vmatrix}}{D} \quad (1.4)$$

1.1.3. La eliminación de incógnitas

La eliminación de incógnitas mediante la combinación de ecuaciones es un método algebraico que se ilustra con un sistema de dos ecuaciones simultáneas:

$$a_{11}x_1 + a_{12}x_2 = b_1 \quad (1.5)$$

$$a_{21}x_1 + a_{22}x_2 = b_2 \quad (1.6)$$

La estrategia básica consiste en multiplicar las ecuaciones por constantes, de tal forma que se elimine una de las incógnitas cuando se combinen las dos ecuaciones:

Por ejemplo, la ecuación (1.5) se multiplica por a_{21} y la ecuación (1.6) por a_{11} para dar

$$a_{11}a_{21}x_1 + a_{12}a_{21}x_2 = a_{21}b_1 \quad (1.7)$$

$$a_{21}a_{11}x_1 + a_{22}a_{11}x_2 = a_{11}b_2 \quad (1.8)$$

Restando la ecuación (1.7) de la (1.8) se elimina el término x_1 de las ecuaciones para obtener

$$a_{12}a_{21}x_2 - a_{22}a_{11}x_2 = a_{21}b_1 - a_{11}b_2 \quad (1.9)$$

de donde se puede despejar

$$x_2 = \frac{a_{21}b_1 - a_{11}b_2}{a_{22}a_{11} - a_{12}a_{21}} \quad (1.10)$$

La ecuación (1.10) se puede entonces sustituir en la ecuación (1.5), de la cual se puede despejar

$$x_1 = \frac{a_{22}b_1 - a_{12}b_2}{a_{11}a_{22} - a_{12}a_{21}} \quad (1.11)$$

Observe que las ecuaciones (1.10) y (1.11) se relacionan directamente con la regla de Cramer, que establece

$$x_1 = \frac{\begin{vmatrix} b_1 & a_{12} \\ b_2 & a_{22} \end{vmatrix}}{\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix}} = \frac{b_1 a_{22} - a_{12} b_2}{a_{11} a_{22} - a_{12} a_{21}}$$

$$x_2 = \frac{\begin{vmatrix} a_{11} & b_1 \\ a_{21} & b_2 \end{vmatrix}}{\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix}} = \frac{a_{11} b_2 - b_1 a_{21}}{a_{11} a_{22} - a_{12} a_{21}}$$

1.2. Eliminación de Gauss simple

El método está ideado para resolver un sistema general de n ecuaciones:

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n = b_1 \quad (1.12a)$$

$$a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \cdots + a_{2n}x_n = b_2 \quad (1.12b)$$

$$\vdots = \vdots$$

$$a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + \cdots + a_{nn}x_n = b_n \quad (1.12c)$$

Como en el caso de dos ecuaciones, la técnica para resolver n ecuaciones consiste en dos fases: la eliminación de las incógnitas y su solución mediante sustitución hacia atrás.

Eliminación hacia adelante de incógnitas

La primera fase consiste en reducir el conjunto de ecuaciones a un sistema triangular superior. El paso inicial será eliminar la primera incógnita, x_1 , desde la segunda hasta la n -ésima ecuación

$$a_{21}x_1 + \frac{a_{21}}{a_{11}}a_{12}x_2 + \cdots + \frac{a_{21}}{a_{1n}}a_{1n}x_n = \frac{a_{21}}{a_{1n}}b_1 \quad (1.13)$$

El procedimiento se repite después con las ecuaciones restantes y da como resultado el siguiente sistema modificado:

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n = b_1 \quad (1.14a)$$

$$a'_{22}x_2 + a'_{23}x_3 + \cdots + a'_{2n}x_n = b'_2 \quad (1.14b)$$

$$\vdots = \vdots$$

$$a'_{n1}x_1 + a'_{n2}x_2 + a'_{n3}x_3 + \cdots + a'_{nn}x_n = b'_n \quad (1.14c)$$

La ecuación (1.12a) se llama la **ecuación pivote**, y a_{11} se denomina el **coeficiente** o **elemento pivote**.

El procedimiento puede continuar usando las ecuaciones pivote restantes. La última manipulación en esta secuencia es el uso de la $(n - 1)$ -ésima ecuación para eliminar el término x_{n-1} de la n -ésima ecuación. Aquí el sistema se habrá transformado en un sistema triangular superior

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n = b_1 \quad (1.15a)$$

$$a'_{22}x_2 + a'_{23}x_3 + \cdots + a'_{2n}x_n = b'_2 \quad (1.15b)$$

$$\vdots = \vdots$$

$$a_{nm}^{(n-1)}x_n = b_n^{(n-1)} \quad (1.15c)$$

Eliminación hacia atrás de incógnitas

De la ecuación (1.15c) ahora se despeja x_n :

$$X_n \frac{b_n^{(n-1)}}{a_{nm}^{(n-1)}} \quad (1.16)$$

Este resultado se puede sustituir hacia atrás en la $(n - 1)$ -ésima ecuación y despejar x_{n-1} . El procedimiento, que se repite para evaluar las x restantes, se representa mediante la fórmula:

$$X_i = \frac{b_i^{(i+1)} - \sum_{j=i+1}^n a_{ij}^{(i-1)} x_j}{a_{ij}^{(i-1)}} \quad (1.17)$$

1.3. Técnicas para mejorar las soluciones

1.3.1. Pivoteo

Como se mencionó ocurren problemas obvios cuando un elemento pivote es cero, ya que el paso de normalización origina una división entre cero debido a que si la magnitud del elemento pivote es pequeña comparada con los otros elementos, entonces se pueden introducir errores de redondeo.

Los renglones se pueden intercambiar de manera que el elemento más grande sea el elemento pivote; esto se conoce como **pivoteo parcial**. Al procedimiento, donde tanto en las columnas como en los renglones se busca el elemento más grande y luego se intercambian, se le conoce como **pivoteo completo**, el cual se usa en muy raras ocasiones debido a que al intercambiar columnas se cambia el orden de las x y, en consecuencia, se agrega complejidad significativa y usualmente injustificada al programa de computadora.

1.4. Sistema de ecuaciones no lineales

Un procedimiento para resolver tales sistemas se basa en la versión multidimensional del método de Newton-Raphson. Así, se escribe para cada ecuación una expansión de la serie de Taylor. Dichas ecuaciones de la forma son escritas para cada una de las ecuaciones no lineales originales. Después, todos los términos $f_{k,i+1}$ se igualan a cero, como sería el caso en la raíz, y la ecuación se escribe como:

$$= x_{1,i+1} \frac{\partial f_{k,i}}{\partial x_1} + x_{2,i+1} \frac{\partial f_{k,i}}{\partial x_2} + \cdots + x_{n,i} \frac{\partial f_{k,i}}{\partial x_n} \quad (1.18)$$

Las derivadas parciales se expresan como

$$[Z] = \begin{bmatrix} \frac{\partial f_{1,i}}{\partial x_1} & \frac{\partial f_{1,i}}{\partial x_2} & \cdots & \frac{\partial f_{1,i}}{\partial x_n} \\ \frac{\partial f_{2,i}}{\partial x_1} & \frac{\partial f_{2,i}}{\partial x_2} & \cdots & \frac{\partial f_{2,i}}{\partial x_n} \\ \frac{\partial f_{n,i}}{\partial x_1} & \frac{\partial f_{n,i}}{\partial x_2} & \cdots & \frac{\partial f_{n,i}}{\partial x_n} \end{bmatrix}$$

Se debe notar que el procedimiento anterior tiene dos desventajas importantes. Primero, a menudo no es fácil evaluar la ecuación (1.4). Por lo que se ha desarrollado una variación del método de Newton-Raphson para evitar tal problema. Como podría esperarse, tal variación se basa en el uso de aproximaciones por diferencias finitas, para calcular las derivadas parciales que aparecen en $[Z]$.

La segunda desventaja del método de Newton-Raphson para multiecuaciones es que usualmente se requiere de excelentes valores iniciales para asegurar la convergencia. Ya que con frecuencia esto es difícil de obtener, se han desarrollado métodos alternos que, aunque son más lentos que el método de Newton-Raphson, dan un mejor comportamiento de convergencia.

1.5. Gauss-Jordan

El método de Gauss-Jordan es una variación de la eliminación de Gauss. La principal diferencia consiste en que cuando una incógnita se elimina en el método de Gauss-Jordan, ésta es eliminada de todas las otras ecuaciones, no sólo de las subsecuentes. Además, todos los renglones se normalizan al dividirlos entre su elemento pivote. De esta forma, el paso de eliminación genera una matriz identidad en vez de una triangular. En consecuencia, no es necesario usar la sustitución hacia atrás para obtener la solución.

Aunque la técnica de Gauss-Jordan y la eliminación de Gauss podrían parecer casi idénticas, la primera requiere más trabajo.

Así, la técnica de Gauss-Jordan involucra aproximadamente 50% más operaciones que la eliminación de Gauss, por lo tanto, la eliminación de Gauss es el método de eliminación sencilla que se prefiere para obtener las soluciones de ecuaciones algebraicas lineales. Sin embargo, una de las razones principales por las que se ha introducido la técnica de Gauss-Jordan, es que aún se utiliza tanto en la ingeniería como en ciertos algoritmos numéricos

2.1. Multiplicación de matrices

Problema 9.3

Tres matrices se definen como:

$$[A] = \begin{bmatrix} 1 & 6 \\ 3 & 10 \\ 7 & 4 \end{bmatrix} \quad [B] = \begin{bmatrix} 1 & 3 \\ 0.5 & 2 \end{bmatrix} \quad [C] = \begin{bmatrix} 2 & -2 \\ -3 & 1 \end{bmatrix}$$

1. Ejecute todas las multiplicaciones que sea posible calcular entre parejas de las matrices
2. Utilice el método del cuadro para justificar por qué no se puede multiplicar a las demás parejas
3. Emplee los resultados del número 1) para ilustrar por qué es importante el orden de la multiplicación.

1. Multiplicar las matrices posibles

```

1 #definir matrices
2 A = [[1,6],[3,10],[7,4]]
3 B = [[1,3],[0.5,2]]
4 C= [[2,-2],[-3,1]]
5
6 def multiMatriz(A, B):
7
8     rowsA, colsA = len(A), len(A[0])
9     rowsB, colsB = len(B), len(B[0])
10
11
12 #condicion para operar
13 if colsA != rowsB:
14     exit("Dimensiones incorrectas")
15
16 C = [[0 for row in range(colsB)] for col in range(rowsA)]
17
18 #imprimir matriz resultante
19 for i in range(rowsA):
20     for j in range(colsB):
21         for k in range(colsA):
22             C[i][j] += A[i][k]*B[k][j]
23 print(C)
24
25 multiMatriz(A,B)
26 multiMatriz(A,C)
27 multiMatriz(B,C)
28 multiMatriz(C,B)

```

2. Como sabemos del álgebra lineal, la multiplicación de matrices está condicionada para matrices de la forma $[A] = m \times n$ por $[B] = n \times l$ donde debemos notar que el número de filas de A y el numero de columnas de B son iguales.
3. Debido a lo explicado en el punto anterior, se dice que en general, las multiplicaciones de $A \cdot B$ y $B \cdot A$ no son iguales. Es decir, no cumplen la propiedad de conmutatividad para el producto.

2.2. Sistema de ecuaciones lineales (1)

Problema 9.5

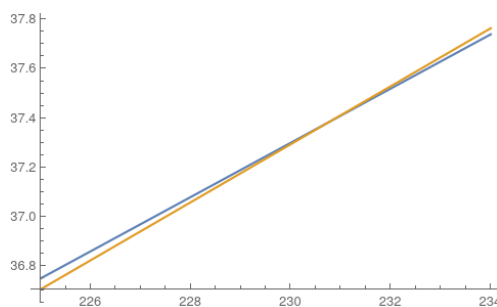
Dado el sistema de ecuaciones:

$$-1.1x_1 + 10x_2 = 120$$

$$-2x_1 + 17x_2 = 174$$

1. Resuélvalo gráficamente y compruebe el resultado con la sustitución en las ecuaciones.
2. Sobre la base de la solución gráfica, ¿qué se espera con respecto de la condición del sistema?
3. Calcule el determinante
4. Resuelva mediante eliminación de incógnitas

1. De la gráfica podemos observar que las soluciones X_2 y X_1 del sistema corresponde a la ordenada y y a la abscisa x del punto donde se intersectan ambas rectas. Esto sería:



$$X_1 \approx 230.84$$

$$X_2 \approx 37.39$$

Ahora sustituimos estos resultados en el sistema para comprobar su veracidad

$$-1.1(230.84) + 10(37.39) = 119.976$$

$$-2(230.84) + 17(37.39) = 173.95$$

2. Como vemos en la gráfica, las rectas se tocan en un punto muy alejado del alejado del 0, esto por sus pendientes poco pronunciadas, esto genera lo que se denomina como un sistema mal condicionado que genera problemas para encontrar sus raíces exactas tanto numéricamente como gráficamente.

3. Calculo del determinante: 1.3

```

1 #def matriz
2 M = [[-1.1,10],
3      [-2,17]]
4
5 def Determinante(M): #determinante para matriz 2x2
6
7     Det = M[0][0]*M[1][1] - M[0][1]*M[1][0]
8
9     print("El determinante de la matriz ", M, " es ",
10          round(Det,4))
11
12 Determinante(M)

```

4. Eliminación de incógnitas

```

1 import numpy
2
3 m=int(input("Numero de renglones: "))
4
5 n=int(input("Numero de columnas: "))
6
7 matrix = numpy.zeros((m,n))
8
9 vector= numpy.zeros((n))
10
11 x=numpy.zeros((m))
12
13 print("Introduce la matriz de coeficientes y el vector solucion")
14
15 for r in range(0,m):
16
17     for c in range(0,n):
18         matrix[(r),(c)]=(input("Elemento M["+str(r+1)+" , "
19                                +str(c+1)+" ]:"))
20
21     vector[(r)]=(input("b["+str(r+1)+" ]: "))
22 print(matrix)
23
24 x2 = (matrix[0][0]*vector[1]-matrix[1][0]*vector[0])/
25 (matrix[0][0]*matrix[1][1]-matrix[0][1]*matrix[1][0])
26 x1 = (matrix[1][1]*vector[0]-matrix[0][1]*vector[1])/
27 (matrix[0][0]*matrix[1][1]-matrix[0][1]*matrix[1][0])
28
29 print("Soluciones al sistema\n\n ")
30 print("X_1: ", x1, "\n")
31 print("X_2: ", x2, "\n")

```

2.3. Sistema de ecuaciones lineales (2)

Problema 9.7

Dadas las ecuaciones

$$\begin{aligned}0.5x_1 - x_2 &= -9.5 \\ 1.02x_1 - 2x_2 &= -18.8\end{aligned}$$

1. Resuelva en forma gráfica.
 2. Calcule el determinante.
 3. Con base en los incisos a) y b), ¿qué es de esperarse con respecto de la condición del sistema?
 4. Resuelva por medio de la eliminación de incógnitas.
 5. Resuelva otra vez, pero modifique ligeramente el elemento a_{11} a 0.52. Interprete sus resultados.
1. De la intersección de las rectas, tenemos que la ordenada corresponde a X_2 y la abscisa a X_1 . Entonces

$$\begin{aligned}X_1 &\approx -1889.99 \\ X_2 &\approx -954.499\end{aligned}$$

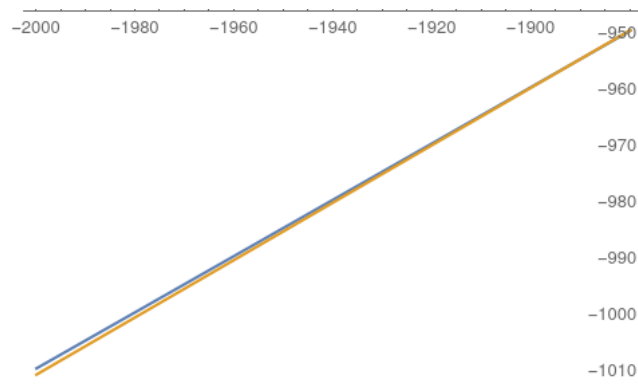


Figura 2.1: Plot de las gráficas en el punto de intersección

2. El determinante del sistema es 0.02

```
1 #def matriz
2 M = [[0.5, -1],
3      [1.02, -2]]
4
5 def Determinante(M): #determinante para matriz 2x2
6
7     Det = M[0][0]*M[1][1] - M[0][1]*M[1][0]
8
9     print("El determinante de la matriz ", M, " es ",
10          round(Det,4))
11
12 Determinante(M)
```

3. El sistema es mal condicionado, convergen sus valores muy alejados del (0,0) por lo que computacionalmente, el resultado va a depender de la cantidad de cifras significativas usadas.

4. Eliminación de incógnitas

```

1 import numpy
2
3 m=int(input("Numero de renglones: "))
4
5 n=int(input("Numero de columnas: "))
6
7 matrix = numpy.zeros((m,n))
8
9 vector= numpy.zeros((n))
10
11 x=numpy.zeros((m))
12
13 print("Introduce la matriz de coeficientes y el vector solucion")
14
15 for r in range(0,m):
16
17     for c in range(0,n):
18         matrix[(r),(c)]=(input("Elemento M["+str(r+1)+", "
19                               +str(c+1)+"]:"))
20
21     vector[(r)]=(input("b["+str(r+1)+"]:"))
22 print(matrix)
23
24 x2 = (matrix[0][0]*vector[1]-matrix[1][0]*vector[0])/
25 (matrix[0][0]*matrix[1][1]-matrix[0][1]*matrix[1][0])
26 x1 = (matrix[1][1]*vector[0]-matrix[0][1]*vector[1])/
27 (matrix[0][0]*matrix[1][1]-matrix[0][1]*matrix[1][0])
28
29 print("Soluciones al sistema\n\n ")
30 print("X_1: ", x1, "\n")
31 print("X_2: ", x2, "\n")

```

5. El determinante con esta modificación resulta: -0.02. Gráficamente podemos ver que la solución, de estar en el cuadrante 3, ahora se localiza en el cuadrante 1.

```

1 #def matriz
2 M = [[0.52, -1],
3      [1.02, -2]]
4
5 def Determinante(M): #determinante para matriz 2x2
6
7     Det = M[0][0]*M[1][1] - M[0][1]*M[1][0]
8
9     print("El determinante de la matriz ", M, " es ",
10          round(Det,4))
11
12 Determinante(M)

```

2.4. Sistema de ecuaciones lineales (3)

Problema 9.9

Use la eliminación de Gauss para resolver:

$$8x_1 + 2x_2 - 2x_3 = -2$$

$$10x_1 + 2x_2 + 4x_3 = 4$$

$$12x_1 + 2x_2 + 2x_3 = 6$$

Emplee pivoteo parcial y compruebe las respuestas sustituyéndolas en las ecuaciones originales

1. Pivoteo parcial

```

1 import numpy as np
2
3 A = np.array([[8, 2, -2], [10, 2, 4], [12, 2, 2]])
4
5 B = np.array([[ -2], [4], [6]])
6
7 # Evitar truncamiento en operaciones
8 A = np.array(A, dtype=float)
9
10 # Matriz aumentada
11 AB = np.concatenate((A, B), axis=1)
12 AB0 = np.copy(AB)
13
14 # Pivoteo parcial por filas
15 tamaño = np.shape(AB)
16 n = tamaño[0]
17 m = tamaño[1]
18
19
20 for i in range(0, n-1, 1):
21     # columna desde diagonal i en adelante
22     columna = abs(AB[i:, i])
23     dondemax = np.argmax(columna)
24
25     if (dondemax != 0):
26         # intercambia filas
27         temp = np.copy(AB[i, :])
28         AB[i, :] = AB[dondemax+i, :]
29         AB[dondemax+i, :] = temp
30
31 AB1 = np.copy(AB)
32

```



```
33 # eliminacion hacia adelante
34 for i in range(0,n-1,1):
35     pivote = AB[i,i]
36     adelante = i + 1
37     for k in range(adelante,n,1):
38         factor = AB[k,i]/pivote
39         AB[k,:] = AB[k,:] - AB[i,:]*factor
40 AB2 = np.copy(AB)
41
42
43 print(" Matriz aumentada: ",AB0)
44
45 print(" Solucion del sistema: ", X)
```

2.5. Sistema de ecuaciones lineales (4)

Problema 9.13

Resuelva el sistema:

$$\begin{aligned}x_1 + x_2 - x_3 &= -3 \\6x_1 + 2x_2 + 2x_3 &= 2 \\-3x_1 + 4x_2 + x_3 &= 1\end{aligned}$$

por medio de

1. eliminación de Gauss simple
2. eliminación de Gauss con pivoteo parcial
3. método de Gauss-Jordan sin pivoteo parcial

1. Gauss simple

```

1 import numpy
2
3 m=int(input("Numero de renglones:"))
4
5 n=int(input("Numero de columnas:"))
6
7 matrix = numpy.zeros((m,n))
8
9 vector= numpy.zeros((n))
10
11 x=numpy.zeros((m))
12
13 print("Ingresa la matriz y el vector solucion")
14
15 for r in range(0,m):
16
17     for c in range(0,n):
18
19         matrix[(r),(c)]=(input("Elemento
20 M["+str(r+1)+"," +str(c+1)+"] :"))
21
22     vector[(r)]=(input("b["+str(r+1)+"] : "))
23
24 print(matrix)
25
26
27 for k in range (0,m):
28

```

```
29     for r in range(k+1,m):
30
31         factor=(matrix[r,k]/matrix[k,k])
32
33         vector[r]=vector[r]-(factor*vector[k])
34
35         for c in range(0,n):
36
37             matrix[r,c]=matrix[r,c]-(factor*matrix[k,c])
38
39
40 #sustitucion hacia atras
41
42 x[m-1]=vector[m-1]/matrix[m-1,m-1]
43
44 print(x[m-1])
45
46
47 for r in range(m-2,-1,-1):
48     suma=0
49
50     for c in range(0,n):
51         suma=suma+matrix[r,c]*x[c]
52
53     x[r]=(vector[r]-suma)/matrix[r,r]
54
55
56 print("Matriz resultante", matrix)
57
58 print("Vector resultante", vector)
59
60 print("Soluciones al sistema: ", x)
```

2. Gauss-Jordan

```

1 import numpy as np
2
3 A = np.array([[1,1,1],[6,2,2],[-3,4,1]])
4
5 B = np.array([[ -3],[2],[1]])
6
7 # Evitar truncamiento en operaciones
8 A = np.array(A,dtype=float)
9
10 # Matriz aumentada
11 AB = np.concatenate((A,B),axis=1)
12 AB0 = np.copy(AB)
13
14 # Pivoteo parcial por filas
15 tamaño = np.shape(AB)
16 n = tamaño[0]
17 m = tamaño[1]
18
19
20 for i in range(0,n-1,1):
21     # columna desde diagonal i en adelante
22     columna = abs(AB[i:,i])
23     dondemax = np.argmax(columna)
24
25     if (dondemax !=0):
26         # intercambia filas
27         temp = np.copy(AB[i,:])
28         AB[i,:] = AB[dondemax+i,:]
29         AB[dondemax+i,:] = temp
30
31 AB1 = np.copy(AB)
32
33 # eliminacion hacia adelante
34 for i in range(0,n-1,1):
35     pivote = AB[i,i]
36     adelante = i + 1
37     for k in range(adelante,n,1):
38         factor = AB[k,i]/pivote
39         AB[k,:] = AB[k,:] - AB[i,:]*factor
40 AB2 = np.copy(AB)
41
42
43 print("Matriz aumentada: ",AB0)
44
45 print("Solucion del sistema: ", X)

```

2.6. Masas atadas por resortes

Problema 9.19

Tres masas están suspendidas verticalmente por una serie de resortes idénticos donde la masa 1 está en la parte superior y la masa 3 está en la parte inferior. Si $g = 9.81 \text{ m/s}^2$, $m_1 = 2 \text{ kg}$, $m_2 = 3 \text{ kg}$, $m_3 = 2.5 \text{ kg}$ y las $k = 10 \text{ kg/s}^2$, despeje los desplazamientos x .

De la ley de Hooke, $F = -kx$, planteamos una ecuación para cada masa:

$$\begin{aligned} F &= -kx_i \\ m_i g &= -kx_i \end{aligned}$$

En forma matricial, ponemos la fuerza en la parte aumentada, quedaría de la siguiente manera:

$$[A] = \left[\begin{array}{ccc|c} -10 & 0 & 0 & 19.62 \\ -10 & -10 & 0 & 29.43 \\ -10 & -10 & -10 & 24.52 \end{array} \right]$$

Donde A, como podemos ver, es una matriz triangular inferior.

```

1 import numpy
2
3 m=int(input("Numero de renglones:"))
4
5 n=int(input("Numero de columnas:"))
6
7 matrix = numpy.zeros((m,n))
8
9 vector= numpy.zeros((n))
10
11 x=numpy.zeros((m))
12
13 print("Ingresa la matriz y el vector solucion")
14
15 for r in range(0,m):
16
17     for c in range(0,n):
18
19         matrix[(r),(c)]=(input("Elemento
20 M["+str(r+1)+", "+str(c+1)+"]:"))
21
22     vector[(r)]=(input("b["+str(r+1)+"]:"))
23
24 print(matrix)
25
26
27 for k in range (0,m):
28

```

```
29     for r in range(k+1,m):
30
31         factor=(matrix[r,k]/matrix[k,k])
32
33         vector[r]=vector[r]-(factor*vector[k])
34
35         for c in range(0,n):
36
37             matrix[r,c]=matrix[r,c]-(factor*matrix[k,c])
38
39
40 #sustitucion hacia atras
41
42 x[m-1]=vector[m-1]/matrix[m-1,m-1]
43
44 print(x[m-1])
45
46
47 for r in range(m-2,-1,-1):
48     suma=0
49
50     for c in range(0,n):
51         suma=suma+matrix[r,c]*x[c]
52
53     x[r]=(vector[r]-suma)/matrix[r,r]
54
55
56 print("Matriz resultante", matrix)
57
58 print("Vector resultante", vector)
59
60 print("Soluciones al sistema: ", x)
```

3.1. Anexo A: Evidencia del funcionamiento de los código reportados

Ejercicio 9.3

Figura 3.1: Evidencia del funcionamiento

```
[[4.0, 15], [8.0, 29], [9.0, 29]]  
[[-16, 4], [-24, 4], [2, -10]]  
[[-7, 1], [-5.0, 1.0]]  
[[1.0, 2], [-2.5, -7]]
```

Ejercicio 9.5

Figura 3.2: Evidencia del funcionamiento

El determinante de la matriz $\begin{bmatrix} -1.1 & 10 \\ -2 & 17 \end{bmatrix}$ es 1.3

Figura 3.3: Evidencia del funcionamiento

```
Numero de renglones: 2
Numero de columnas: 2
Introduce la matriz de coeficientes y el vector solucion
Elemento M[1,1]:-1.1
Elemento M[1,2]:10
b[1]: 120
Elemento M[2,1]:-2
Elemento M[2,2]:17
b[2]: 174
[[ -1.1  10. ]
 [ -2.   17. ]]
Soluciones al sistema

X_1:  230.769230769
X_2:  37.3846153846
```


Ejercicio 9.7

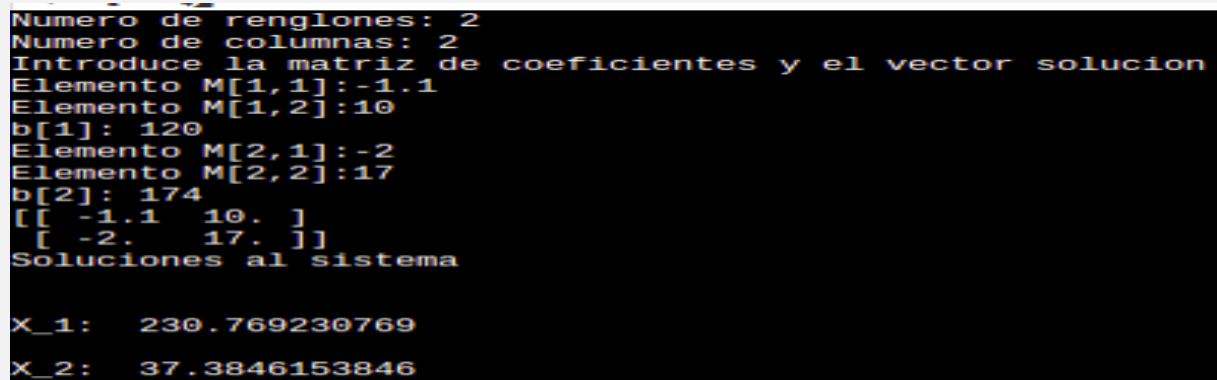
Figura 3.4: Evidencia del funcionamiento



El determinante de la matriz $\begin{bmatrix} 0.5 & -1 \\ 1.02 & -2 \end{bmatrix}$ es 0.02

Ejercicio 9.9

Figura 3.5: Evidencia del funcionamiento



```
Numero de renglones: 2
Numero de columnas: 2
Introduce la matriz de coeficientes y el vector solucion
Elemento M[1,1]:-1.1
Elemento M[1,2]:10
b[1]: 120
Elemento M[2,1]:-2
Elemento M[2,2]:17
b[2]: 174
[[ -1.1 10. ]
 [ -2. 17. ]]
Soluciones al sistema

X_1: 230.769230769
X_2: 37.3846153846
```

Ejercicio 9.13

Figura 3.6: Evidencia del funcionamiento

```

Elemento M[1,2]:1
Elemento M[1,3]:-1
b[1]: -3
Elemento M[2,1]:6
Elemento M[2,2]:2
Elemento M[2,3]:2
b[2]: 2
Elemento M[3,1]:-3
Elemento M[3,2]:4
Elemento M[3,3]:1
b[3]: 1
[[ 1.  1. -1.]
 [ 6.  2.  2.]
 [-3.  4.  1.]]
Soluciones al sistema

X_1: 2.0
X_2: -5.0

```

Ejercicio 9.19

Figura 3.7: Evidencia del funcionamiento

```

ElementoM[2,1]:-10
ElementoM[2,2]:-10
ElementoM[2,3]:0
b[2]: 29.43
ElementoM[3,1]:-10
ElementoM[3,2]:-10
ElementoM[3,3]:-10
b[3]: 24.52
[[-10.  0.  0.]
 [-10. -10.  0.]
 [-10. -10. -10.]]
0.491
Matriz resultante [[-10.  0.  0.]
 [ 0. -10.  0.]
 [ 0.  0. -10.]]
Vector resultante [ 19.62  9.81 -4.91]
Soluciones al sistema: [-1.962 -0.981 0.491]

```

CAPÍTULO 4

BIBLIOGRAFÍA

1. Chapra, S. C., Canale, R. P. (2011). *Numerical methods for engineers* (Vol. 1221). New York: McGraw-hill.