**Instituto Tecnológico Metropolitano ITM**

**Facultad de ingenierías**



**Optimización Matemática**

**Profesor: Mauricio Vásquez**

**Presentado Por:**

**Juan Pablo Polanco Rúa**

**Oscar David Doria García**

**Robert Arley Carmona Velásquez**

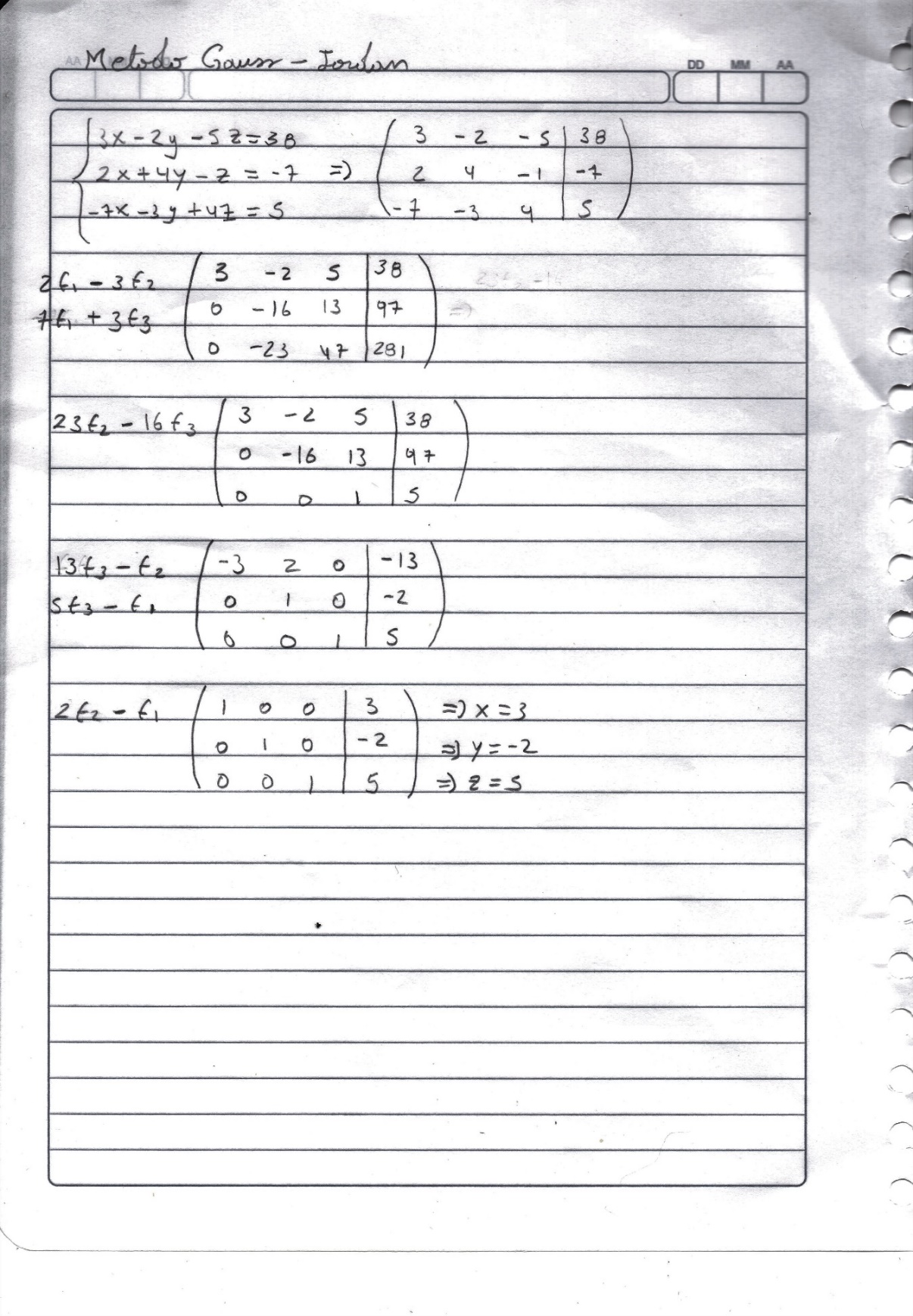
**2020**

**Gauss-Jordan**

* **Descripción matemática**

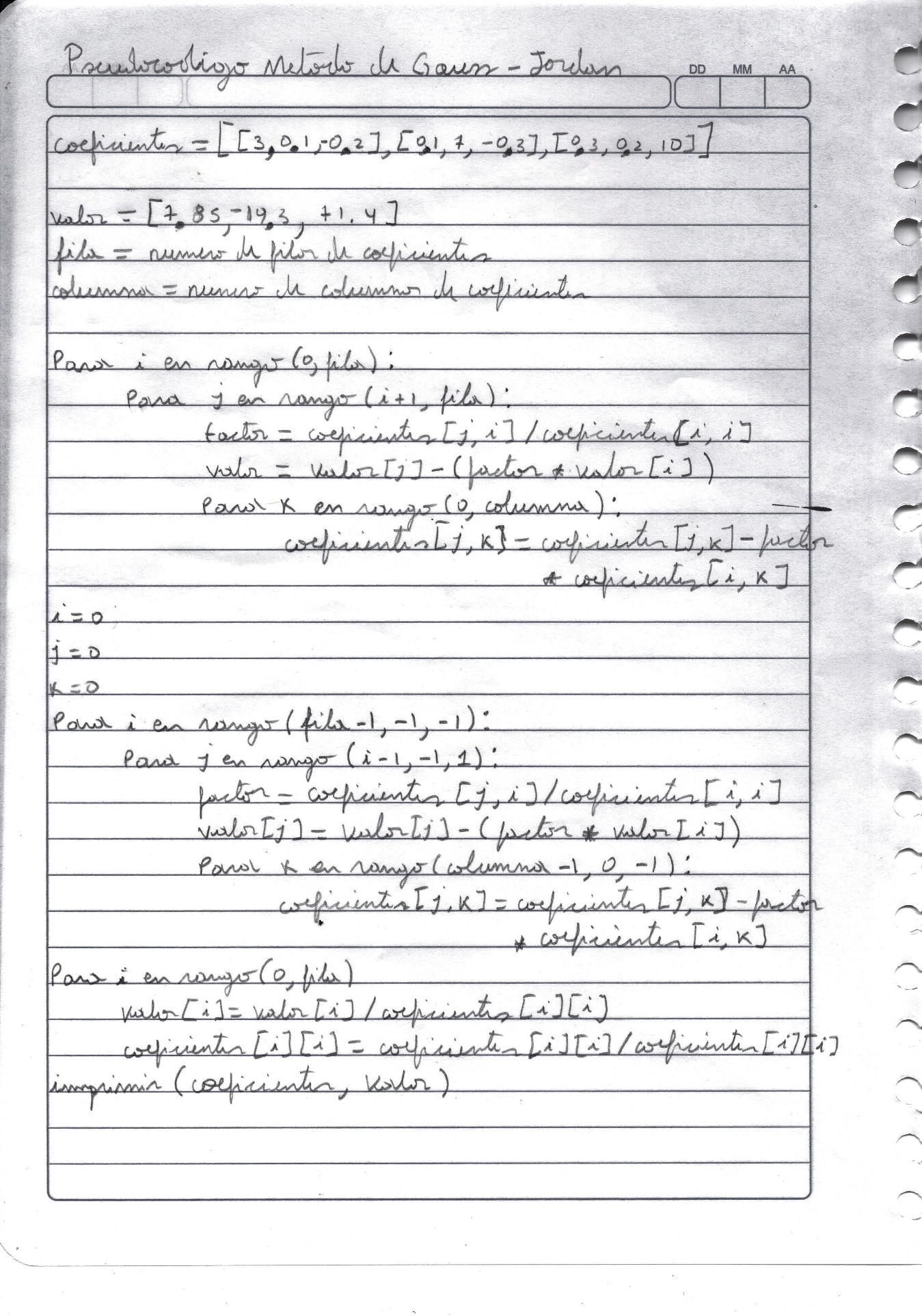
El método de Gauss-Jordan utiliza operaciones con matrices para resolver sistemas de ecuaciones de n número de variables. Para aplicar este método solo hay que recordar que cada operación que se realice se aplicara a toda la fila o a toda la columna en su caso. El objetivo de este método es tratar de convertir la parte de la matriz donde están los coeficientes de las variables en una matriz identidad. Esto se logra mediante simples operaciones de suma, resta y multiplicación.

El procedimiento es el siguiente:



Como puede verse el método Gauss-Jordan es una herramienta útil en la resolución de este tipo de problemas y actualmente existen programas matemáticos que lo utilizan para una gran variedad de cálculos en una gran variedad de áreas, tanto científicas como socioeconómicas.

* **Pseudocódigo Gauss-Jordan**



* **Prueba de escritorio**

|  |  |
| --- | --- |
| coeficientes | [[3,-0.1,-0.2], [0.1,7,-0.3], [0.3,-0.2,10]] |
| columna | 3 |
| factor | 0 |
| fila | 3 |
| i | 0 |
| j | 0 |
| k | 0 |
| valor | [7.85,-19.3,71.4] |

|  |  |
| --- | --- |
| coeficientes | [[3,-0.1,-0.2], [0.1,7,-0.3], [0.3,-0.2,10]] |
| columna | 3 |
| factor | 0.03333333333333333 |
| fila | 3 |
| i | 0 |
| j | 1 |
| k | 0 |
| valor | [7.85,-19.561667,71.4] |

|  |  |
| --- | --- |
| coeficientes | [[3,-0.1,-0.2], [0,7,-0.3], [0.3,-0.2,10]] |
| columna | 3 |
| factor | 0.03333333333333333 |
| fila | 3 |
| i | 0 |
| j | 1 |
| k | 0 |
| valor | [7.85,-19.561667,71.4] |

|  |  |
| --- | --- |
| coeficientes | [[3,-0.1,-0.2], [0,7.00333,-0.3], [0.3,-0.2,10]] |
| columna | 3 |
| factor | 0.03333333333333333 |
| fila | 3 |
| i | 0 |
| j | 1 |
| k | 1 |
| valor | [7.85,-19.561667,71.4] |

|  |  |
| --- | --- |
| coeficientes | [[3,-0.1,-0.2], [0,7.00333,-0.29333], [0.3,-0.2,10]] |
| columna | 3 |
| factor | 0.03333333333333333 |
| fila | 3 |
| i | 0 |
| j | 1 |
| k | 2 |
| valor | [7.85,-19.561667,71.4] |

|  |  |
| --- | --- |
| coeficientes | [[3,-0.1,-0.2], [0,7.00333,-0.29333], [0.3,-0.2,10]] |
| columna | 3 |
| factor | 0.09999999999999999 |
| fila | 3 |
| i | 0 |
| j | 2 |
| k | 2 |
| valor | [7.85,-19.561667,70.615] |

|  |  |
| --- | --- |
| coeficientes | [[3,-0.1,-0.2], [0,7.00333,-0.29333], [0,-0.2,10]] |
| columna | 3 |
| factor | 0.09999999999999999 |
| fila | 3 |
| i | 0 |
| j | 2 |
| k | 0 |
| valor | [7.85,-19.561667,70.615] |

|  |  |
| --- | --- |
| coeficientes | [[3,-0.1,-0.2], [0,7.00333,-0.29333], [0,-0.19,10]] |
| columna | 3 |
| factor | 0.09999999999999999 |
| fila | 3 |
| i | 0 |
| j | 2 |
| k | 1 |
| valor | [7.85,-19.561667,70.615] |

|  |  |
| --- | --- |
| coeficientes | [[3,-0.1,-0.2], [0,7.00333,-0.29333], [0,-0.19,10.02]] |
| columna | 3 |
| factor | 0.09999999999999999 |
| fila | 3 |
| i | 0 |
| j | 2 |
| k | 2 |
| valor | [7.85,-19.561667,70.615] |

|  |  |
| --- | --- |
| coeficientes | [[3,-0.1,-0.2], [0,7.00333,-0.29333], [0,-0.19,10.02]] |
| columna | 3 |
| factor | -0.027129938124702525 |
| fila | 3 |
| i | 1 |
| j | 2 |
| k | 2 |
| valor | [7.85,-19.561667,70.08429319] |

|  |  |
| --- | --- |
| coeficientes | [[3,-0.1,-0.2], [0,7.00333,-0.29333], [0,0,10.02]] |
| columna | 3 |
| factor | -0.027129938124702525 |
| fila | 3 |
| i | 1 |
| j | 2 |
| k | 0 |
| valor | [7.85,-19.561667,70.08429319] |

|  |  |
| --- | --- |
| coeficientes | [[3,-0.1,-0.2], [0,7.00333,-0.29333], [0,0,10.12]] |
| columna | 3 |
| factor | -0.027129938124702525 |
| fila | 3 |
| i | 1 |
| j | 2 |
| k | 2 |
| valor | [7.85,-19.561667,70.08429319] |

|  |  |
| --- | --- |
| coeficientes | [[3,-0.1,-0.2], [0,7.00333,-0.29333], [0,0,10.12]] |
| columna | 3 |
| factor | -0.027129938124702525 |
| fila | 3 |
| i | 2 |
| j | 2 |
| k | 2 |
| valor | [7.85,-19.561667,70.08429319] |

|  |  |
| --- | --- |
| coeficientes | [[3,-0.1,-0.2], [0,7.00333,0], [0,0,10.12]] |
| columna | 3 |
| factor | -0.029298052955428894 |
| fila | 3 |
| i | 2 |
| j | 1 |
| k | 2 |
| valor | [7.85,-17.5083333,70.08429319] |

|  |  |
| --- | --- |
| coeficientes | [[3,-0.1,-0.2], [0,7.00333,0], [0,0,10.12]] |
| columna | 3 |
| factor | -0.029298052955428894 |
| fila | 3 |
| i | 2 |
| j | 1 |
| k | 1 |
| valor | [7.85,-17.5083333,70.08429319] |

|  |  |
| --- | --- |
| coeficientes | [[3,-0.1,-0.2], [0,7.00333,0], [0,0,10.12]] |
| columna | 3 |
| factor | -0.0199775945196883338 |
| fila | 3 |
| i | 2 |
| j | 0 |
| k | 2 |
| valor | [9.25,-17.5083333,70.08429319] |

|  |  |
| --- | --- |
| coeficientes | [[3,-0.1,0], [0,7.00333,0], [0,0,10.12]] |
| columna | 3 |
| factor | -0.0199775945196883338 |
| fila | 3 |
| i | 2 |
| j | 0 |
| k | 2 |
| valor | [9.25,-17.5083333,70.08429319] |

|  |  |
| --- | --- |
| coeficientes | [[3,-0.1,0], [0,7.00333,0], [0,0,10.12]] |
| columna | 3 |
| factor | -0.0199775945196883338 |
| fila | 3 |
| i | 2 |
| j | 0 |
| k | 1 |
| valor | [9.25,-17.5083333,70.08429319] |

|  |  |
| --- | --- |
| coeficientes | [[3,-0.1,0], [0,7.00333,0], [0,0,10.12]] |
| columna | 3 |
| factor | -0.014278914802475014 |
| fila | 3 |
| i | 1 |
| j | 0 |
| k | 2 |
| valor | [9,-17.5083333,70.08429319] |

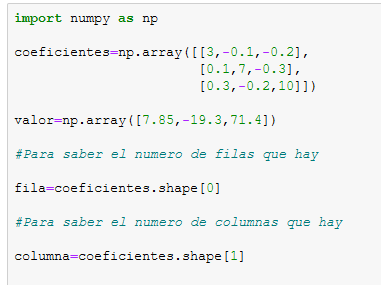
|  |  |
| --- | --- |
| coeficientes | [[3,0,0], [0,7.00333,0], [0,0,10.12]] |
| columna | 3 |
| factor | -0.014278914802475014 |
| fila | 3 |
| i | 1 |
| j | 0 |
| k | 1 |
| valor | [9,-17.5083333,70.08429319] |

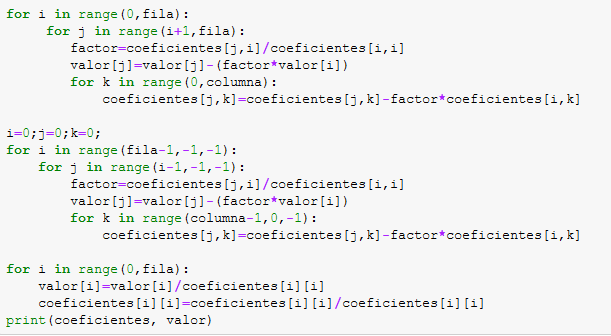
|  |  |
| --- | --- |
| coeficientes | [[3,0,0], [0,7.00333,0], [0,0,10.12]] |
| columna | 3 |
| factor | -0.014278914802475014 |
| fila | 3 |
| i | 0 |
| j | 0 |
| k | 1 |
| valor | [9,-17.5083333,70.08429319] |

|  |  |
| --- | --- |
| coeficientes | [[1,0,0], [0,7.00333,0], [0,0,10.12]] |
| columna | 3 |
| factor | -0.014278914802475014 |
| fila | 3 |
| i | 0 |
| j | 0 |
| k | 1 |
| valor | [3,-17.5083333,70.08429319] |

|  |  |
| --- | --- |
| coeficientes | [[1,0,0],[0,1,0], [0,0,10.12]] |
| columna | 3 |
| factor | -0.014278914802475014 |
| fila | 3 |
| i | 1 |
| j | 0 |
| k | 1 |
| valor | [3,-2.5,70.08429319] |
| coeficientes | [[1,0,0]  ,[0,1,0],  [0,0,1]] |
| columna | 3 |
| factor | -0.014278914802475014 |
| fila | 3 |
| i | 2 |
| j | 0 |
| k | 1 |
| valor | [3,-2.5,7] |

* **Código.**





**Interpolación cuadrática**

* **Descripción matemática.**

Se denomina interpolación cuadrática a la que se realiza cuando tres puntos conocidos, y no alineados (, ), (, ) y (, ), se ajustan mediante la parábola que pasa por ellos. El polinomio que se ajusta a esos puntos es de la forma P(x) , conocido también con el nombre de función cuadrática, cuya gráfica es una parábola.

Los coeficientes a, b y c desconocidos son las soluciones del sistema:

cuyas ecuaciones se obtienen imponiendo la condición de que P(x) pase por , y .

Por tres puntos no alineados siempre pasa una parábola. Por tanto, este sistema será compatible determinado. Si los puntos estuviesen alineados, la incógnita "a" (coeficiente de x) tomaría el valor cero: la parábola degeneraría en una recta.

Ejemplo: Hallar la función cuadrática de interpolación correspondiente a los valores:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| X | | 1 | 3 | 4 | 5 |
| Y | 4 | | 9 | ? | 18 |

**Determina su valor cuando x = 4.**

La ecuación general de una parábola es de la forma

La ecuación de la parábola es

Si x = 4 la función toma el valor y = 13

(3,10),(2,3),(-1,6) f(x) = a

|  |
| --- |
| a = 2 |

|  |
| --- |
|  |

|  |
| --- |
|  |

* **Pseudocódigo.**

# Cuando ya se tienen los valores

Puntos = array ([[3,10]

,[2,3]

,[-1,6]])

#Cuando se van a ingresar los valores por pantalla

Para i en rango (0,3)

Puntos [i][0] = flotante (ingresar (“valor para x” +str(i-1))

Puntos [i][1] = flotante (ingresar (“valor para y” +str(i+1)))

Coeficientes = zeros ((3,4))

X, y, z = symbols (“x y z”)

Para i en rango (0,3):

coeficientes[i][0]=puntos[i][0]\*\*2

coeficientes[i][1]=puntos[i][0]

coeficientes[i][2]=1

coeficientes[i][3]=puntos[i][1]

sistema=Matrix(coeficientes)

r=solve\_linear\_system (sistema, x, y, z)

funcion = r.get(x)\*x\*\*2+r.get(y)\*x+r.get(z)

* **Prueba de Escritorio.**

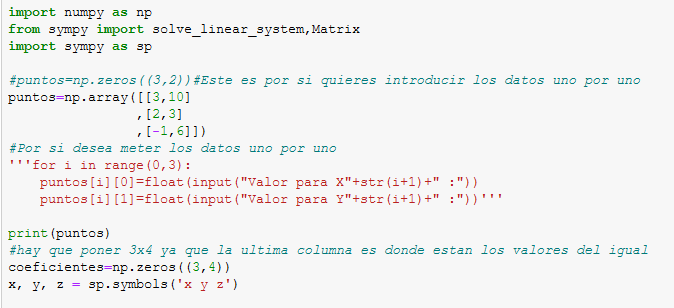
|  |  |
| --- | --- |
| Puntos | [[3,10], [2,3] , [-1,6]] |
| Coeficientes | [[0,0,0,0], [0,0,0,0], [0,0,0,0], [0,0,0,0]] |
| i | 0 |

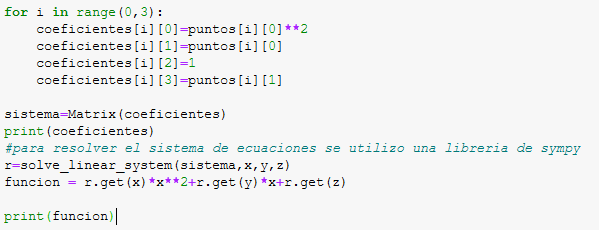
|  |  |
| --- | --- |
| Puntos | [[3,10], [2,3] , [-1,6]] |
| Coeficientes | [[9,3,1,10], [0,0,0,0], [0,0,0,0], [0,0,0,0]] |
| i | 0 |

|  |  |
| --- | --- |
| Puntos | [[3,10], [2,3] , [-1,6]] |
| Coeficientes | [[9,3,1,10], [4,2,1,3], [0,0,0,0], [0,0,0,0]] |
| i | 1 |

|  |  |
| --- | --- |
| Puntos | [[3,10], [2,3] , [-1,6]] |
| Coeficientes | [[9,3,1,10], [4,2,1,3], [1,-1,1,6]] |
| i | 2 |

**Código**





**Gauss-Seidel**

* **Descripción matemática.**

El Método de Gauss-Seidel consiste en hacer iteraciones a partir de un vector inicial para encontrar los valores de las incógnitas hasta llegar a una tolerancia deseada, cada vez que se desee encontrar un nuevo valor de una xi, se usan los valores anteriores de las x, y también utiliza valores actuales de las x encontradas antes (desde x0 hasta xi-1).

La ecuación es la siguiente:

Supongamos un sistema de ecuaciones 3x3, si los elementos de la diagonal no son todos cero, la primera ecuación se puede resolver para X1, la segunda para X2 y la tercera para X3

(1)

Y obtenemos:

(2)

(3)

Para iniciar, comenzamos asignándole los valores iniciales a las X, podemos asignarlos como ceros. Sustituimos en la ecuación (1) y obtenemos , este nuevo X1  y X3 que por el momento es igual a 0 los sustituimos en la ecuación (2) obteniendo un nuevo X2 repetimos en la ecuación (3) remplazando el nuevo X2  y X1, Con este nuevo valor de X3 repetimos el proceso esta vez con los nuevos valores obtenidos hasta que la solución converja suficientemente cerca a los valores verdaderos.

Calculo del error:

* **Pseudocódigo**

m = lea ‘¿ingrese número de filas?’

n = lea ‘¿ingrese número de filas?’

matrix = np.zeros((m,n))

x = np.zeros ((m))

comp = np.zeros((m))

error= []

para r desde 0 hasta m

para c desde 0 hasta n

matrix[r,c]= lea “ingrese valor de la posición ”, r+1 ,c+1

vector[r]= lea ‘ingrese solución ’, r

itera= lea ‘ingrese el número máximo de iteraciones’

tol = lea ‘ingrese la tolerancia deseada’

k=0

mientras k<itera

suma=0

k=k+1

para r desde 0 hasta m

suma=0

para c desde 0 hasta n

si c ≠ r entonces

suma=suma+matrix[r,c]\*x[c]

imprima x[r]

//comprobación

para r desde 0 hasta n

suma=0

para c desde 0 hasta n

suma = suma+matrix[r,c]\*x[c]

comp[r] = suma

dif= |comp[r]-vector[r]|

agregar dif al vector error

imprima error[r]

imprima “iteración: ” , k

si todos los elementos de error son <= tol:

detener el ciclo

* **Prueba de escritorio**

Se pide valores para m y n

m=3

n=3

Se crean las variables y se inicializan

Matrix = Se inicializa con todo en 0, sus dimensiones las definen m y n

X = Se inicializa con todo en 0, su dimensión la define m

Vector = Se inicializa con todo en 0, su dimensión la define n

Comp = Se inicializa con todo en 0, su dimensión la define m

Error=[]

Se ingresan la matriz de coeficientes y el vector solución

matrix =

vector =

se ingresa la tolerancia y el máximo de iteraciones

tol = 0.03

itera = 10

k=0

**entra al ciclo mientras k<itera**

suma=0

k=k+1

**entra al for r desde 0 hasta m=3**

suma=0

**entra al for c desde 0 hasta n=3**

verifica si (c!=r)

no entra porque c=0, r=0

**segunda vuelta ciclo for c=1**

verifica si (c!=r)

entra porque c=1,r=0

suma=0+1\*0

**tercer vuelta ciclo for c=2**

verifica si (c!=r)

entra porque c=2,r=0

suma=0+1\*0

**sale del ciclo for c**

x[0]= (2-0)/1 => x[0] = 2

**segunda vuelta ciclo for r=1**

suma=0

**entra al for c desde 0 hasta n=3**

verifica si (c!=r)

entra porque c=0, r=1

suma=0+2\*2 => suma=4

**segunda vuelta ciclo for c=1**

verifica si (c!=r)

no entra porque c=1,r=1

**tercer vuelta ciclo for c=2**

verifica si (c!=r)

entra porque c=2,r=1

suma=4+3\*0

**sale del ciclo for c**

x[1]=(11-4)/3 => x[1]=2.333

**tercer vuelta ciclo for r=2**

suma=0

**entra al for c desde 0 hasta n=3**

verifica si (c!=r)

entra porque c=0, r=2

suma=0+1\*2 => suma=2

**segunda vuelta ciclo for c=1**

verifica si (c!=r)

entra porque c=1,r=2

suma=2+(-5)\*2.333 =>suma=-9.666

**tercer vuelta ciclo for c=2**

verifica si (c!=r)

no entra porque c=2,r=2

**sale del ciclo for c**

x[2]=(29-(-9.666))/6 => x[2]=6.444

**sale del ciclo for r**

valores actuales de x =

ahora se hará la comprobación

**entra al ciclo for r desde 0 hasta n=3**

suma=0

**entra al ciclo for c desde 0 hasta m=3**

suma=0+1\*2 => suma=2

**segunda vuelta ciclo for c=1**

suma=2+1\*2.333 => suma=4.333

**tercera vuelta ciclo for c=2**

suma=4.333+1\*6.444 => suma=10.777

**sale del ciclo for c**

comp[0]=10.777

dif=|comp[0]-vector[0]| => dif=10.777- 2 => dif= 8.777

agregamos dif al vector error

**segunda vuelta ciclo for r=1**

suma=0

**entra al ciclo for c desde 0 hasta m=3**

suma=0+2\*2 => suma=4

**segunda vuelta ciclo for c=1**

suma=4+3\*2.333 => suma=11

**tercera vuelta ciclo for c=2**

suma=11 +5\*6.444 => suma=43.22

**sale del ciclo for c**

comp[1]=43.22

dif=|comp[1]-vector[1]| => dif=43.22 - 11 => dif= 32.22

agregamos dif al vector error

**tercer vuelta ciclo for r=2 1 -5 6**

suma=0

**entra al ciclo for c desde 0 hasta m=3**

suma=0+1\*2 => suma=2

**segunda vuelta ciclo for c=1**

suma=2+(-5)\*2.333 => suma=-9.666

**tercera vuelta ciclo for c=2**

suma=-9.666 +6\*6.444 => suma=28.974

**sale del ciclo for c**

comp[2]=28.974

dif=|comp[2]-vector[2]| => dif=28.974 - 29 => dif= 0.026

agregamos dif al vector error

primer iteración lista

verificamos que todos los errores del vector error sean menores a la tolerancia=0.03

tenemos

error= y tol=0.03

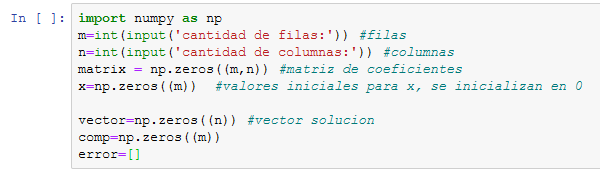
se debe cumplir que

8.777<=0.03 y 32.22<=0.03 y 0.026<=0.03

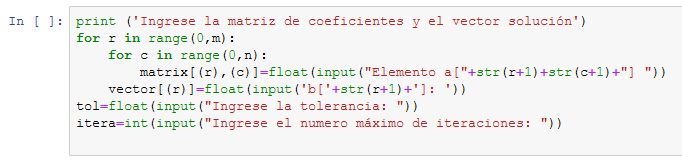
Vemos que se cumple solamente para el error en error [2], entonces se regresa al while y se repite todo el proceso hasta que se cumpla la condición anterior o se llegue al máximo de iteraciones.

**Código**

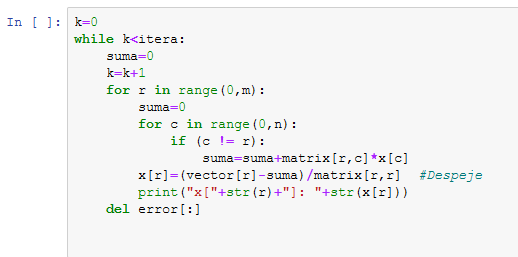
Inicialización de variables que se van a utilizar.



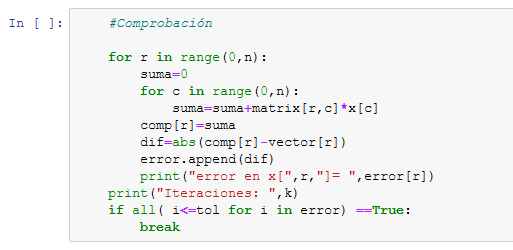
Ingreso de valores de la matriz de coeficientes y el vector solución



Inicio de las iteraciones e inicio del método de gauss-seidel



Inicio del ciclo para comprobación

****

**Referencias**

* CHAPRA, Steven C y Raymond P. Canale. Métodos Numéricos para Ingenieros. México: <https://ayudasingenieria.com/files/METODOS_NUMERICOS/chapra.pdf>