Національний технічний університет України «КПІ ім. Ігоря Сікорського»
Факультет Інформатики та Обчислювальної Техніки
Кафедра Обчислювальної Техніки

Лабораторна робота № 2

з дисципліни «Чисельні методи»

на тему

**«Розв'язання систем лінійних алгебраїчних рівнянь (СЛАР) прямими методами.
Звичайний метод Гауса та метод квадратних коренів»**

Виконав:
студент гр. ІП-93
Домінський Валентин

Викладач:
доц. Рибачук Л.В.

Київ – 2021

Зміст

1 Постановка задачі

Розв'язати систему рівнянь методом Гауса з кількістю значущих цифр $m = 6$.

$$\begin{bmatrix} 8.3 & 3.42 & 4.1 & 1.9 \\ 3.92 & 8.45 & 7.98 & 2.46 \\ 3.77 & 8.01 & 8.04 & 2.28 \\ 2.21 & 2.85 & 1.69 & 6.99 \end{bmatrix} \cdot X = \begin{bmatrix} -9.85 \\ 12.21 \\ 14.65 \\ -8.35 \end{bmatrix}$$

Вивести всі проміжні результати та розв'язок системи. Навести результат перевірки: вектор нев'язки $\boldsymbol{r} = \boldsymbol{b} - \boldsymbol{Ax}$, де $\boldsymbol{x}$ - отриманий розв'язок.

Розв'язати задану систему рівнянь за допомогою програмного забезпечення Mathcad. Навести результат перевірки: вектор нев'язки $\boldsymbol{r} = \boldsymbol{b} - \boldsymbol{Ax_m}$, де $\boldsymbol{x_m}$ - отриманий у Mathcad розв'язок.

Порівняти корені рівнянь, отримані у Mathcad, із власними результатами за допомогою методу середньоквадратичної похибки.

2 Розв’язок

Нижче наведені результати виконання програми.

Розширена матриця А:

```
Extended Matrix =
8.3 3.42 4.1 1.9 -9.85
0.0 -14.471582 -12.796429 -3.308673 -35.702806
0.0 0.0 -8.331822 45.075576 -78.43977
0.0 0.0 0.0 44.022875 -21.6207
```

Розширена матриця А з одиницями на головній діагоналі:

```
Extended Matrix In Row Echelon form =
1.0 0.412048 0.493976 0.228916 -1.186747
0.0 1.0 0.884245 0.228632 2.467098
0.0 0.0 1.0 -5.41005 9.41448
0.0 0.0 0.0 1.0 -0.491124
```

Вектор коренів рівнянь ***x***:

```
X =
-3.013084 -3.39588 6.757474 -0.491124
```

Вектор нев’язки ***r*** = ***b*** – ***Ax***:

```
R =
[-0.000001 -0.000002 -0.000003 -0.000001]
```

Увесь вивід:

```
Start Matrix =
8.3 3.42 4.1 1.9
3.92 8.45 7.98 2.46
3.77 8.01 8.04 2.28
2.21 2.85 1.69 6.99

 Right Part =
-9.85 12.21 14.65 -8.35

 Rows = 4
 Columns = 4
 Exteneded Rows = 4
 Exteneded Columns = 5
 n = 4

 Extended Matrix =
8.3 3.42 4.1 1.9 -9.85
3.92 8.45 7.98 2.46 12.21
3.77 8.01 8.04 2.28 14.65
2.21 2.85 1.69 6.99 -8.35
```

```
Extended Matrix  =
8.3 3.42 4.1 1.9 -9.85
0.0 -14.471582 -12.796429 -3.308673 -35.702806
0.0 -14.214748 -13.600796 -3.119629 -42.103316
0.0 -7.28362 -2.247059 -24.352036 21.509729

 Extended Matrix  =
8.3 3.42 4.1 1.9 -9.85
0.0 -14.471582 -12.796429 -3.308673 -35.702806
0.0 0.0 1.050108 -0.132678 7.161238
0.0 0.0 -8.331822 45.075576 -78.43977

 Extended Matrix  =
8.3 3.42 4.1 1.9 -9.85
0.0 -14.471582 -12.796429 -3.308673 -35.702806
0.0 0.0 -8.331822 45.075576 -78.43977
0.0 0.0 0.0 44.022875 -21.6207

 Extended Matrix  =
8.3 3.42 4.1 1.9 -9.85
0.0 -14.471582 -12.796429 -3.308673 -35.702806
0.0 0.0 -8.331822 45.075576 -78.43977
0.0 0.0 0.0 44.022875 -21.6207
```

```
Extended Matrix  =
8.3 3.42 4.1 1.9 -9.85
0.0 -14.471582 -12.796429 -3.308673 -35.702806
0.0 0.0 -8.331822 45.075576 -78.43977
0.0 0.0 0.0 44.022875 -21.6207

 X =
-3.013084 -3.39588 6.757474 -0.491124

 Extended Matrix =
8.3 3.42 4.1 1.9 -9.85
0.0 -14.471582 -12.796429 -3.308673 -35.702806
0.0 0.0 -8.331822 45.075576 -78.43977
0.0 0.0 0.0 44.022875 -21.6207
Matrix multipled by X =
 [-9.849999 12.210002 14.650003 -8.349999]
R =
 [-0.000001 -0.000002 -0.000003 -0.000001]
```

З вигляду вектору нев’язки випливає, що метод Гауса є доволі точним для розв’язання систем з несиметричною матрицею А, на відміну від методу квадратного кореня, який є абсолютно точним для симетричних матриць.

3 Розв’язок у Mathcad

Нижче наведено розв’язок системи у Mathcad

$k := 9 - 5 \quad \alpha := 0.2\ k \quad \beta := \alpha \quad \mathbf{ORIGIN} := 1 \quad N := 4$

$$X := \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$matrix := \begin{bmatrix} 8.30 & 2.62+\alpha & 4.10 & 1.90 \\ 3.92 & 8.45 & 8.78-\alpha & 2.46 \\ 3.77 & 7.21+\alpha & 8.04 & 2.28 \\ 2.21 & 3.65-\alpha & 1.69 & 6.99 \end{bmatrix} = \begin{bmatrix} 8.3 & 3.42 & 4.1 & 1.9 \\ 3.92 & 8.45 & 7.98 & 2.46 \\ 3.77 & 8.01 & 8.04 & 2.28 \\ 2.21 & 2.85 & 1.69 & 6.99 \end{bmatrix}$$

$$rightPart := \begin{bmatrix} -10.65+\beta \\ 12.21 \\ 15.45-\beta \\ -8.35 \end{bmatrix} = \begin{bmatrix} -9.85 \\ 12.21 \\ 14.65 \\ -8.35 \end{bmatrix} \qquad extenededMatrix := \mathrm{augment}(matrix, rightPart)$$

$$extenededMatrix = \begin{bmatrix} 8.3 & 3.42 & 4.1 & 1.9 & -9.85 \\ 3.92 & 8.45 & 7.98 & 2.46 & 12.21 \\ 3.77 & 8.01 & 8.04 & 2.28 & 14.65 \\ 2.21 & 2.85 & 1.69 & 6.99 & -8.35 \end{bmatrix}$$

Guess Values

$$X$$

Constraints

$$matrix \cdot X = rightPart$$

Solver

$$\mathbf{X} := \mathbf{find}(\mathbf{X}) = \begin{bmatrix} -3.013084 \\ -3.395879 \\ 6.757473 \\ -0.491124 \end{bmatrix}$$

$$r := rightPart - (matrix \cdot \mathbf{X})$$

$$r = \begin{bmatrix} -0.000000000000004 \\ -0.000000000000004 \\ -0.000000000000011 \\ 0 \end{bmatrix}$$

$$x := \begin{bmatrix} -3.013084 \\ -3.39588 \\ 6.757474 \\ -0.491124 \end{bmatrix}$$

$$Z := 10^{-4} - 10^{-6}$$

$$Z = 0.000099$$

$$xm := \mathbf{X}$$

$$\delta := \sqrt{\frac{1}{N} \sum_{k=1}^{N} \left(x_k - xm_k\right)^2} \qquad \delta = 0.000001$$

Вектор нев’язки з округлення:

$$r = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Вектор нев’язки без округлення:

$$r = \begin{bmatrix} -0.000000000000004 \\ -0.000000000000004 \\ -0.000000000000011 \\ 0 \end{bmatrix}$$

Порівняння отриманого результату (п. 2) із результатом з Mathcad за допомогою методу середньоквадратичної похибки:

$$\delta := \sqrt{\frac{1}{N} \sum_{k=1}^{N} \left(x_k - xm_k\right)^2} \qquad \delta = 0.000001$$

У технічних розрахунках точність вимірювань характеризують відносною похибкою. Результат вважають гарним, якщо відносна похибка не перевищує 0,1 %. Отже Наш результат є гарним

4 Лістинг програми

Lab2.py

```python
# need for multiplicating matrices in the end
import numpy as np

# region Starting Values

matrix = [[8.30, 3.42, 4.10, 1.90],
        [3.92, 8.45, 7.98, 2.46],
        [3.77, 8.01, 8.04, 2.28],
        [2.21, 2.85, 1.69, 6.99]]

rightPart =[-9.85, 12.21, 14.65, -8.35]

onesDiagonal = False

n = len(matrix)
X = [0] * n
rounding = 6

# endregion Starting Values

# copy matrix to create extended matrix
extendedMatrix = list(map(list, matrix))

# region Prints

# Print vector
def PrintVector(vectorName,vector):
  print("\n",vectorName,"=")
  for i in vector:
    print(i, end = ' ')
  print()

# print matrix
def PrintMatrix(matrixName,matrix):
  print("\n",matrixName,"=")
  for i in matrix:
    for j in i:
      print(j, end=" ")
    print()

# print additional parametrs
def PrintParametrs():
  print("\n Rows =", rows)
  print(" Columns =", columns)
  print(" Exteneded Rows =", extendedRows)
  print(" Exteneded Columns =", extendedColumns)
  print(" n =", n)

# just printing
def PrintAll():
  PrintMatrix("Start Matrix",matrix)

  PrintVector("Right Part", rightPart)
```

```
  PrintParametrs()

  PrintMatrix("Extended Matrix",extendedMatrix)

# endregion Prints

# add right part to main matrix
RPCounter = 0
while RPCounter < len(rightPart):
  extendedMatrix[RPCounter].append(rightPart[RPCounter])
  RPCounter += 1

# region Getting rows and columns

rows = len(matrix)
columns = len(matrix)

extendedRows = len(extendedMatrix)
extendedColumns = len(extendedMatrix)+1

# endregion Getting rows and columns

PrintAll()
tempMatrix = list(map(list, extendedMatrix))

# Iterating through matrix
# Forward Elimination without Row Echelon
for i in range(rows):
  for j in range(i+1,columns):
    # Getting max number
    maxNum = [max(k, key=abs) for k in zip(*tempMatrix)][0]

    columnArray = [sub[i] for sub in extendedMatrix]

    maxNumIndex = columnArray.index(maxNum)

    # Swapping rows
    firstTempRow = extendedMatrix[:][i]

    rowToChange = extendedMatrix[:][maxNumIndex]

    extendedMatrix[i] = rowToChange
    extendedMatrix[maxNumIndex] = firstTempRow

    # Doing triangular matrix
    if matrix[j][i] == 0:continue
    topElement = extendedMatrix[i][i]
    bottomElement = extendedMatrix[j][i]

    multiplier = topElement / bottomElement

    for k in range(i,extendedColumns):
      extendedMatrix[j][k] = round(extendedMatrix[i][k] - (extendedMatrix[j][k] * multiplier),
rounding)

  PrintMatrix("Extended Matrix ",extendedMatrix)
```

```python
    # Creating temp matrix for correct numbers and indexes
    tempMatrix = list(map(list, extendedMatrix))

    for z in range(i+1):
        for m in tempMatrix:
            del m[0]
    for z in range(i+1):
        del tempMatrix[0]

PrintMatrix("Extended Matrix ",extendedMatrix)

# Make Row Echelon Form
if onesDiagonal == True:
    for i in range(n):
        for j in range(n):
            if i == j:
                if matrix[j][i] == 0:continue
                topElement = 1
                bottomElement = extendedMatrix[j][i]
                multiplier = topElement / bottomElement
                for k in range(i,extendedColumns):
                    extendedMatrix[j][k] = round(extendedMatrix[j][k] * multiplier, rounding)
    PrintMatrix("Extended Matrix In Row Echelon form",extendedMatrix)

# Search for X
# Back-Substitution
for i in range(rows-1, -1, -1):
    element = extendedMatrix[i][extendedColumns-1]
    for j in range(i+1, extendedColumns-1):
        element -= extendedMatrix[i][j] * X[j]
    finalElement = element / extendedMatrix[i][i]
    X[i] = round(finalElement, rounding)

PrintVector("X",X)
PrintMatrix("Extended Matrix",extendedMatrix)

# region Check the results

multiplied = np.round(np.dot(matrix,X),rounding)

print("Matrix multipled by X =\n",multiplied)

R = np.round(np.subtract(rightPart,multiplied),rounding)

np.set_printoptions(suppress=True)

print("R =\n",R)

# endregion Check the results
```

Висновок:

Я навчився розв’язувати СЛАР прямими методами (метод Гауса), отримав більше знань для роботи з MathCad та на практиці з’ясував, як порівнювати результати методом середньоквадратичної похибки.