Міністерство науки і освіти, молоді та спорту України

Національний технічний університет України

“ Київський політехнічний інститут ”

Кафедра ОТ

ЗВІТ

про виконання лабораторної роботи №1

з дисципліни

«Чисельні методи»

Тема: Розв’язання систем лінійних алгебраїчних рівнянь (СЛАР) прямими методами. Метод Гауса та метод квадратних коренів

Прийняв: Виконав

доц. Фіногенов О.Д. студент 3-го курсу

групи ІП-22

Сочка Олександр

Київ - 2015

**Постановка задачі**

Розв’язати систему рівнянь з кількістю значущих цифр *m* = 6. Якщо матриця системи симетрична, то розв’язання проводити за методом квадратних коренів, якщо матриця системи несиметрична, то використати метод Гауса. Вивести всі проміжні результати (матриці А, що отримані в ході прямого ходу методу Гауса, матрицю зворотного ходу методу Гауса, або матрицю Т та вектор y для методу квадратних коренів), та розв’язок системи. Навести результат перевірки: вектор нев’язки ***r* = *b* – *Ax*,** де ***x*** - отриманий розв’язок.

Розв’язати задану систему рівнянь за допомогою програмного забезпечення Mathcad. Навести результат перевірки: вектор нев’язки ***r* = *b* – *Axm*,** де ***xm*** - отриманий у Mathcad розв’язок.

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

,

де ***x*** - отриманий у програмі розв’язок, ***xm*** - отриманий у Mathcad розв’язок.

**Вхідна система рінянь**

6.92\*x1 + 1.28\*x2 + 0.79\*x3 + 1.15\*x4 -0.66\*x5 = 2.1

0.92\*x1 + 3.5\*x2 + 1.3\*x3 -1.62\*x4 + 1.02\*x5 = 0.72

1.15\*x1 -2.46\*x2 + 6.1\*x3 + 2.1\*x4 + 1.483\*x5 = 3.87

1.33\*x1 + 0.16\*x2 + 2.1\*x3 + 5.44\*x4 -18\*x5 = 13.8

1.14\*x1 -1.68\*x2 -1.217\*x3 + 9\*x4 -3\*x5 = -1.08

**Проміжні результати та кінцевий результат**

Straight move #1

Swapped rows 1 and 1

6.920000 1.280000 0.790000 1.150000 -0.660000 2.100000

-0.000000 3.329827 1.194971 -1.772890 1.107746 0.440809

0.000000 -2.672717 5.968714 1.908887 1.592682 3.521012

0.000000 -0.086012 1.948165 5.218974 -17.873150 13.396387

0.000000 -1.890867 -1.347145 8.810549 -2.891272 -1.425954

Straight move #2

Swapped rows 2 and 2

6.920000 1.280000 0.790000 1.150000 -0.660000 2.100000

-0.000000 3.329827 1.194971 -1.772890 1.107746 0.440809

-0.000000 0.000000 6.927869 0.485860 2.481825 3.874831

-0.000000 0.000000 1.979032 5.173179 -17.844537 13.407774

-0.000000 0.000000 -0.668571 7.803800 -2.262230 -1.175637

Straight move #3

Swapped rows 3 and 3

6.920000 1.280000 0.790000 1.150000 -0.660000 2.100000

-0.000000 3.329827 1.194971 -1.772890 1.107746 0.440809

-0.000000 0.000000 6.927869 0.485860 2.481825 3.874831

0.000000 0.000000 0.000000 5.034387 -18.553500 12.300880

-0.000000 0.000000 0.000000 7.850688 -2.022722 -0.801698

Straight move #4

Swapped rows 4 and 4

6.920000 1.280000 0.790000 1.150000 -0.660000 2.100000

-0.000000 3.329827 1.194971 -1.772890 1.107746 0.440809

-0.000000 0.000000 6.927869 0.485860 2.481825 3.874831

0.000000 0.000000 0.000000 5.034387 -18.553500 12.300880

-0.000000 0.000000 0.000000 0.000000 26.909844 -19.983848

Backward move #1:

6.920000 1.280000 0.790000 1.150000 0.000000 1.609869

-0.000000 3.329827 1.194971 -1.772890 0.000000 1.263446

-0.000000 0.000000 6.927869 0.485860 0.000000 5.717889

0.000000 0.000000 0.000000 5.034387 0.000000 -1.477362

-0.000000 0.000000 0.000000 0.000000 26.909844 -19.983848

Backward move #2:

6.920000 1.280000 0.790000 0.000000 0.000000 1.947342

-0.000000 3.329827 1.194971 0.000000 0.000000 0.743184

-0.000000 0.000000 6.927869 0.000000 0.000000 5.860467

0.000000 0.000000 0.000000 5.034387 0.000000 -1.477362

-0.000000 0.000000 0.000000 0.000000 26.909844 -19.983848

Backward move #3:

6.920000 1.280000 0.000000 0.000000 0.000000 1.279060

-0.000000 3.329827 0.000000 0.000000 0.000000 -0.267674

-0.000000 0.000000 6.927869 0.000000 0.000000 5.860467

0.000000 0.000000 0.000000 5.034387 0.000000 -1.477362

-0.000000 0.000000 0.000000 0.000000 26.909844 -19.983848

Backward move #4:

6.920000 0.000000 0.000000 0.000000 0.000000 1.381955

-0.000000 3.329827 0.000000 0.000000 0.000000 -0.267674

-0.000000 0.000000 6.927869 0.000000 0.000000 5.860467

0.000000 0.000000 0.000000 5.034387 0.000000 -1.477362

-0.000000 0.000000 0.000000 0.000000 26.909844 -19.983848

Resulting matrix:

6.920000 0.000000 0.000000 0.000000 0.000000 1.381955

-0.000000 3.329827 0.000000 0.000000 0.000000 -0.267674

-0.000000 0.000000 6.927869 0.000000 0.000000 5.860467

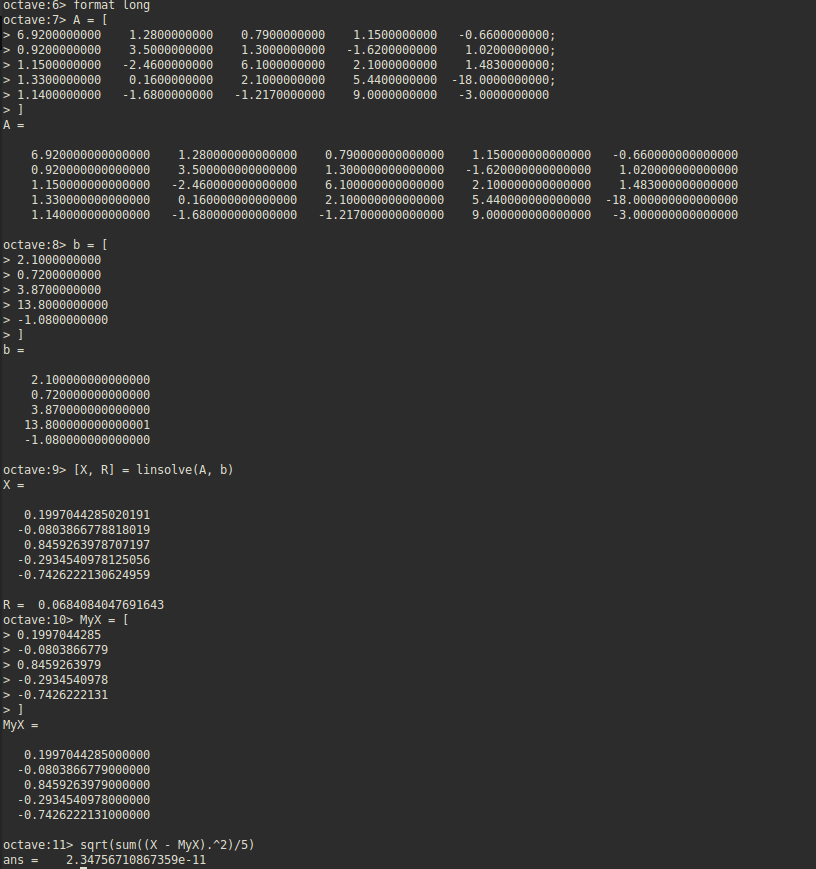
0.000000 0.000000 0.000000 5.034387 0.000000 -1.477362

-0.000000 0.000000 0.000000 0.000000 26.909844 -19.983848

Результат: X = [0.199704, -0.080387, 0.845926, -0.293454, -0.742622]

Вектор нев’язки: [2.5e-11, -8.6e-11, 1.917e-10, 7.99e-10, 2.177e-10]

**Перевірка результатів**



Результати обробки у MatLab

X = [ 0.1997044285020191, -0.0803866778818019, 0.8459263978707197, -0.2934540978125056, -0.7426222130624959]

Вектор нев’язки: [0, 1.11022302462516e-16, -1.33226762955019e-15, -1.77635683940025e-15, 4.44089209850063e-16]

Абсолютна різниця між результатами MatLab і написаної програми: [-2.01910710373454e-12, -1.81980680524774e-11, 2.92803559176491e-11, 1.25056631716802e-11, -3.75041109279550e-11]

Середньоквадратична похибка: sqrt(sum((X - MyX).^2)/5) = 2.34e-11

**Лістинг програми**

import scala.io.Source

object Lab1 extends App {

def printAb(msg: String = "") = {

if (msg.nonEmpty) println(s"$msg: ")

for (line <- Ab) {

for (elem <- line) print(f"$elem%16.10f,")

println()

}

println()

}

val N = 5

val Ab =

for (line <- Source.fromFile("input/lab1.txt").getLines().toArray) yield

for (word <- line split ' ') yield word.toDouble

printAb("Matrix read")

for (majorRowIndex <- 0 until N - 1) {

println(s"Straight move #${majorRowIndex + 1}")

val bestIndex = 0.until(N)

.map(j => (Ab(j)(majorRowIndex), majorRowIndex))

.sortBy(x => math.abs(x.\_1))

.head.\_2

println(s"Swapped rows ${majorRowIndex + 1} and ${bestIndex + 1}")

val row = Ab(majorRowIndex)

Ab(majorRowIndex) = Ab(bestIndex)

Ab(bestIndex) = row

for (row <- Ab.drop(majorRowIndex + 1)) {

val k = row(majorRowIndex) / Ab(majorRowIndex)(majorRowIndex)

for ((el, colIndex) <- row.zipWithIndex)

row(colIndex) -= Ab(majorRowIndex)(colIndex) \* k

}

printAb()

}

for (majorIndex <- N - 1 to 1 by -1) {

for (rowIndex <- 0 until majorIndex) {

Ab(rowIndex)(N) -= Ab(majorIndex)(N) / Ab(majorIndex)(majorIndex) \* Ab(rowIndex)(majorIndex)

Ab(rowIndex)(majorIndex) = 0

}

printAb(s"Backward move #${N - majorIndex}")

}

printAb("Resulting matrix")

println("X = ")

for ((row, index) <- Ab.zipWithIndex)

println(f"${row.last / row(index)}%.10f")

}