Лабораторная работа №4
LU-разложение

> *Выполнил студент Гринёв Максим Б9119-01.03. 02миопд*. 16/03/2022

Постановка задачи

Пусть дана система n линейных алгебраических уравнений с n неизвестными, записанная в матричной форме

$$A\overline{x} = \overline{f}, \tag{1}$$

где A - матрица коэффициентов при неизвестных системы (1), $\overline{b}$ - вектор-столбец ва ее свободных членов, $\overline{x}$ - столбец неизвестных (искомый вектор):

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix}, \ \overline{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}, \ \overline{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix},$$

Система (1). в развернутом виде может быть выписана так:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n = b_2 \\ \dots \\ a_{n1}x_1 + a_{n2}x_2 + \dots + a_{nn}x_n = b_n \end{cases}$$

Требуется найти решение этой системы, т.е совокупность чисел $x_1, x_2, \dots, x_n$, обращающих систему (1) в систему тождеств. В силу того, что $detA \neq 0$, такое решение существует и единственно.

Методы решения систем линейных алгебраических уравнений делятся на две большие группы: так называемые **точные методы** и **методы последовательных приближений**. Точные методы характеризуются тем, что с их помощью принципиально возможно, про- делав конечное число операций, получить точные значения неизвестных. При этом, конечно, предполагается, что коэффициенты и правые части системы известны точно, а все вычисле- ния производятся без округлений. Чаще всего они осуществляются в два этапа. На первом этапе преобразуют систему к тому или иному простому виду. На втором этапе решают упро- щенную систему и получают значения неизвестных.

В этой лабораторной работе мы будем работать с методом LU-разложения

LU-разложение — это представление матрицы A в виде $A = L \cdot U$, где L — нижнетреугольная матрица с еденичной диагональю, а U — верхнетреугольная матрица. LU-разложение является модификациeё метода Гаусса. Основные применения данного алгоритма — решение систем алгебраических уравнений, вычисление определителя, вычисление обратной матрицы и др.

Рассмотрим алгоритм на примере матрицы A:

$$A = \begin{pmatrix} 10 & -7 & 0 \\ -3 & 6 & 2 \\ 5 & -1 & 5 \end{pmatrix}$$

Алгоритм

1. Создаем матрицы

$$L = \begin{pmatrix} l_{1,1} & 0 & 0 \\ l_{2,1} & l_{2,2} & 0 \\ l_{3,1} & l_{3,2} & l_{3,3} \end{pmatrix}$$

и

$$U = A = \begin{pmatrix} l_{1,1} & 0 & 0 \\ l_{2,1} & l_{2,2} & 0 \\ l_{3,1} & l_{3,2} & l_{3,3} \end{pmatrix}$$

2. Для каждого столбца $j = 1..3$ матрицы L будем вычислять $l_{i,j}$ как $l_{i,j} = \frac{u_{j,i}}{u_{i,j}}$
   Для каждой строки c_i вычислим $c_i = c_i - l_{i,j} \cdot c_j$

3. Выполняем шаг 2 пока $j \leq 3$

4. Получим:

$$L = \begin{pmatrix} 1 & 0 & 0 \\ l_{2,1} & 1 & 0 \\ l_{3,1} & l_{3,2} & 1 \end{pmatrix}$$

   и

$$U = \begin{pmatrix} u_{1,1} & u_{1,2} & u_{1,3} \\ 0 & u_{2,2} & u_{2,3} \\ 0 & 0 & u_{3,3} \end{pmatrix}$$

   такие, что $A = L \cdot U$

Результаты после каждого шага:

$$1. \quad L = \begin{pmatrix} 1 & 0 & 0 \\ -0.3 & l_{2,2} & 0 \\ 0.5 & l_{3,2} & l_{3,3} \end{pmatrix}$$

и

$$U = \begin{pmatrix} 10 & -7 & 0 \\ 0 & -0.1 & 6 \\ 0 & 2.5 & 5 \end{pmatrix}$$

2. $$L = \begin{pmatrix} 1 & 0 & 0 \\ -0.3 & 1 & 0 \\ 0.5 & -25 & l_{3,3} \end{pmatrix}$$

и

$$U = \begin{pmatrix} 10 & -7 & 0 \\ 0 & -0.1 & 6 \\ 0 & 0 & 155 \end{pmatrix}$$

3. $$L = \begin{pmatrix} 1 & 0 & 0 \\ -0.3 & 1 & 0 \\ 0.5 & -25 & 1 \end{pmatrix}$$

и

$$U = \begin{pmatrix} 10 & -7 & 0 \\ 0 & -0.1 & 6 \\ 0 & 0 & 155 \end{pmatrix}$$

Алгоритм

Алгоритм реализован на языке **Python**:

```
import math
import cmath
from copy import copy, deepcopy
import numpy as np


def solveMatrixLU(A,B):

        def LUSum(LU, iIndex, jIndex):
            total = 0
            for kIndex in range(iIndex):
                total += LU[iIndex][kIndex] * LU[kIndex][jIndex]
            return total

        def ULSum(LU, iIndex, jIndex):
            total = 0
            for kIndex in range(iIndex):
                total += LU[jIndex][kIndex] * LU[kIndex][iIndex]

            return total

        def ySum(iIndex, LU, yAr):
            total = 0
            for pIndex in range(iIndex):
                total += LU[iIndex][pIndex] * yAr[pIndex]
            return total

        def xSum(iIndex, LU, xAr, N):
            total = 0
            for pIndex in range(1, iIndex):
                total += LU[N - iIndex][N - pIndex] * xAr[N - pIndex]
            return total

        initialMatrix = deepcopy(A)
        lu = deepcopy(A)
        initialSol = list(copy(B))

        n = len(lu)

        for i in range(1, n):
            lu[i][0] = initialMatrix[i][0] / lu[0][0]

        for i in range(1, n):
            for j in range(i, n):
                lu[i][j] = initialMatrix[i][j] - LUSum(lu, i, j)

            for j in range(i + 1, n):
                lu[j][i] = 1 / lu[i][i] * (initialMatrix[j][i] -
```

```
ULSum(lu, i, j))

        y = [0 for i in range(n)]

        for i in range(n):
            y[i] = initialSol[i] - ySum(i, lu, y)

        x = [0 for i in range(n)]

        for i in range(1, n + 1):
            x[n - i] = 1 / lu[n - i][n - i] * (y[n - i] - xSum(i, lu,
x, n))

        return x

def numpySolution(givenMatrix, givenSolution):
    return np.linalg.solve(givenMatrix, givenSolution)


def returnSolution(finalMatrix, finalSol):
    rootsList = [0] * len(finalMatrix[0])
    for equationI in reversed(range(len(finalMatrix))):
        tempRoot = finalSol[equationI]
        indexOfRoot = 0
        if equationI == len(finalMatrix) - 1:
            for rootJ in range(len(finalMatrix[equationI])):
                if finalMatrix[equationI][rootJ] != 0:
                    indexOfRoot = rootJ
                    rootsList[indexOfRoot] = tempRoot
        else:
            for rootJ in range(len(finalMatrix[equationI])):
                if finalMatrix[equationI][rootJ] != 0 and
rootsList[rootJ] == 0:
                    indexOfRoot = rootJ
            for rootJ in range(len(finalMatrix[equationI])):
                if finalMatrix[equationI][rootJ] != 0 and rootJ !=
indexOfRoot:
                    tempRoot -= rootsList[rootJ] *
finalMatrix[equationI][rootJ]
            rootsList[indexOfRoot] = tempRoot

    return rootsList


def printE(myX, pythonX):
    for i in range(len(myX)):
        print("E" + str(i) + " = ", end="")
        print(abs(pythonX[i] - myX[i]))
    print("\n")



def main():

    A = [
        [0.411, 0.421, -0.333, 0.313, -0.141, -0.381, 0.245],
```

```
        [0.241, 0.705, 0.139, -0.409, 0.321, 0.0625, 0.101],
        [0.123, -0.239, 0.502, 0.901, 0.243, 0.819, 0.321],
        [0.413, 0.309, 0.801, 0.865, 0.423, 0.118, 0.183],
        [0.241, -0.221, -0.243, 0.134, 1.274, 0.712, 0.423],
        [0.281, 0.525, 0.719, 0.118, -0.974, 0.808, 0.923],
        [0.246, -0.301, 0.231, 0.813, -0.702, 1.223, 1.105]]

    B = [0.096, 1.252, 1.024, 1.023, 1.155, 1.937, 1.673]


    my = solveMatrixLU(A, B)
    numpy = numpySolution(A, B)

    print(my)
    print(numpy)
    printE(my, numpy)

if __name__ == '__main__':
    main()
```

Тесты

Тестовая система:

$$A = \begin{bmatrix} 0.411 & 0.421 & -0.333 & 0.313 & -0.141 & -0.381 & 0.245 \\ 0.241 & 0.705 & 0.139 & -0.409 & 0.321 & 0.0625 & 0.101 \\ 0.123 & -0.239 & 0.502 & 0.901 & 0.243 & 0.819 & 0.321 \\ 0.413 & 0.309 & 0.801 & 0.865 & 0.423 & 0.118 & 0.183 \\ 0.241 & -0.221 & -0.243 & 0.134 & 1.274 & 0.712 & 0.423 \\ 0.281 & 0.525 & 0.719 & 0.118 & -0.974 & 0.808 & 0.923 \\ 0.246 & -0.301 & 0.231 & 0.813 & -0.702 & 1.223 & 1.10 \end{bmatrix}, \ \overline{b} = \begin{bmatrix} 0.096 \\ 1.252 \\ 1.024 \\ 1.023 \\ 1.155 \\ 1.937 \\ 1.673 \end{bmatrix},$$

```
Решение алгоритмом LU-разложения:
[11.091969628167396, -2.515736321595345, 0.720986479267067,
-2.5446744665696444, -1.6048265844707768, 3.6239736592517513,
-4.949589813830162]

Решение numPy:
[11.09196963 -2.51573632  0.72098648 -2.54467447 -1.60482658
3.62397366
 -4.94958981]

Сравнение погрешностей двух решений:
E0 = 3.552713678800501e-15
E1 = 3.552713678800501e-15
E2 = 1.7763568394002505e-15
```

```
E3 = 1.7763568394002505e-15
E4 = 4.440892098500626e-15
E5 = 6.661338147750939e-15
E6 = 9.769962616701378e-15
```

Заключение

В этой лабораторной работе мы разобрались с тем как используется метод LU-разложения. Однажды найдя LU-разложение для матрцицы мы можем очень быстро решать системы линейных алгебраических уравнений с различной правой частью.

Сложность алгоритма: $\frac{2 \cdot n^3}{3} + O(n^2)$

Мы создали алгоритм на языке **Python** и выяснили погрешность в сравнении с библиотекой вычислительной математики **numPy**.