Лабораторная работа №3
Метод оптимального исключения

> *Выполнил студент Гринёв Максим Б9119-01.03. 02миопд*. 14/03/2022

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 точным методом оптимального исключения

Пусть дана система уравнений $A\bar{x} = \bar{b}$. Обозначив b_i через a_{in+1}, преобразуем эту систему к эквивалентной системе более простого вида. Допустим, что $a_{11} \neq 0$. Разделим все коэффициенты первого уравнения системы на a_{11}, который назовем ведущим элементом первого шага, тогда

$$x_1 + a_{12}^{(1)} \cdot x_2 + ... + a_{1n}^{(1)} \cdot x_n = a_{1n+1}^{(1)}$$

Здесь $a_{1j}^{(1)} = a_{1j}/a_{11},\ j = 2, 3, \cdots, n+1$.

Предположим, что после преобразования первых k $(k \geq 1)$ уравнений система приведена к эквивалентной системе

$$\begin{cases} x_1 + \cdots\cdots\cdots\cdots + a_{1k+1}^{(k)} x_{k+1} + \cdots + a_{1n}^{(k)} x_n & = & a_{1n+1}^{(k)} \\ x_2 + \cdots\cdots\cdots\cdots + a_{2k+1}^{(k)} x_{k+1} + \cdots + a_{2n}^{(k)} x_n & = & a_{2n+1}^{(k)} \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots & & \cdots \\ x_k + a_{kk+1}^{(k)} x_{k+1} + \cdots + a_{kn}^{(k)} x_n & = & a_{kn+1}^{(k)} \\ a_{k+11} x_1 + a_{k+12} x_2 + \cdots + a_{k+1n+1} x_k + \cdots + a_{k+1n} x_n & = & a_{k+1n+1} \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots & & \cdots \\ a_{n1} x_1 + a_{n2} x_2 + \cdots + a_{nk+1} x_k + \cdots + a_{nn} x_n & = & a_{nn+1} \end{cases}$$

Исключим неизвестные $x_1, x_2, \cdots, x_k$ из $(k+1)$ уравнения посредством вычитания из него первых k уравнений, умноженных соответственно на числа $a_{k+11}, a_{k+12}, \cdots, a_{k+1k}$, и разделив вновь полученное уравнение на коэффициенты при x_{k+1}. Теперь $(k+1)$ уравнение примет вид:

$$x_{k+1} + a_{k+1k+2}^{(k+1)} \cdot x_{k+2} + \cdots + a_{k+1n}^{(k+1)} \cdot x_n = a_{k+1n+1}^{(k+1)}$$

Исключая с помощью этого уравнения неизвестное x_{k+1} из первых k уравнений (3), получаем опять систему вида (3), но с заменой индекса k на $k+1$, причем

$$a_{i1}^{(1)} = \frac{a_{1i}}{a_{11}},\ i = 2, 3, ..., n+1.$$

$$a_{k+1p}^{(k+1)} = \frac{a_{k+1p} - \sum\limits_{r=1}^{k} a_{rp}^{(k)} a_{k+1r}}{a_{k+1k+1} - \sum\limits_{r=1}^{k} a_{rk+1}^{(k)} a_{k+1r}};$$

$$a_{ip}^{(k+1)} = a_{ip}^{(k)} - a_{k+1p}^{(k+1)} a_{ik+1}^{(k)},\ i = 1, 2, \cdots, k;$$

$$p = k+2, k+3, \cdots, n+1; \quad k = 0, 1, \cdots, n-1.$$

После преобразования всех уравнений находим решение исходной системы $x_i = a_{in+1}^{(n)}, i = 1, 2, \cdots, n$.

Указанная схема оптимального исключения работает в случае неравенства нулю ведущих элементов. Наилучшим вариантом методом оптимального исключения является вариант с выбором максимального по модулю элемента в строке. В этом случае структура исключения сохраняется, меняется лишь порядок исключения неизвестных. Теперь в качестве ведущего элемента будем брать максимальный по модулю элемент того уравнения, который получается из $(k+1)$-го уравнения исходной системы после исключения первых k шагов. Ведущим элементом первого шага будет максимальный по модулю элемент первого уравнения системы (3). При проведении расчетов удобно пользоваться следующей вычислительной схемой:

Матрица k-го шага

$1\ 0 \ldots 0$	$a_{1k+1}^{(k)}$	$a_{1k+2}^{(k)} \cdots\cdots a_{1n+1}^{(k)}$
$0\ 1 \cdots 0$	$a_{2k+1}^{(k)}$	$a_{2k+2}^{(k)} \cdots\cdots a_{2n+1}^{(k)}$
$\cdots\ \cdots\ \cdots$	$\cdots$	$\cdots\ \cdots\ \cdots$
$0\ 0 \ldots 1$	$a_{kk+1}^{(k)}$	$a_{kk+2}^{(k)} \cdots\cdots a_{kn+1}^{(k)}$
$a_{k+11} a_{k+12} \ldots a_{k+1k}$	a_{k+1k+1}	$a_{k+1k+2} \cdots\cdots a_{k+1n+1}$
$a_{k+21} a_{k+22} \cdots a_{k+2k}$	a_{k+2k+1}	$a_{k+2k+2} \cdots\cdots a_{k+2n+1}$

$a_{k+21} a_{k+22} \quad a_{k+2k}$	a_{k+2k+1}	$a_{k+2k+2} \ldots\ldots a_{k+2n+1}$
$\ldots\ \ldots\ \ldots$	$\ldots$	$\ldots\ \ldots\ \ldots$
$a_{n1} a_{n2} \ldots a_{nk}$	a_{nk+1}	$a_{nk+2} \ldots\ldots a_{nn+1}$

Матрица $(k+1)$-го шага после преобразования:

$1\ 0 \ldots 0$ $0\ 1 \ldots 0$ $\ldots\ \ldots\ \ldots$ $0\ 0 \ldots 1$	0 0 $\ldots$ 0	$a_{ip}^{(k+1)} = a_{ip}^{(k)} + a_{k+1p}^{(k+1)} a_{ik+1}$ $i = 1, 2, \cdots, k$ $\ldots\ \ldots\ \ldots$ $p = k+2, \ldots, n+1$
$0\ 0 \ldots 0$	1	$a_{k+1p}^{(k+1)} = \frac{a_{k+1p} - \sum\limits_{r=1}^{k} a_{rp}^{(k)} a_{k+1r}}{a_{k+1k+1} - \sum\limits_{r=1}^{k} a_{rk+1}^{(k)} a_{k+1r}}$
$a_{k+21}\ a_{k+22}\ \ldots\ a_{k+2k}$ $\ldots\ \ldots\ \ldots$ $a_{n1}\ a_{n2}\ \ldots\ a_{nk}$	a_{k+2k+1} $\ldots$ a_{nk+1}	$a_{k+2k+2} \ldots\ldots a_{k+2n+1}$ $\ldots\ \ldots\ \ldots$ $a_{nk+2} \ldots\ldots a_{nn+1}$

Пример. Решить методом оптимального исключения систему

$$\begin{cases} 5x_1 + 2x_2 + 3x_3 = 3 \\ x1 + 6x_2 + x_3 = 5 \\ 3x_1 - 4x_2 - 2x_3 = 8 \end{cases}$$

Вычислительная схема

k	$a_{i1}^{(k)}$	$a_{i2}^{(k)}$	$a_{i3}^{(k)}$	$a_{i4}^{(k)}$	k	$a_{i1}^{(k)}$	$a_{i2}^{(k)}$	$a_{i3}^{(k)}$	$a_{i4}^{(k)}$
	5	2	3	3		1	$2/5$	$3/5$	$3/5$
0	1	6	1	5	1	1	6	1	5
	3	-4	-2	8		3	-4	-2	8
k	$a_{i1}^{(k)}$	$a_{i2}^{(k)}$	$a_{i3}^{(k)}$	$a_{i4}^{(k)}$	k	$a_{i1}^{(k)}$	$a_{i2}^{(k)}$	$a_{i3}^{(k)}$	$a_{i4}^{(k)}$
	1	0	$4/7$	$2/7$		1	0	0	2
2	0	1	$1/14$	$11/14$	3	0	1	0	1
	3	-4	-2	8		0	0	1	-3

Ответ: $x_1 = 2,\ x_2 = 1,\ x_3 = -3.$

Алгоритм

Алгоритм реализован на языке **Python**:

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


def solveMatrixOptimalException(A,B):

        def upSum(nArray, oArray, kIndex, jIndex):
            total = 0
            for index in range(kIndex):
                total += nArray[index][jIndex] * oArray[kIndex]
[index]
            return total

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

        n = len(initialMatrix)

        finalMatrix = []
        for i in range(n):
            finalMatrix.append([*initialMatrix[i], initialSol[i]])

        A_B = deepcopy(finalMatrix)

        major = finalMatrix[0][0]
        for i in range(n + 1):
            finalMatrix[0][i] /= major

        for k in range(1, n):

            for j in range(k + 1, n + 1):
                s1 = upSum(finalMatrix, A_B, k, j)
                s2 = upSum(finalMatrix, A_B, k, k)
                finalMatrix[k][j] = (A_B[k][j] - s1) / (A_B[k][k] -
s2)

            for i in range(k - 1, -1, -1):
                for j in range(k + 1, n + 1):
                    finalMatrix[i][j] = finalMatrix[i][j] -
finalMatrix[k][j] * finalMatrix[i][k]

            for i in range(k):
                finalMatrix[i][k] = 0
                finalMatrix[k][i] = 0

            finalMatrix[k][k] = 1
```

```python
        matr = []
        sol = []
        for i in range(len(finalMatrix)):
            matr.append(finalMatrix[i][:len(finalMatrix)])
            sol.append((finalMatrix[i][len(finalMatrix)]))

        return returnSolution(matr, sol)

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 = solveMatrixOptimalException(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 оптимального исключения:
[11.091969628167696, -2.515736321595433, 0.7209864792670765,
-2.544674466569711, -1.6048265844708345, 3.623973659251825,
-4.949589813830322]

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

Сравнение погрешностей двух решений:
E0 = 2.9665159217984183e-13
E1 = 9.14823772291129e-14
E2 = 1.1213252548714081e-14
E3 = 6.838973831690964e-14
E4 = 5.3290705182007514e-14
E5 = 6.705747068735946e-14
E6 = 1.501021529293116e-13
```

Заключение

В этой лабораторной работе мы разобрались с тем как используется метод оптимального исключения. Существенное достоинство этого метода состоит в том, что он позволяет решать системы более высокого порядка при одних и тех же затратах оперативной памяти. Мы создали алгоритм на языке **Python** и выяснили погрешность в сравнении с библиотекой вычислительной математики **numPy**.