Лабораторная работа №2
Метод Гаусса с выбором главного элемента

Выполнил студент Гринёв Максим Б9119-01.03. 02миопд. 09/03/2022

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 Гаусса с выбором главного элемента

Метод Гаусса с выбором главного элемента

Процесс решения системы линейных алгебраических уравнений по методу Гаусса с выбором главного элемента сводится к построению системы с треугольной матрицей, эквивалентной исходной системе.

Рассмотрим расширенную прямоугольную матрицу, состоящую из коэффициентов при неизвестных и свободных членов системы (1).

$$M = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1j} & \dots & a_{1q} & \dots & a_{1n} & b_1 \\ a_{21} & a_{22} & \dots & a_{2j} & \dots & a_{2q} & \dots & a_{2n} & b_2 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ a_{p1} & a_{p2} & \dots & a_{pj} & \dots & a_{pq} & \dots & a_{pn} & b_p \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nj} & \dots & a_{nq} & \dots & a_{nn} & b_n \end{bmatrix}, \tag{2}$$

Выберем наибольший по модулю элемент a_{pq}, не принадлежащий столбцу свободных членов матрицы M. Этот элемент называется главным элементом. Строка матрицы M, содержащая главный элемент, называется главной строкой. Столбец матрицы M. содержащий главный элемент, называется главным столбцом. Далее, производя некоторые операции, построим матрицу $M^{(1)}$ с меньшим на единицу числом строк и столбцов. Матрица $M^{(1)}$ получатся преобразованием из M, при котором главная строка и главный столбец матрицы M исключается. Над матрицей $M^{(1)}$ повторяем те же операции, что и над матрицей $, после чего получаем матрицу $M^{(2)}$, и т.д. Таким образом, мы построим последовательность матриц

$$M, M^{(1)}, M^{(2)}, \dots, M^{(n-1)}, \tag{3}$$

последняя из которых представляет собой двухэлементную матрицу - строку; ее также считаем главной строкой.

Для получения системы с треугольной матрицей, эквивалентной системе (1), объединяем все главные строки матриц последовательности (3), начиная с последней $M^{(n-1)}$.

Рассмотрим подробнее эту схему для системы четырех уравнений с четырьмя неизвестными.

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + a_{14}x_n = b_1$$

$$a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + a_{24}x_n = b_2$$

$$a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + a_{34}x_n = b_3$$

$$a_{41}x_1 + a_{42}x_2 + a_{43}x_3 + a_{44}x_n = b_4$$

Вычисления удобно записать на расчетном бланке.

В таблице ниже показан процесс построения последовательности матриц (3). Главные элементы отмечены рамкой. В III столбце помещены значения m_i, равные отношению соответствующего элемента главного столбца к главному элементу с противоположным знаком. В строках, отмеченных звездочкой, выписываем элементы соответствующих главных строк,

I	II	III	IV	V	VI	VII	VIII
M	N	m_i	a_{i1}	a_{i2}	a_{i3}	a_{i4}	b_i
	1	$m_1 = -\frac{a_{13}}{a_{23}}$	a_{11}	a_{12}	a_{13}	a_{14}	b_1
	2		a_{21}	a_{22}	$\boxed{a_{23}}$	a_{24}	b_2
$M^{(0)}$	3	$m_3 = -\frac{a_{33}}{a_{23}}$	a_{31}	a_{32}	a_{33}	a_{34}	b_3
	4	$m_4 = -\frac{a_{43}}{a_{23}}$	a_{41}	a_{42}	a_{43}	a_{44}	b_4
	2*		$\alpha_{21} = \frac{a_{21}}{a_{23}}$	$\alpha_{22} = \frac{a_{22}}{a_{23}}$	$\alpha_{23} = 1$	$\alpha_{24} = \frac{a_{24}}{a_{23}}$	$\beta_2 = \frac{b_2}{a_{23}}$
	1	$m_1^{(1)} = -\frac{a_{12}^{(1)}}{a_{32}^{(1)}}$	$a_{11}^{(1)} = a_{11} + a_{21}m_1$	$a_{12}^{(1)} = a_{12} + a_{22}m_1$	0	$a_{14}^{(1)} = a_{14} + a_{24}m_1$	$b_1^{(1)} = b_1 + b_2m_1$
$M^{(1)}$	3		$a_{31}^{(1)} = a_{31} + a_{21}m_3$	$\boxed{a_{32}^{(1)} = a_{32} + a_{22}m_3}$	0	$a_{34}^{(1)} = a_{34} + a_{24}m_3$	$b_3^{(1)} = b_3 + b_2m_3$
	4	$m_4^{(1)} = -\frac{a_{42}^{(1)}}{a_{32}^{(1)}}$	$a_{41}^{(1)} = a_{41} + a_{21}m_4$	$a_{42}^{(1)} = a_{42} + a_{22}m_4$	0	$a_{44}^{(1)} = a_{44} + a_{24}m_4$	$b_4^{(1)} = b_4 + b_2m_4$
	3*		$\alpha_{31} = \frac{a_{31}^{(1)}}{a_{32}^{(1)}}$	$\alpha_{32} = 1$	0	$\alpha_{34} = \frac{a_{34}^{(1)}}{a_{32}^{(1)}}$	$\beta_3 = \frac{b_3^{(1)}}{a_{32}^{(1)}}$
	1	$m_1^{(2)} = -\frac{a_{11}^{(2)}}{a_{41}^{(2)}}$	$a_{11}^{(2)} = a_{11}^{(1)} + a_{31}^{(1)}m_1^{(1)}$	0	0	$a_{14}^{(2)} = a_{14}^{(1)} + a_{34}^{(1)}m_1^{(1)}$	$b_1^{(2)} = b_1^{(1)} + b_3^{(1)}m_1^{(1)}$
$M^{(2)}$	4		$\boxed{a_{41}^{(2)} = a_{41}^{(1)} + a_{31}^{(1)}m_4^{(1)}}$	0	0	$a_{44}^{(2)} = a_{44}^{(1)} + a_{34}^{(1)}m_4^{(1)}$	$b_4^{(2)} = b_4^{(1)} + b_3^{(1)}m_4^{(1)}$
	4*		$\alpha_{41} = 1$	0	0	$\alpha_{44} = \frac{a_{44}^{(2)}}{a_{41}^{(2)}}$	$\beta_4 = \frac{b_4^{(2)}}{a_{41}^{(2)}}$
$M^{(3)}$	1		0	0	0	$a_{14}^{(3)} = a_{14}^{(2)} + a_{44}^{(2)}m_1^{(2)}$	$b_4^{(3)} = b_1^{(2)} + b_4^{(2)}m_1^{(2)}$
	1*		0	0	0	1	$\beta_4 = \frac{b_4^{(3)}}{a_{14}^{(3)}} = x_4$
			1				$x_1 = \beta_4 - \alpha_{44}x_4$
				1			$x_2 = \beta_3 - \alpha_{34}x_4 - \alpha_{31}x_1$
					1		$x_3 = \beta_2 - \alpha_{24}x_4 - \alpha_{22}x_2 - \alpha_{21}x_1$

поделенных на их главные элементы. Объединив уравнения, отмеченные звездочкой, мы получим систему

$$\begin{cases} x_4 = \beta_4 & (1^*) \\ x_1 + \alpha_{44}x_4 = \beta_1 & (4^*) \\ \alpha_{31}x_1 + x_2 + \alpha_{34}x_4 = \beta_3 & (3^*) \\ \alpha_{21}x_1 + \alpha_{22}x_2 + x_3 + \alpha_{24}x_4 = \beta_2 & (2^*) \end{cases}$$

с треугольной матрицей, эквивалентную исходной системе. Из этой системы последовательно находим значения компонент искомого вектора x по формулам

$$x_4 = \beta_4$$

$$x_1 = \beta_1 - \alpha_{44}x_4$$

$$x_2 = \beta_3 - \alpha_{31}x_1 - \alpha_{34}x_4$$

$$x_3 = \beta_2 - \alpha_{21}x_1 - \alpha_{22}x_2 - \alpha_{24}x_4$$

Эту часть процесса отражают последние четыре строки таблицы 1.

Сам процесс исключения неизвестных называют прямым ходом, а решение системы с треугольной матрицей - обратным ходом.

Алгоритм

Алгоритм реализован на языке **Python**:

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

def solveMatrixMaxEl(matrix, solution):

        def deleteRowAndCol(rowIndex, colIndex):
            poppedRow = copy(initialMatrix[rowIndex])
            poppedSol = initialSol[rowIndex]
            deletedInitialRows.append(copy(poppedRow))
            deletedInitialSol.append(poppedSol)

            for colJ in range(len(poppedRow)):
                poppedRow[colJ] /= maxEl

            poppedSol /= maxEl

            for rowI in range(len(initialMatrix)):
                initialMatrix[rowI][colIndex] = 0

            initialMatrix.pop(rowIndex)
            initialSol.pop(rowIndex)

            finalSol.append(poppedSol)
            finalMatrix.append(copy(poppedRow))

        def changeElements():
            for rowI in range(len(initialMatrix)):
                initialSol[rowI] += deletedInitialSol[steps - 1] *
mList[steps - 1][rowI]
                for colJ in range(len(initialMatrix[rowI])):
                    initialMatrix[rowI][colJ] +=
(deletedInitialRows[steps - 1][colJ] * mList[steps - 1][rowI]) \
                        if abs(initialMatrix[rowI][colJ]) != 0 else 0

        def findMaxAbs():
            maxElement = initialMatrix[0][0]
            rowIndex = 0
            colIndex = 0
            for rowI in range(len(initialMatrix)):
                for colJ in range(len(initialMatrix[rowI])):
                    if abs(maxElement) < abs(initialMatrix[rowI]
[colJ]):
                        maxElement = initialMatrix[rowI][colJ]
                        rowIndex = rowI
                        colIndex = colJ
```

```
            return rowIndex, colIndex, maxElement

        initialMatrix = deepcopy(matrix)
        initialSol = list(copy(solution))
        finalMatrix = []
        finalSol = []
        deletedInitialRows = []
        deletedInitialSol = []
        mList = []

        for steps in range(len(matrix)):
            tempMList = []
            if steps != 0:
                changeElements()
            rowD, colD, maxEl = findMaxAbs()
            for i in range(len(initialMatrix)):
                if i != rowD:
                    mi = -(initialMatrix[i][colD] / maxEl)
                    tempMList.append(mi)
            deleteRowAndCol(rowD, colD)

            mList.append(copy(tempMList))
            tempMList.clear()

        return returnSolution(finalMatrix, finalSol)


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
```

```python
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():

    maxElM = [
        [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]]

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


    maxElRoots = solveMatrixMaxEl(maxElM, maxElSol)
    maxElPythonRoots = numpySolution(maxElM, maxElSol)

    print(maxElRoots)
    print(maxElPythonRoots)
    printE(maxElRoots, maxElPythonRoots)


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 Гаусса:
```

```
[11.091969628167407, -2.5157363215953468, 0.7209864792670684,
-2.544674466569646, -1.6048265844707805, 3.6239736592517557,
-4.949589813830171]

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

Сравнение погрешностей двух решений:
E0 = 7.105427357601002e-15
E1 = 5.329070518200751e-15
E2 = 3.1086244689504383e-15
E3 = 3.552713678800501e-15
E4 = 6.661338147750939e-16
E5 = 2.220446049250313e-15
E6 = 8.881784197001252e-16
```

Заключение

В этой лабораторной работе мы разобрались с тем как используется метод гаусса с выбором главного элемента. Особенностью этого метода является такая перестановка уравнений, чтобы на k-ом шаге ведущим элементом оказывался наибольший по модулю элемент k-го столбца. Мы создали алгоритм на языке **Python** и выяснили погрешность в сравнении с библиотекой вычислительной математики **numPy**.