МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

имени М.В.Ломоносова

Факультет вычислительной математики и кибернетики

Компьютерный практикум по учебному курсу

«ВВЕДЕНИЕ В ЧИСЛЕННЫЕ МЕТОДЫ»

ЗАДАНИЕ № 1

ОТЧЕТ

о выполненном задании

студента 201 учебной группы факультета ВМК МГУ

Бутылкина Андрея Сергеевича

гор. Москва

2022 г.

Содержание

Подвариант 1

Постановка задачи

Дана система уравнений Ax=f порядка n×n с невырожденной матрицей A. Написать программу, решающую систему линейных алгебраических уравнений заданного пользователем размера (n – параметр программы) методом Гаусса и методом Гаусса с выбором главного элемента. Предусмотреть возможность задания элементов матрицы системы и ее правой части как во входном файле данных, так и путем задания специальных формул. Также программа должна вычислять: определитель матрицы $det(A)$, обратную матрицу A^{-1}, число обусловленности $M_A = \|A\| \times \|A^{-1}\|$

Цели и задачи практической работы

- Изучить метод Гаусса и метод Гаусса с выбором главного элемента для решение системы линейных алгебраических уравнений с невырожденной матрицей коэффициентов;
- Решить заданную СЛАУ методом Гаусса и методом Гаусса с выбором главного элемента;
- Найти определитель матрицы, обратную матрицу, число обусловленности матрицы;
- Реализовать данные алгоритмы на любом языке программирования (в данном случае язык C);
- Провести тестирование программы;
- Исследовать вопрос вычислительной устойчивости метода Гаусса (при больших значениях параметра n);
- Правильность решения СЛАУ подтвердить системой тестов (используются ресурсы on-line системы http://www.wolframalpha.com).

Описание алгоритма решения

Пусть задана система линейных алгебраических уравнений:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1j}x_j + \ldots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2j}x_j + \ldots + a_{2n}x_n = b_2 \\ \ldots \\ a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{ij}x_j + \ldots + a_{in}x_n = b_i \\ \ldots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nj}x_j + \ldots + a_{nn}x_n = b_n \end{cases}$$

Записывая в матричном виде:

$$Ax = b$$

Где A - невырожденная матрица. Тогда метод Гаусса заключается в приведении матрицы коэффициентов к верхнему треугольному виду и выполнении обратного хода. Рассмотрим эти этапы подробнее.

Приведение матрицы к верхней треугольной форме:

На $i-$ом шаге разделим $i-$ую строку на элемент a_{ii}, после вычтем из $j-$ой строки ($j = i+1, i+2, \ldots, n$) $i-$ую строку умноженную на a_{ji}. После всех итераций получим:

$$\begin{cases} x_1 + c_{12}x_2 + \ldots + c_{1j}x_j + \ldots + c_{1n}x_n = d_1 \\ x_2 + \ldots + c_{2j}x_j + \ldots + c_{2n}x_n = d_2 \\ \ldots \\ x_i + \ldots c_{ij}x_j + \ldots + c_{in}x_n = d_i \\ \ldots \\ x_n = d_n \end{cases}$$

Обратный ход:

Обратный ход состоит в последовательном определении неизвестных из системы в обратном порядке:

$$\begin{cases} x_n = d_n \\ x_{n-1} = d_{n-1} - c_{n-1n}x_n \\ \ldots \\ x_i = d_i - \sum\limits_{j=i+1}^{n} c_{ij}x_j \end{cases}$$

Описанная выше процедура решения СЛАУ методом Гаусса может оказаться неустойчивой по отношению к случайным ошибкам, которые неизбежны при компьютерных расчетах в результате округления чисел из-за конечной длины машинного слова. Решает эту проблему метод Гаусса с выбором главного элемента. Он заключается в том, чтобы на $i-$ом шаге переставить столбцы i и j местами. Столбец j такой, что $|a_{ij}| = max(|a_{ik}|),\ \ k = i, \ldots, n.$

Определитель матрицы коэффициентов можно найти при приведении матрицы к треугольной форме. Определитель треугольной матрицы — произведение элементов на главной диагонали, что в нашем случае 1. При делении строки на число определитель также делится на это число. При нечётном количестве перестановок столбцов определитель следует умножить на -1.

Обратную матрицу можно найти методом Жордана-Гаусса. При приведении матрицы коэффициентов к треугольной форме и дальнейшем приведении к единичной будем производить те же элементарные преобразования с единичной матрицей. Тогда на месте единичной матрицы I окажется обратная матрица A^{-1}.

Число обусловленности матрицы А:

$$M_A = \|A\| \times \|A^{-1}\|$$

Будем использовать бесконечную матричную норму.

Тесты

Тест 1

СЛАУ:

$$\begin{cases} 2x_1 + 2x_2 - x_3 + x_4 = 4 \\ 4x_1 + 3x_2 - x_3 + 2x_4 = 6 \\ 8x_1 + 5x_2 - 3x_3 + 4x_4 = 12 \\ 3x_1 + 3x_2 - 2x_3 + 4x_4 = 6 \end{cases}$$

Точное решение:

$$x_1 = \frac{3}{5},\ x_2 = 1,\ x_3 = -1,\ x_4 = -\frac{1}{5}$$

Метод Гаусса:

$$x_1 = 0.6,\ x_2 = 1,\ x_3 = -1,\ x_4 = -0.2$$

Метод Гаусса с выбором главного элемента:

$$x_1 = 0.6,\ x_2 = 1,\ x_3 = -1,\ x_4 = -0.2$$

Определитель: 10 (точное значение — 10)

Число обусловленности: 60 (точное значение — 60)

Точная обратная матрица:

$$\begin{pmatrix} -0.6 & 0.1 & 0.3 & -0.2 \\ 1 & 0.5 & -0.5 & 0 \\ -1 & 1.5 & -0.5 & 0 \\ -0.8 & 0.3 & -0.1 & 0.4 \end{pmatrix}$$

Обратная матрица:

$$\begin{pmatrix} -0.6 & 0.1 & 0.3 & -0.2 \\ 1 & 0.5 & -0.5 & 0 \\ -1 & 1.5 & -0.5 & 0 \\ -0.8 & 0.3 & -0.1 & 0.4 \end{pmatrix}$$

Тест 2

СЛАУ:

$$\begin{cases} x_1 + x_2 + 3x_3 - 2x_4 = 1 \\ 2x_1 + 2x_2 + 4x_3 - x_4 = 2 \\ 3x_1 + 3x_2 + 5x_3 - 2x_4 = 1 \\ 2x_1 + 2x_2 + 8x_3 - 3x_4 = 2 \end{cases}$$

Определитель: 0 (точное значение — 0)

Программа выведет на stderr ошибку: incorrect input: det(A) = 0, и завершит программу.

Тест 3

СЛАУ:

$$\begin{cases} 2x_1 + 5x_2 - 8x_3 + 3x_4 = 8 \\ 4x_1 + 3x_2 - 9x_3 + x_4 = 9 \\ 2x_1 + 3x_2 - 5x_3 - 6x_4 = 7 \\ x_1 + 8x_2 - 7x_3 = 12 \end{cases}$$

Точное решение:

$$x_1 = 3,\ x_2 = 2,\ x_3 = 1,\ x_4 = 0$$

Метод Гаусса:

$$x_1 = 3,\ x_2 = 2,\ x_3 = 1,\ x_4 = 0$$

Метод Гаусса с выбором главного элемента:

$$x_1 = 3,\ x_2 = 2,\ x_3 = 1,\ x_4 = 0$$

Определитель: -189 (точное значение — -189)

Число обусловленности: 80.28571429 (точное значение — 80.2857)

Точная обратная матрица:

$$\begin{pmatrix} -1\frac{136}{189} & 1\frac{2}{9} & -\frac{124}{189} & \frac{163}{189} \\ -\frac{41}{63} & \frac{1}{3} & -\frac{17}{63} & \frac{32}{63} \\ -\frac{187}{189} & \frac{5}{9} & -\frac{76}{189} & \frac{106}{189} \\ -\frac{2}{27} & \frac{1}{9} & -\frac{5}{27} & \frac{2}{27} \end{pmatrix}$$

Обратная матрица:

$$\begin{pmatrix} -1.71957672 & 1.22222222 & -0.65608466 & 0.86243386 \\ -0.65079365 & 0.33333333 & -0.26984127 & 0.50793651 \\ -0.98941799 & 0.55555556 & -0.40211640 & 0.56084656 \\ -0.07407407 & 0.11111111 & -0.18518519 & 0.07407407 \end{pmatrix}$$

Тест 4

Матрица коэффициентов задается формулой:

$$A_{ij} = \begin{cases} \frac{i+j}{m+n}, & i \neq j \\ n + m^2 + \frac{j}{m} + \frac{i}{n}, & i = j \end{cases}$$

Правая часть задается формулой:

$$b_i = m \cdot i + n$$

Параметры: $n = 30$, $m = 20$

Решение, полученное методом Гаусса:

$x_1 = 0.09422086$, $x_2 = 0.13966464$, $x_3 = 0.18509927$, $x_4 = 0.23052474$, $x_5 = 0.27594107$, $x_6 = 0.32134824$, $x_7 = 0.36674627$, $x_8 = 0.41213516$, $x_9 = 0.45751490$, $x_{10} = 0.50288551$, $x_{11} = 0.54824699$, $x_{12} = 0.59359933$, $x_{13} = 0.63894255$, $x_{14} = 0.68427664$, $x_{15} = 0.72960160$, $x_{16} = 0.77491745$, $x_{17} = 0.82022418$, $x_{18} = 0.86552179$, $x_{19} = 0.91081029$, $x_{20} = 0.95608968$, $x_{21} = 1.00135996$, $x_{22} = 1.04662114$, $x_{23} = 1.09187322$, $x_{24} = 1.13711620$, $x_{25} = 1.18235009$, $x_{26} = 1.22757488$, $x_{27} = 1.27279058$, $x_{28} = 1.31799719$, $x_{29} = 1.36319472$, $x_{30} = 1.40838317$.

Решение, полученное методом Гаусса с выбором главного элемента:
$x_1 = 0.09422086$, $x_2 = 0.13966464$, $x_3 = 0.18509927$, $x_4 = 0.23052474$, $x_5 = 0.27594107$, $x_6 = 0.32134824$, $x_7 = 0.36674627$, $x_8 = 0.41213516$, $x_9 = 0.45751490$, $x_{10} = 0.50288551$, $x_{11} = 0.54824699$, $x_{12} = 0.59359933$, $x_{13} = 0.63894255$, $x_{14} = 0.68427664$, $x_{15} = 0.72960160$, $x_{16} = 0.77491745$, $x_{17} = 0.82022418$, $x_{18} = 0.86552179$, $x_{19} = 0.91081029$, $x_{20} = 0.95608968$, $x_{21} = 1.00135996$, $x_{22} = 1.04662114$, $x_{23} = 1.09187322$, $x_{24} = 1.13711620$, $x_{25} = 1.18235009$, $x_{26} = 1.22757488$, $x_{27} = 1.27279058$, $x_{28} = 1.31799719$, $x_{29} = 1.36319472$, $x_{30} = 1.40838317$.

Полученные решения довольно близки к точному решению данной СЛАУ.

Определитель:
110325917572619586901791736281355878259104412620680393433621559539316058247659520.0

Число обусловленности: 1.12212453

Из-за достаточного высокого порядка матрицы, привести обратную матрицу нет возможности.

Выводы

Метод Гаусса с выбором главного элемента давал более точные решения чем классический метод Гаусса. Данный алгоритм имеет сложность $O(n^3)$, поэтому программа работает довольно длительный промежуток времени. Поэтому можем сказать, что в случае небольших порядков матрицы выбор метода Гаусса отличный вариант.

Подвариант 2

Постановка задачи

Дана система уравнений Ax=f порядка n×n с невырожденной матрицей A. Написать программу численного решения данной системы линейных алгебраических уравнений (*n* – параметр программы), использующую итерационного метода верхней релаксации итерационный процесс имеет следующий вид:

$$(D + \omega A^{(-)})\frac{x^{k+1} - x^k}{\omega} + Ax^k = f,$$

где D, $A^{(-)}$ — соответсвенно диагональная и нижняя треугольная матрицы, k — номер текущей итерации, ω — итерационный параметр. Предусмотреть возможность задания элементов матрицы системы и ее правой части как во входном файле данных, так и путем задания специальных формул.

Цели и задачи практической работы

- Изучить метод верхней релаксации, используемый для численного решения систем линейных алгебраических уравнений;
- Изучить скорость сходимости этих методов в зависимости от выбора итерационного параметра ω;
- Разработать критерий остановки итерационного процесса, гарантирующий получение приближенного решения исходной СЛАУ с заданной точностью;
- Решить заданную СЛАУ методом верхней релаксации;
- Реализовать данные алгоритмы на любом языке программирования (в данном случае язык C);
- Провести тестирование программы;
- Правильность решения СЛАУ подтвердить системой тестов (используются ресурсы on-line системы http://www.wolframalpha.com).

Описание алгоритма решения

Пусть задана система линейных алгебраических уравнений:

$$\begin{cases} a_{11}x_1 \ + \ a_{12}x_2 \ + \ \dots \ + \ a_{1j}x_j \ + \ \dots \ + \ a_{1n}x_n = b_1 \\ a_{21}x_1 \ + \ a_{22}x_2 \ + \ \dots \ + \ a_{2j}x_j \ + \ \dots \ + \ a_{2n}x_n = b_2 \\ \dots \\ a_{i1}x_1 \ + \ a_{i2}x_2 \ + \ \dots \ + \ a_{ij}x_j \ + \ \dots \ + \ a_{in}x_n = b_i \\ \dots \\ a_{n1}x_1 \ + \ a_{n2}x_2 \ + \ \dots \ + \ a_{nj}x_j \ + \ \dots \ + \ a_{nn}x_n = b_n \end{cases}$$

Записывая в матричном виде:

$$Ax = b$$

Где A - невырожденная матрица и положительно определена.
Представим матрицу коэффициентов в виде суммы:

$$A \ = \ D \ + \ T_H \ + \ T_B$$

$$D_{ij} = \begin{cases} 0, & i \neq j \\ a_{ii}, & i = j \end{cases}$$

$$(T_H)_{ij} = \begin{cases} a_{ij}, & i > j \\ 0, & i \leq j \end{cases}$$

$$(T_B)_{ij} = \begin{cases} 0, & i \geq j \\ a_{ij}, & i < j \end{cases}$$

Запишем рекуррентное соотношение:

$$(D + \omega T_H)\frac{x_{k+1} - x_k}{\omega} + Ax_k = f$$

где k — номер текущей итерации, ω — итерационный параметр.
Соотношение для построения алгоритма вычисления очередной итерации:

$$\left(\frac{1}{\omega} + T_H\right) x_{k+1} \ + \ \left[\left(1 - \frac{1}{\omega}\right) D + T_B\right] x_k = f$$

В качестве критерия остановки итерационного процесса, гарантирующий получение решения исходной СЛАУ с заданной точностью ε, возьмем следующее условие:

$$|x_{k+1} - x_k| < \ \frac{\varepsilon}{2}$$

Тесты

В тестах 1-3 матрицы не являлись положительно определенными, поэтому в программе решается СЛАУ вида:

$$A^T A = A^T b$$

где A^T — транспонированная к A матрица. При этом матрица $A^T A$ является положительно определенной и решение данной СЛАУ совпадает с исходным.

Тест 1

СЛАУ:

$$\begin{cases} 2x_1 + 2x_2 - x_3 + x_4 = 4 \\ 4x_1 + 3x_2 - x_3 + 2x_4 = 6 \\ 8x_1 + 5x_2 - 3x_3 + 4x_4 = 12 \\ 3x_1 + 3x_2 - 2x_3 + 4x_4 = 6 \end{cases}$$

Точное решение:

$$x_1 = \frac{3}{5},\ x_2 = 1,\ x_3 = -1,\ x_4 = -\frac{1}{5}$$

Метод релаксации ($\omega = 0.6$, количество итераций — 1439) :

$x_1 = 0.60001238$, $x_2 = 0.99998486$, $x_3 = -0.99997295$, $x_4 = -0.19998482$

Метод релаксации ($\omega = 1.2$, количество итераций — 550) :

$x_1 = 0.59999705$, $x_2 = 1.00000729$, $x_3 = -0.99999827$, $x_4 = -0.20000251$

Метод релаксации ($\omega = 1.8$, количество итераций — 1947) :

$x_1 = 0.60000467$, $x_2 = 0.99999304$, $x_3 = -0.99999194$, $x_4 = -0.19999429$

Тест 2

СЛАУ:

$$\begin{cases} x_1 + x_2 + 3x_3 - 2x_4 = 1 \\ 2x_1 + 2x_2 + 4x_3 - x_4 = 2 \\ 3x_1 + 3x_2 + 5x_3 - 2x_4 = 1 \\ 2x_1 + 2x_2 + 8x_3 - 3x_4 = 2 \end{cases}$$

Определитель: 0 (точное значение — 0)

Программа выведет на stderr ошибку: incorrect input: det(A) = 0, и завершит программу.

Тест 3

СЛАУ:

$$\begin{cases} 2x_1 + 5x_2 - 8x_3 + 3x_4 = 8 \\ 4x_1 + 3x_2 - 9x_3 + x_4 = 9 \\ 2x_1 + 3x_2 - 5x_3 - 6x_4 = 7 \\ x_1 + 8x_2 - 7x_3 = 12 \end{cases}$$

Точное решение:

$$x_1 = 3,\ x_2 = 2,\ x_3 = 1,\ x_4 = 0$$

Метод релаксации ($\omega = 0.6$, количество итераций — 6663) :

$x_1 = 2.99982332$, $x_2 = 1.99993079$, $x_3 = 0.99990044$, $x_4 = -0.00001419$

Метод релаксации ($\omega = 1.2$, количество итераций — 1548) :

$x_1 = 2.99994995$, $x_2 = 1.99998023$, $x_3 = 0.99997174$, $x_4 = -0.00000402$

Метод релаксации ($\omega = 1.841238$, количество итераций — 188) :

$x_1 = 2.99993887$, $x_2 = 1.99997635$, $x_3 = 0.99996571$, $x_4 = -0.00000489$

Тест 4

Матрица коэффициентов задается формулой:

$$A_{ij} = \begin{cases} \frac{i+j}{m+n}, & i \neq j \\ n + m^2 + \frac{j}{m} + \frac{i}{n}, & i = j \end{cases}$$

Правая часть задается формулой:

$$b_i = m \cdot i + n$$

Метод релаксации ($\omega = 0.6$, количество итераций — 19) :

$x_1 = 0.09422087$, $x_2 = 0.13966465$, $x_3 = 0.18509928$, $x_4 = 0.23052476$, $x_5 = 0.27594108$, $x_6 = 0.32134825$, $x_7 = 0.36674628$, $x_8 = 0.41213517$, $x_9 = 0.45751491$, $x_{10} = 0.50288552$, $x_{11} = 0.54824700$, $x_{12} = 0.59359934$, $x_{13} = 0.63894256$, $x_{14} = 0.68427665$, $x_{15} = 0.72960161$, $x_{16} = 0.77491745$, $x_{17} = 0.82022418$, $x_{18} = 0.86552179$, $x_{19} = 0.91081029$, $x_{20} = 0.95608968$, $x_{21} = 1.00135996$, $x_{22} = 1.04662114$, $x_{23} = 1.09187322$, $x_{24} = 1.13711619$, $x_{25} = 1.18235008$, $x_{26} = 1.22757486$, $x_{27} = 1.27279056$, $x_{28} = 1.31799717$, $x_{29} = 1.36319469$, $x_{30} = 1.40838314$.

Метод релаксации ($\omega = 0.973$, количество итераций — 6) :

$x_1 = 0.09422086$, $x_2 = 0.13966464$, $x_3 = 0.18509927$, $x_4 = 0.23052474$, $x_5 = 0.27594107$, $x_6 = 0.32134824$, $x_7 = 0.36674627$, $x_8 = 0.41213516$, $x_9 = 0.45751490$, $x_{10} = 0.50288551$, $x_{11} = 0.54824699$, $x_{12} = 0.59359933$, $x_{13} = 0.63894255$, $x_{14} = 0.68427664$, $x_{15} = 0.72960160$, $x_{16} = 0.77491745$, $x_{17} = 0.82022418$, $x_{18} = 0.86552179$, $x_{19} = 0.91081029$, $x_{20} = 0.95608968$, $x_{21} = 1.00135996$, $x_{22} = 1.04662114$, $x_{23} = 1.09187322$, $x_{24} = 1.13711620$, $x_{25} = 1.18235009$, $x_{26} = 1.22757488$, $x_{27} = 1.27279058$, $x_{28} = 1.31799719$, $x_{29} = 1.36319472$, $x_{30} = 1.40838316$.

Метод релаксации ($\omega = 1.8$, количество итераций — 86) :

$x_1 = 0.09422086$, $x_2 = 0.13966464$, $x_3 = 0.18509927$, $x_4 = 0.23052475$, $x_5 = 0.27594107$, $x_6 = 0.32134825$, $x_7 = 0.36674628$, $x_8 = 0.41213516$, $x_9 = 0.45751491$, $x_{10} = 0.50288552$, $x_{11} = 0.54824700$, $x_{12} = 0.59359935$, $x_{13} = 0.63894256$, $x_{14} = 0.68427665$, $x_{15} = 0.72960162$, $x_{16} = 0.77491746$, $x_{17} = 0.82022419$, $x_{18} = 0.86552180$, $x_{19} = 0.91081030$, $x_{20} = 0.95608969$, $x_{21} = 1.00135997$, $x_{22} = 1.04662115$, $x_{23} = 1.09187323$, $x_{24} = 1.13711621$, $x_{25} = 1.18235009$, $x_{26} = 1.22757488$, $x_{27} = 1.27279058$, $x_{28} = 1.31799719$, $x_{29} = 1.36319472$, $x_{30} = 1.40838316$.

Выводы

Метод верхней релаксации оказался значительно быстрее при довольно большом порядке матрицы, однако на матрицах четвертого порядка производил гораздо больше итераций. Также стоит отметить, что скорость сходимости сильно зависит от параметра ω. Точность этого метода ниже, чем у метода Гаусса. Также этот метод применим только к положительно определенным матрицам, что накладывает ещё большие условия на применимость. Однако в некоторых задачах метод верхней релаксации может быть полезен.

Код программы

Код для двух подвариантов написан в одной программе. Пользователь может выбрать ввести матрицу из файла или задать формирование ее по формуле. Также можно задать параметры m и n во время выполнения программы. Пользователь может выбрать какой из подвариантов запустить (предусмотрен запуск обоих подвариантов). При запуске метода верхней релаксации можно во время выполнения программы задать ω и ε.
Использованный язык — C

```
#include <stdio.h>
#include <math.h>


enum
{
    MAX_ELEM = 30
};


long double A[MAX_ELEM][MAX_ELEM]; //изначальная матрица
long double A_g1[MAX_ELEM][MAX_ELEM]; //метода Гаусса
long double A_g2[MAX_ELEM][MAX_ELEM]; //метода Гаусса с
выбором главного элемента
long double A_T[MAX_ELEM][MAX_ELEM]; //транспонированная
матрица
long double A_up_rel[MAX_ELEM][MAX_ELEM]; //A_T * A
long double b[MAX_ELEM]; //правая часть
long double b_g1[MAX_ELEM]; //метода Гаусса
long double b_g2[MAX_ELEM]; //метода Гаусса с выбором
главного элемента
long double b_up_rel[MAX_ELEM]; //A_T * b
int x_rev[MAX_ELEM]; //перестановка решений
long double x_prev[MAX_ELEM]; //x_k
long double x_now[MAX_ELEM]; //x_{k + 1}
long double w; //\omega
```

```
long double eps; //\epsilon
int n, m; // порядок, параметр
int iter; // кол-во итераций
long double A_reverse[MAX_ELEM][MAX_ELEM]; // обратная
матрица
long double det_A = 1; //определитель

void input_var2(void) { //формула input параметров
    while (1) {
        printf("Please enter Matrix Order\n");
        scanf("%d", &n);
        printf("and Parameter m\n");
        scanf("%d", &m);
        if (n <= 0) {
            fprintf(stderr, "incorrect input\n");
            continue;
        }
        if (m == 0) {
            fprintf(stderr, "incorrect input\n");
            continue;
        }
        if (m == -n) {
            fprintf(stderr, "incorrect input\n");
            continue;
        }
        break;
    }
}

void fill_formula(void) { //формирование СЛАУ по формуле
    for (int i = 1; i <= n; ++i) {
        for (int j = 1; j <= n; ++j) {
            A[i - 1][j - 1] = ((i == j) ? (n + m * m +
```

```
((long double)j) / m + ((long double)i) / n) : ((long
double)(i + j) / (m + n)));
        }
    }

    for (int i = 1; i <= n; ++i) {
        b[i - 1] = m * i + n;
    }
}

void fill_file(void) { //формирование СЛАУ из файла
    FILE *fd_matrix, *fd_right;
    while (1) {
        char s[256];
        printf("Please enter the filename for the
matrix\n");
        scanf("%s", s);
        fd_matrix = fopen(s, "r");
        if (fd_matrix == NULL) {
            fprintf(stderr, "incorrect input\n");
            continue;
        }
        break;
    }

    while (1) {
        char s[256];
        printf("Please enter the filename for the right
side\n");
        scanf("%s", s);
        fd_right = fopen(s, "r");
        if (fd_right == NULL) {
            fprintf(stderr, "incorrect input\n");
```

```
            continue;
        }
        break;
    }

    int tempor_buf[MAX_ELEM * MAX_ELEM];
    n = 0;
    while (fscanf(fd_matrix, "%d", tempor_buf + n) != EOF)
{
        ++n;
    }

    double n_vr = sqrt(n);
    if (n_vr == (int) n_vr) {
        n = n_vr;
    } else {
        fprintf(stderr, "incorrect input\n");
        fclose(fd_matrix);
        fclose(fd_right);
        fill_file();
    }

    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) {
            A[i][j] = tempor_buf[n * i + j];
        }
    }

    int prav_b;

    for (int i = 0; i < n; ++i) {
        if (fscanf(fd_right, "%d", &prav_b) == EOF) {
            fprintf(stderr, "incorrect input\n");
```

```
            fclose(fd_matrix);
            fclose(fd_right);
            fill_file();
        }
        b[i] = prav_b;
    }

    fclose(fd_matrix);
    fclose(fd_right);
}

void sub_vec(long double av[MAX_ELEM], long double
bv[MAX_ELEM], long double vr) { //разность векторов
    for (int i = 0; i < n; ++i) {
        bv[i] -= vr * av[i];
    }
}

void triangolize(long double At[MAX_ELEM][MAX_ELEM], long
double bt[MAX_ELEM], int flag) { //приведение к треугольной
форме
    for (int i = 0; i < n; ++i) {
        long double vr = At[i][i];
        bt[i] /= vr;
        for (int j = i; j < n; ++j) {
            At[i][j] /= vr;
        }
        if (flag) {
            for (int j = 0; j < n; ++j) {
                A_reverse[i][j] /= vr;
            }
        }
```

```
        for (int j = i + 1; j < n; ++j) {
            bt[j] -= At[j][i] * bt[i];
            if (flag) {
                sub_vec(A_reverse[i], A_reverse[j], At[j]
[i]);
            }
            sub_vec(At[i], At[j], At[j][i]);
        }
    }
}

long double abs_d(long double a) { //абсолютное значение
числа типа long double
    if (a > 0) {
        return a;
    } else {
        return -a;
    }
}

int max_elem(long double av[MAX_ELEM]) { //поиск индекса с
максималоьным абсолютным значением
    int ma_uk = 0;
    for (int i = 1; i < n; ++i) {
        if (abs_d(av[i]) > abs_d(av[ma_uk])) {
            ma_uk = i;
        }
    }
    return ma_uk;
}

void swap_column(long double At[MAX_ELEM][MAX_ELEM], int
uk, int left) { //преставить местами столбцы
    for (int i = 0; i < n; ++i) {
```

```
        long double vr =  At[i][uk];
        At[i][uk] = At[i][left];
        At[i][left] = vr;
    }
}

void triangolize_with_melem(long double At[MAX_ELEM]
[MAX_ELEM], long double bt[MAX_ELEM]) { //приведение к
треугольной форме с главным элементом
    int ch_nch = 0;
    for (int i = 0; i < n; ++i) {
        int uk = max_elem(At[i]);
        int vr_rev = x_rev[i];
        x_rev[i] = x_rev[uk];
        x_rev[uk] = vr_rev;
        ch_nch = (ch_nch + 1) % 2;
        swap_column(At, uk, i);

        long double vr = At[i][i];
        if (vr == 0) {
            det_A = 0;
            break;
        }
        det_A *= vr;
        bt[i] /= vr;
        for (int j = i; j < n; ++j) {
            At[i][j] /= vr;
        }

        for (int j = i + 1; j < n; ++j) {
            bt[j] -= At[j][i] * bt[i];
            sub_vec(At[i], At[j], At[j][i]);
        }
```

```
    }
    if (ch_nch) {
        det_A *= -1;
    }
}

void rever_proc(long double At[MAX_ELEM][MAX_ELEM], long double bt[MAX_ELEM], int flag) { //обратный ход
    for (int i = n - 1; i > 0; --i) {
        for (int j = i - 1; j >= 0; --j) {
            bt[j] -= At[j][i] * bt[i];
            if (flag) {
                sub_vec(A_reverse[i], A_reverse[j], At[j][i]);
            }
            sub_vec(At[i], At[j], At[j][i]);

        }
    }
}

void gauss(void) { //метод Гаусса
    triangolize(A_g1, b_g1, 1);
    rever_proc(A_g1, b_g1, 1);
}

void gauss_with_melem(void) { //метод Гаусса с главным элементом
    triangolize_with_melem(A_g2, b_g2);
    if (det_A == 0) {
        return;
    }
    rever_proc(A_g2, b_g2, 0);
```

```
}

long double matrix_norm(long double At[MAX_ELEM][MAX_ELEM])
{ //бесконечная матричная норма
    long double ans = 0;
    for (int i = 0; i < n; ++i) {
        long double vr = 0;
        for (int j = 0; j < n; ++j) {
            vr += abs_d(At[i][j]);
        }
        if (vr > ans) {
            ans = vr;
        }
    }

    return ans;
}

long double cond_num(void) { //нахождение обусловленности
матрицы
    long double ans = matrix_norm(A) *
matrix_norm(A_reverse);
    return ans;
}

long double vector_norm(long double av[MAX_ELEM], long
double bv[MAX_ELEM]) { //векторная норма
    long double ans = 0;
    for (int i = 0; i < n; ++i) {
        ans += abs_d(av[i] - bv[i]);
    }
    return ans;
}
```

```
void upper_relaxation(void) { //метод верхней релаксации
    do {
        ++iter;
        for (int i = 0; i < n; ++i) {
            x_prev[i] = x_now[i];
        }
        for (int i = 0; i < n; ++i) {
            long double sum1 = 0, sum2 = 0;
            for (int j = 0; j < i; ++j) {
                sum1 += A_up_rel[i][j] * x_now[j];
            }
            for (int j = i; j < n; ++j) {
                sum2 += A_up_rel[i][j] * x_prev[j];
            }

            x_now[i] = x_prev[i] + w * (b_up_rel[i] - sum1
- sum2) / A_up_rel[i][i];
        }
    } while (vector_norm(x_prev, x_now) > eps / 2);
}

void matrix_transpose(long double At[MAX_ELEM][MAX_ELEM]) {
//транспонирование матрицы
    for (int i = 0; i < n; ++i) {
        for (int j = i; j < n; ++j) {
            long double vr = At[i][j];
            At[i][j] = At[j][i];
            At[j][i] = vr;
        }
    }
}

void mul_matrix(long double At[MAX_ELEM][MAX_ELEM], long
```

```
double Bt[MAX_ELEM][MAX_ELEM], long double Ct[MAX_ELEM]
[MAX_ELEM]) { //умножение матриц
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) {
            for (int k = 0; k < n; ++k) {
                Ct[i][j] += At[i][k] * Bt[k][j];
            }
        }
    }
}

void mul_matrix_vec(long double At[MAX_ELEM][MAX_ELEM],
long double Bt[MAX_ELEM], long double Ct[MAX_ELEM]) { //
умножение матрицы на вектор
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) {
            Ct[i] += At[i][j] * Bt[j];
        }
    }
}

int main (void) {
    int inp_var;
    while (1) {
        printf("You have chosen to set the elements of the
system matrix and its right side in the input file (press
1),"
               " or by setting a special formula (press
2)\n");
        scanf("%d", &inp_var);
        if (inp_var == 1) {
            fill_file();
            break;
        } else if (inp_var == 2) {
```

```
        input_var2();
        fill_formula();
        break;
    } else {
        fprintf(stderr, "incorrect input\n");
    }
}

for (int i = 0; i < n; ++i) {
    A_reverse[i][i] = 1;
}

for (int i = 0; i < n; ++i) {
    x_rev[i] = i;
    for (int j = 0; j < n; ++j) {
        A_g1[i][j] = A[i][j];
        A_g2[i][j] = A[i][j];
        A_T[i][j] = A[i][j];
    }
    b_g1[i] = b[i];
    b_g2[i] = b[i];
}

gauss_with_melem();
if (det_A == 0) {
    fprintf(stderr, "incorrect input: det(A) = 0\n");
    return 1;
}

int var_method;
while (1) {
    printf("If you want to use the Gauss method press 1\n"
```

```
           "if you want to use the upper relaxation method
press 2\n"
           "if you want to use both methods press 3\n");
    scanf("%d", &var_method);
    if (var_method < 1 || var_method > 3) {
        fprintf(stderr, "incorrect input\n");
        continue;
    }
    break;
}

if (var_method == 1 || var_method == 3) {
    gauss();

    printf("Gauss method:\n");
    for (int i = 0; i < n; ++i) {
        printf("x%d = %.8Lf ", i + 1, b_g1[i]);
    }
    printf("\n");

    printf("Gauss method with main elem choice:\n");
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) {
            if (x_rev[j] == i) {
                printf("x%d = %.8Lf ", i + 1, b_g2[j]);
            }
        }
    }
    printf("\n");

    printf("Determinant: %.8Lf\n", det_A);
    printf("Inverses matrix:\n");
    for (int i = 0; i < n; ++i) {
```

```
            for (int j = 0; j < n; ++j) {
                printf("%.8Lf ", A_reverse[i][j]);
            }
            printf("\n");
        }
        printf("Condition number: %.8Lf\n", cond_num());
    }

    if (var_method > 1) {
        matrix_transpose(A_T);
        mul_matrix(A_T, A, A_up_rel);
        mul_matrix_vec(A_T, b, b_up_rel);

        while (1) {
            printf("Please enter Parameter for upper
relaxation method\n");
            scanf("%Lf", &w);
            if (w == 0) {
                fprintf(stderr, "incorrect input\n");
            } else {
                break;
            }
        }

        while (1) {
            printf("Please enter accuracy Parameter for
upper relaxation method\n");
            scanf("%Lf", &eps);
            if (eps > 0) {
                break;
            } else {
                fprintf(stderr, "incorrect input\n");
            }
```

```
        }

        upper_relaxation();

        printf("Upper relaxation method\niterations: %d\n",
iter);
        for (int i = 0; i < n; ++i) {
            printf("x%d = %.8Lf ", i + 1, x_now[i]);
        }
        printf("\n");
    }

    return 0;
}
```

Список литературы

[1] Костомаров Д. П., Фаворский А. П. Вводные лекции по численным методам: Учеб. Пособие. - М.: Университетская книга, Логос, 2006.
[2] Тыртышников,Евгений Евгеньевич (). Основы алгебры : учеб. для студентов вузов. - М.: Физматлит, 2020