**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №5**

**по дисциплине «Вычислительная математика»**

Тема: Метод Гаусса

|  |  |  |
| --- | --- | --- |
| Студент гр. 3342 |  | Лодыгин И. А. |
| Преподаватель |  | Пуеров Г.Ю. |

Санкт-Петербург

2025

## Задание

В ходе выполнения работы студенты должны найти решение системы линейных уравнений с n неизвестными, заданной матрицей коэффициентов и вектором свободных членов , методом Гаусса. Выполнение работы состоит из следующих этапов:

1) с помощью преподавателя определить систему уравнений, которую нужно решить;

2) для решения системы уравнений разработать программу на языке C++, использующую подпрограмму-функцию. Данная функция имеет следующие параметры:

- a - матрица коэффициентов системы уравнений размера , тип ;

- - вектор свободных членов размера , тип ;

- - выходной вектор результата решения размера , тип ;

- - размер системы (матрицы a и вектора свободных членов ), тип.

В разрабатываемой программе должна быть описана константа nmax, равная максимальным размерам используемых матриц и векторов. Функция GAUSS в качестве значения типа возвращает:

а) 0 - в случае нормального завершения процесса вычисления;

б) 1 - в случае вырожденности матрицы а;

в) 2 - если n < 2;

г) 3 - если n > n\_max.

Провести вычисления с использованием разработанной программы и исследовать обусловленность задачи с использованием пакета Matlab, при этом для определения числа обусловленности матрицы A рекомендуется использовать функцию cond(A) [14]. Кроме того, для проверки получаемых результатов можно провести вычисления с помощью пакетов Matlab и Derive

**Теоретическая часть**

Рассматривается система линейных уравнений n-го порядка Ax=b. Суть метода исключения по главным элементам (метод Гаусса) заключается в следующем. Находится наибольший по абсолютной величине коэффициент. Для исключения из i-го уравнения необходимо умножить k-е уравнение на и вычесть его из i-го уравнения, после чего процесс повторяется для исключения другого неизвестного из оставшихся -1 уравнений и т. д. В результате система приводится к треугольному виду, из которого легко находятся неизвестные . Процесс приведения системы к треугольному виду называется прямым ходом, а нахождение неизвестных - обратным ходом метода Гаусса.

**Использования библиотеки Eigen для анализа результатов**

Для анализа и решения системы уравнений подойдет библиотека C++ Eigen. Она помогает через встроенные методы вычислить число обусловленности через сингулярные числа, реальное решение системы линейных уравнений и нормы матриц.

float computeConditionNumber(const Eigen::MatrixXf& eigenMatrix) — считает число обусловленности через формулу для спектральной нормы. ||A||\_2 \* ||A^-1||\_2=b\_max/b\_min (максимальное и минимальное сингулярные числа матрицы A). Сингулярное число — число, которое показывает масштаб преобразования, выполняемого этой матрицей. Или растягивает вектор, или сужает.

Eigen::VectorXf solveLinearSystem(const Eigen::MatrixXf& eigenA, const Eigen::VectorXf& eigenB) — через встроенную функцию возвращает решение системы.

float computeRelativeResidual(const Eigen::MatrixXf& eigenA, const Eigen::VectorXf& eigenB, const Eigen::VectorXf& eigenX) — считает относительную невязку. ||Ax — b|| / ||b||.

**Написание программы для реализации алгоритма хорд**

Реализуем алгоритм на языке C++. Для этого напишем некоторые функции.

int gauss(std::vector<std::vector<float>>& a, std::vector<float>& b, std::vector<float>& X, int n) – сам алгоритм Гаусса. Заполнять матрицу корней будем через переданную ссылку на X.

Запускаем преобразование системы. Начинаем со стобца 1. Мы среди всех строк выбираем ту, которая содержит в 1-ом столбце максимальный по модулю элемент, меняем его местами с 1-ой строкой. После этого из каждого уравнения i ниже вычитаем 1-ую строку, умноженную на a\_ij / a\_00. Таким образом мы получили тождественное уравнение, у которой все строки ниже 1-ой содержат 0 в 1-ом столбце. Закрепляем эту 1-ую строку и аналогично считаем для 2-ой строки и 2-ого столбца.

Также будем параллельно проверять матрицу на вырожденность, когда мы находим максимальный элемент в столбце k, начиная со строки k. И если у нас все элементы были равны 0, то после преобразования матрицы в треугольную мы получим, что a[k][k] = 0. А так как определитель треугольной матрицы равен произведению элементов из диагонали, то мы получим итоговое значение 0. Матрица в таком случае будет вырожденной.

По итогу мы придем к нужной нам системе. Откуда легко находятся все корни. X[k] = (b[k] + сумма(i от k не включительно до n - 1) [(-1) \* a[k][i] \* X[i]]) / a[k][k].

void randomMatrix(std::vector<std::vector<float>>& A, std::vector<float>& b, int n) – функция для создания рандомной матрицы

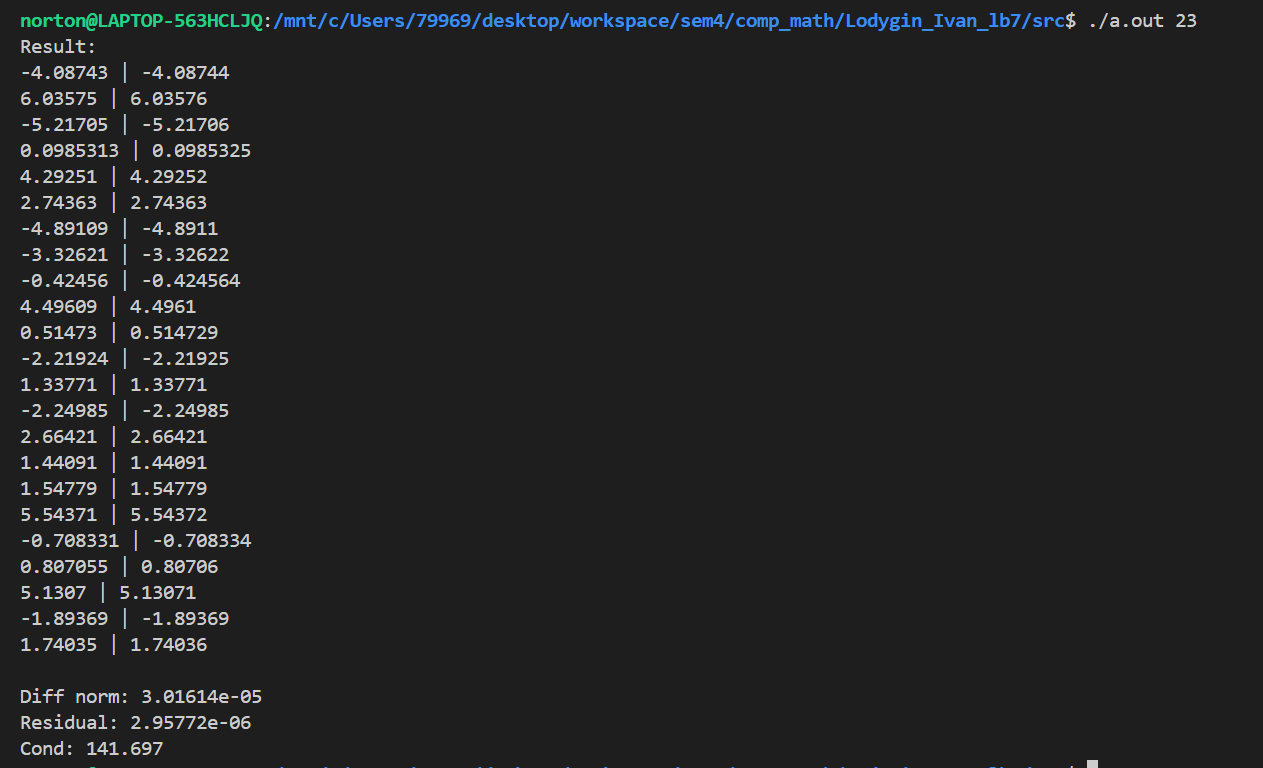
Тестирование программы приведено в таблице 1.

Таблица 1 – Результаты тестирования

|  |  |  |
| --- | --- | --- |
| № п/п | Входные данные | Выходные данные |
|  | 3 | Result:  1.005 | 1.005  0.0764036 | 0.0764036  0.0541682 | 0.0541683 |
|  | 18 | Result:  -0.273905 | -0.273905  0.022049 | 0.0220487  -0.392769 | -0.392768  0.3871 | 0.3871  0.646071 | 0.64607  -0.09551 | -0.0955097  0.00168442 | 0.00168432  0.435137 | 0.435137  -0.369664 | -0.369664  0.256811 | 0.256811  -0.429943 | -0.429943  0.728554 | 0.728554  1.25439 | 1.25439  0.435584 | 0.435584  0.461854 | 0.461853  -0.419642 | -0.419641  0.892678 | 0.892678  1.55581 | 1.55581 |
|  | 15 | Result:  -0.878414 | -0.878415  -2.26955 | -2.26956  0.681449 | 0.68145  4.01816 | 4.01817  -4.53427 | -4.53427  0.877925 | 0.877924  1.04463 | 1.04463  1.41381 | 1.41382  -1.02655 | -1.02655  2.28025 | 2.28025  2.32252 | 2.32252  -5.92978 | -5.92979  0.131222 | 0.131222  2.98766 | 2.98766  6.15323 | 6.15323 |
|  | 23 | Result:  -4.21401 | -4.21401  0.188336 | 0.188337  -1.09851 | -1.0985  1.75103 | 1.75103  -1.28781 | -1.28781  -0.374769 | -0.374767  0.854684 | 0.854683  -1.6362 | -1.6362  3.81558 | 3.81558  -4.12081 | -4.1208  0.0795325 | 0.0795317  1.20036 | 1.20036  1.1763 | 1.1763  0.54134 | 0.54134  1.79565 | 1.79564  1.43425 | 1.43425  -1.22793 | -1.22793  -1.15361 | -1.1536  0.988227 | 0.988225  -3.60299 | -3.60299  1.97439 | 1.97439  -0.359657 | -0.359657  1.76138 | 1.76138  -1.1624  -0.149487  1.24914  -0.661199  0.273639 |

**Принцип работы с данной программой**

Программа состоит из 2 файлов, main.cpp (ввод/вывод данных и создание матрицы) и gauss.cpp (сам алгоритм). Компилируется и запускается main.cpp. И по переданному n строит матрицу n\*n и выдает корни для этой системы. А справа от найденного корня выводится корень, который действительно соответствует этой системе, то есть посчитан по встроенной программе.



После этого выводятся число обусловленности матрицы и относительная невязка.

**Исследование обусловленности**

Число обусловленности матрицы определяет чувствительность решения системы линейных уравнений к погрешностям исходных данных. Он отражает, насколько чувствительна функция к изменениям или ошибкам на входе и насколько ошибка на выходе является результатом ошибки на входе.

Матрицы с большим числом обусловленности называются плохо обусловленными, и наоборот, матрицы с малым значением — хорошо обусловленными. Для плохо обусловленных матриц малые возмущения её элементов приводят к значительному изменению решения.

Влияние обусловленности попробуем оценить через невязку.

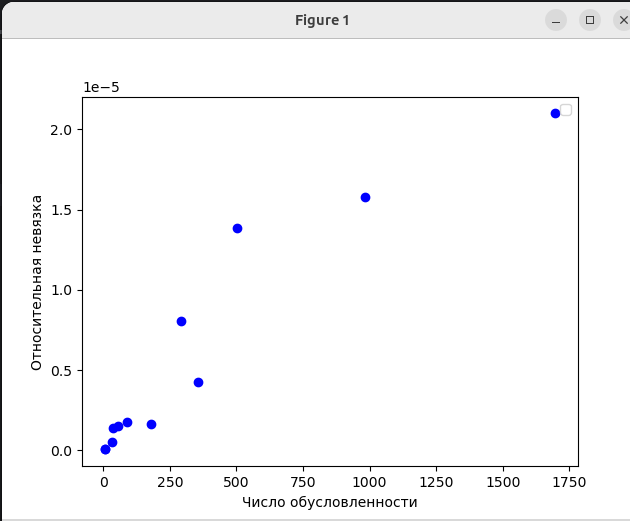
Результаты исследования см. в таблице 2.

Таблица 2 – Результаты влияния обусловленности на итоговую неточность

|  |  |  |
| --- | --- | --- |
| Residual | n | Cond |
| 5.72586e-08 | 3 | 4.96234 |
| 6.03367e-08 | 3 | 6.04762 |
| 5.44321e-07 | 18 | 34.0385 |
| 1.37319e-06 | 23 | 36.2191 |
| 1.54227e-06 | 15 | 55.6539 |
| 1.77647e-06 | 40 | 88.847 |
| 1.63378e-06 | 39 | 181.099 |
| 8.06121e-06 | 71 | 292.225 |
| 4.24093e-06 | 134 | 354.177 |
| 1.38845e-05 | 77 | 502.653 |
| 1.58188e-05 | 59 | 980.932 |
| 2.10004e-05 | 66 | 1697.34 |

По данным результатам можем заметить следующее. Выбор размера не влияет никак на обусловленность и относительную невязку. Это зависит от того, какие элементы матрицы будут выбраны.

Также отметим, что большая обусловленность порождает большую невязку. Нельзя точно сказать, что если одна обусловленность больше другой, то невязка будет иметь подобное отношение, так как на этой влиет множество других параметров, которые здесь не учитываются. Однако в большинство случаев именно такая ситуация и наблидается: на множестве систем с маленькой обусловленностью невязка будет маленького масштаба, а на множестве с большой обусловленностью – невязка будет более крупного масшатаба.



## Вывод

В ходе лабораторной работы была написана программа на языке C++, выполняющая решение системы линейных уравнений через метод Гаусса. Программа получает на вход некое число n и по нему строит случайную матрицу коэффициентов n\*n и свободных членов. Матрица проверяется на вырожденность и, если это возможно, решает систему, выдавая окончательный ответ.

Метод был протестирован с помощью технологии Eigen, которая выдает точный результат решения системы. Тестирование показало, что метод показывает хорошую работоспособность.

Также алгоритм был исследован на обусловленность. То есть мы оценили, насколько погрешность входных данных может повлиять на решение системы линейных уравнений. Это мы увидели через относительную невязку — различие настоящих данных и получившихся. Мы пришли к выводу, что число обусловленности влияет на эту невязку: большие числа обусловленности пораждают большие по масштабу эти невязки.