НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Факультет ПИиКТ

Отчёт по лабораторной работе на тему:

Решение СЛАУ

Метод Гаусса

Выполнил студент

Агнистова Алина Юрьевна

Группа № P3225

Преподаватель: Перл Ольга Вячеславовна

г. Санкт-Петербург

2024

**Описание метода и расчётные формулы**

Метод Гаусса – метод решения СЛАУ (систем линейных алгебраических уравнений), заключающийся в последовательном исключении неизвестных из уравнения с помощью элементарных преобразований строк (приведение матрицы к треугольному виду).

Прямой ход метода Гаусса – поочерёдное преобразование уравнений системы для избавления от переменных неизвестных. На этом этапе матрица системы будет приведена к треугольному виду.

Обратный ход метода Гаусса – последовательное вычисление искомых неизвестных от последнего уравнения к первому.

Метод Гаусса применим только в случае совместности системы.

СЛАУ может иметь:

* единственное решение;
* бесконечно много решений;
* 0 решений.

СЛАУ считается совместной, когда она имеет единственное решение. По теореме Кронекера-Капелли СЛАУ совместна тогда и только тогда, когда ранг матрицы коэффициентов системы равен рангу расширенной матрицы .

Ранг матрицы – число ненулевых строк ступенчатой матрицы.

Прямой ход в методе реализуется согласно формулам:

Формулы применяются последовательно и пошагово преобразуют матрицу системы к треугольному виду.

Обратный ход реализуется по формуле:

Таким образом на этом этапе будут вычислены все неизвестные.

После того, как решение найдено следует оценить его погрешность. Одна из величин, характеризующая степень отклонения полученного решения от точного – невязка.

Невязка – разность между правой и левой частями уравнений при подстановке в них полученных неизвестных. Важно учитывать, что при расчёте невязки рассматривается исходная матрица системы, а не преобразованная методом Гаусса. Определяется по формуле:

**Блок-схема реализованного метода**

Изображение выглядит как снимок экрана, черно-белый, черный, линия

Автоматически созданное описание

**Код метода**

#include <iostream>

#include <vector>

#include <iomanip>

#include <cmath>

**using** **namespace** std;

**bool** isSolutionExists = **true**;

string errorMessage = "";

**bool** hasSolution(**int** n, vector<vector<**double**>> matrix) {

**for** (**int** i = 0; i < n; i++) {

**int** zeroes = 0;

**for** (**int** j = 0; j < n; j++) {

**if** (matrix[i][j] == **double**(0)) {

zeroes++;

}

}

**if** (zeroes == n && matrix[i][n] != **double**(0)) {

**return** **false**;

}

**else** **if** (zeroes && matrix[i][n] == **double**(0)) {

**return** **false**;

}

}

**return** **true**;

}

vector<**double**> solveByGauss(**int** n, vector<vector<**double**>> matrix) {

vector<**double**> x(n);

vector<**double**> r(n);

vector<vector<**double**>> originalMatrix = matrix;

**double** d = 0;

//Прямой ход

**for** (**int** k = 0; k < n; k++) {

**double** maxEl = fabs(matrix[k][k]);

**int** idx = k;

**for** (**int** i = k + 1; i < n; i++){

**if** (fabs(matrix[i][k]) > maxEl){

maxEl = fabs(matrix[i][k]);

idx = i;

}

}

**if** (idx != k){

swap(matrix[k], matrix[idx]);

}

**for** (**int** j = k + 1; j < n; j++) {

d = matrix[j][k] / matrix[k][k];

**for** (**int** i = k; i <= n; i++) {

matrix[j][i] = matrix[j][i] - d \* matrix[k][i];

}

}

}

**if** (!hasSolution(n, matrix)) {

isSolutionExists = **false**;

errorMessage = "The system has no roots of equations or has an infinite set of them.";

**return** {};

}

//Обратный ход

**double** s;

**for** (**int** k = n - 1; k >= 0; k--) {

s = 0;

**for** (**int** j = k + 1; j < n; j++) {

s += matrix[k][j] \* x[j];

}

x[k] = (matrix[k][n] - s) / matrix[k][k];

}

//Расчёт невязок

**for** (**int** i = 0; i < n; i++) {

**double** a = 0.0;

**for** (**int** j = 0; j < n; j++) {

a = a + originalMatrix[i][j] \* x[j];

}

r[i] = originalMatrix[i][n] - a;

}

x.insert(x.end(), r.begin(), r.end());

**return** x;

}

**Примеры и результаты работы программы**

Во всех примерах расчёт невязок не входит в сравнение результатов, так как в нашем случае невязки зависят от написанного программного кода.

Пример 1

Выбранный случай: Матрица, имеющая единственное решение.

Предполагаемый результат:

|  |
| --- |
| 2 |
| 3 |
| -1 |

Входные данные:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| a | | | b | n |
| 2 | 1 | -1 | 8 | 3 |
| -3 | -1 | 2 | -11 |
| -2 | 1 | 2 | -3 |

Выходные данные:

|  |
| --- |
| 2 |
| 3 |
| -1 |

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Пример 2

Выбранный случай: Несовместная матрица.

Предполагаемый результат: Сообщение о невозможности вычисления.

Входные данные:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| a | | | b | n |
| 1 | 1 | 1 | 3 | 3 |
| 2 | 2 | 2 | 6 |
| 1 | -1 | 1 | 2 |

Выходные данные: Сообщение о невозможности вычисления.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Пример 3

Выбранный случай: Матрица с бесконечным числом решений.

Предполагаемый результат: Сообщение о невозможности вычисления.

Входные данные:

|  |  |  |  |
| --- | --- | --- | --- |
| a | | b | n |
| 1 | 2 | 3 | 2 |
| 2 | 4 | 6 |

Выходные данные: Сообщение о невозможности вычисления.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Пример 4

Выбранный случай: Матрица, где в каждом строке только один ненулевой элемент.

Предполагаемый результат:

|  |
| --- |
| 4 |
| 3 |
| 2 |

Входные данные:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| a | | | b | n |
| 0 | 0 | 1 | 2 | 3 |
| 0 | 1 | 0 | 3 |
| 1 | 0 | 0 | 4 |

Выходные данные:

|  |
| --- |
| 4 |
| 3 |
| 2 |

Изображение выглядит как снимок экрана, текст, черный

Автоматически созданное описание

Пример 5

Выбранный случай: Матрица с нулевым коэффициентом при ведущем элементе.

Предполагаемый результат:

|  |
| --- |
| 0 |
| -1 |
| 2 |

Входные данные:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| a | | | b | n |
| 0 | 2 | 3 | 4 | 3 |
| 5 | 6 | 7 | 8 |
| 9 | 10 | 11 | 12 |

Выходные данные:

|  |
| --- |
| 2.1711e-15 |
| -1 |
| 2 |

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание

Проанализировав результаты запуска реализованного метода, можно сделать вывод, что код некорректно работает при случае матрицы с нулевым коэффициентом при ведущем элементе (пример 5), что вызвано точности вычислений с помощью программы, так как полученный результат можно считать числом равным нулю.

Для оценки метода приведена таблица сравнения метода Гаусса с методом Гаусса с выбором главного элемента и методом Гаусса-Зейделя:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Метод Гаусса | Метод Гаусса с выбором главного элемента | Метод Гаусса-Зейделя |
| Описание | Прямой метод, приводящий матрицу к треугольному виду и последующим обратным ходом для нахождения решений. | Отличен от метода Гаусса тем, что на каждом шаге выбирается максимальный (абсолютно) элемент в столбце для повышения численной стабильности. | Итерационный метод, использующий предыдущие приближения для вычисления новых значений. |
| Вычислительная сложность | Средняя (эффективен для небольших и средних систем, возможны ошибки округления) | Средняя (численно более стабилен, но требует дополнительных вычислений) | Высокая (требует хорошего начального приближения) |
| Применимость | Системы с точными данными. | Системы, где требуется высокая численная стабильность. | Системы, где допустимы приближённые решения. |
| Численная стабильность | Возможны проблемы округления. | Высокая | Может быть нестабильным для некоторых типов матриц. |

Таким образом, метод Гаусса будет оптимальным в случаях, когда допускается численная нестабильность и расчёты ведутся на системах маленького и среднего размеров. В случаях, когда численная стабильность должна быть высокой лучше отдать предпочтение методу Гаусса с выбором главного элемента. Метод Гаусса-Зейделя же чаще всего применяется для больших и разреженных систем, где использование итерационных методов и приближённых решений является допустимым.

Метод Гаусса – классический метод решения СЛАУ. Метод сохраняет высокий уровень эффективности для небольших и средних систем, однако может стать численно нестабильным из-за возможного деления на малые числа, что может привести к аккумуляции ошибок округления и неточного итоговому результату.

Временная алгоритмическая сложность складывается из нескольких действий: прямой ход метода, обратный ход метода, расчёт невязок, проверка матрицы на совместность. При прямом ходе используется цикл, в который вложен вложенный цикл, внутри каждого цикла мы проходим по всем элемента матрицы размера , что даёт нам алгоритмическую сложность . При обратном ходе для каждого вычисления переменной мы суммируем произведения найденных решений и получаем результат, это реализовано за счёт вложенного цикла, количество операций внутри каждого из вложенных , тогда алгоритмическая сложность составляет . Расчёт невязок также имеет алгоритмическую сложность , так как для каждого из уравнений мы работаем с переменными. Проверка матрицы на совместность также имеет квадратичную алгоритмическую сложность. Таким образом, общая алгоритмическая сложность реализованного программного метода = .

Что касается численных ошибок численного метода, то основным источником ошибок являются ошибки округления.

**Вывод**

В результате выполнения лабораторной работы была реализовано решение СЛАУ методом Гаусса на языке программирования C++. Программа разделяется на 3 основных аспекта: проверка матрицы на совместность, решение методом Гаусса (этот этап разделяется на прямой и обратный ход) и расчёт невязок. Алгоритмическая сложность метода составила .

Метод Гаусса – метод решения СЛАУ, отличающийся высокой эффективностью и точностью для небольших и средних систем, однако возможна численная нестабильность из-за ошибок округления.