Санкт-Петербургский Государственный Университет

Отчет по предмету «Алгоритмы и анализ сложности»

Решение СЛАУ методом Гаусса

Направление 02.03.02 «Фундаментальная информатика и информационные технологии»

Выполнил студент 3 курса

Савченко Илья Анатольевич

Группы 21.Б12-ПУ

2023 г.

Содержание

1 Описание алгоритма

1.1 Краткое описание

Метод Гаусса, также известный как метод исключения Гаусса, представляет собой алгоритм решения систем линейных уравнений и нахождения ранга матрицы. Он был разработан немецким математиком и физиком Карлом Фридрихом Гауссом в начале XIX века. Алгоритм использует элементарные преобразования матрицы (перестановка строк, умножение, деление строки на число и прибавление одной строки к другой) для приведения системы уравнений к упрощенному виду.

1.2 Постановка задачи

Пусть дана система линейных уравнений:

$$\begin{cases} a_{11}x_1 + \cdots + a_{1n}x_n = b_1 \\ \cdots\ \cdots\ \cdots\ \cdots\ \cdots\ \cdots\ \cdots\ \cdots \\ a_{n1}x_1 + \cdots + a_{nn}x_n = b_n \end{cases}$$

где a_{ij} - коэффициенты системы, x_i - неизвестные переменные, и b_i - правые части уравнений.

Требуется решить данную систему линейных уравнений.

1.3 Идея алгоритма

Идея метода заключается в последовательном преобразовании системы n линейных уравнений с n неизвестными в эквивалентную систему с верхнетреугольной матрицей коэффициентов, то есть к матрице вида:

$$\begin{bmatrix} a'_{11} & \cdots & a'_{1n} \\ \vdots & \ddots & \vdots \\ 0 & \cdots & a'_{nn} \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix}$$

Поскольку матрицу верхнетреугольного вида, проще решить методом обратной подстановки.

Шаги метода следующие:

1. Прямой ход (элиминация):

- Выполнить элементарные преобразования строк с целью получить верхнюю треугольную матрицу.

- Для каждого уравнения i, начиная с первого, вычесть из уравнений с номерами $i + 1$ до m подходящую кратную i-ю строку так, чтобы получить нули под главной диагональю.

2. Обратный ход:

- Начиная с последнего уравнения, найти значение соответствующей переменной и подставить его обратно в предыдущие уравнения.

- Повторять этот процесс, двигаясь к первому уравнению, пока не будут найдены все значения переменных.

Полученные значения переменных x_i представляют собой решение исходной системы линейных уравнений.

1.4 Алгоритм

1.4.1 Рекомендованные структуры данных

Матрица (двумерный массив) A[1...n, 1…n], вектор (одномерный массив) b[1…n].

1.4.2 Обычный метод Гаусса

Реализация:https://github.com/st1gmat/gauss_emp_analysis/blob/main/gauss_not mod.py

1. ***Прямой ход (Forward Elimination):***

Проход по каждой строке матрицы A (обозначим текущую строку как ii).

Проверка, что диагональный элемент $A[i][i]$ не равен нулю. Если равен, возвращаемся (решение не существует из-за деления на ноль).

Для каждой строки j, находящейся ниже строки i, вычисление множителя $factor = A[j][i]/A[i][i]$

Вычитание $factor \times$ текущая строка i из строки j, начиная с элемента i и заканчивая последним элементом строки.

2. ***Обратный ход (Back Substitution):***

Создание вектора решений x и инициализация его нулями.

Начиная с последней строки $n - 1$ и двигаясь вверх к первой строке 0, вычисление элемента решения:

- $x[i] = b[i]$ (копирование значения из вектора свободных членов).

- Для каждого элемента $j > i$, вычитание $A[i][j] \times x[j]$.
- Деление на диагональный элемент $A[i][i]$.

3. Возвращение вектора решений x.

1.4.3 Априорный анализ обычного метода Гаусса

Предположим, что на i-ом шаге

$$\left[\begin{array}{cccccc|c} a_{11} & a_{12} & \dots & \dots & \dots & a_{1n} & b_1 \\ 0 & a_{22} & \dots & \dots & \dots & a_{2n} & b_2 \\ 0 & 0 & \ddots & & & & \vdots \\ \vdots & & & a_{ii} & \dots & a_{in} & b_i \\ \vdots & & & a_{i+1,i} & \dots & a_{i+1,n} & b_{i+1} \\ & & & \vdots & & \vdots & \vdots \\ 0 & & \dots & a_{ni} & \dots & a_{nn} & b_n \end{array}\right]$$

Заметим, что необходимо модифицировать элементы $(n-i) \times (n-i)$ подматрицы A, состоящей из строк и столбцов с индексами $i+1, \dots, n$; каждая из этих записей должна быть умножена на некоторый коэффициент, следовательно, $(n-i)^2$ умножений. Для вычисления этих коэффициентов необходимо $(n-i)$ делений, по той же причине необходимо выполнить $(n-i)^2$ сложений. Для $n-i$ элементов вектора b имеем $(n-i)$ умножений и $(n-i)$ делений. В итоге, получаем, что количество операций для умножения/деления:

$$M_{\text{п}} = \sum_{i=1}^{n-1} \left((n-i)^2 + 2(n-i)\right)$$

А для сложения или вычитания:

$$A_{\text{п}} = \sum_{i=1}^{n-1} \left((n-i)^2 + (n-i)\right)$$

Используем формулы

$$\sum_{k=1}^{n} k = \frac{n(n+1)}{2}, \qquad \sum_{k=1}^{n} k^2 = \frac{n(n+1)(2n+1)}{6}$$

И следовательно, выражая получаем:

$$M_{\text{п}} = \frac{2n^3 + 3n^2 - 5n}{6}, \qquad A_{\text{п}} = \frac{n^3 - n}{3}$$

Для обратного шага понадобится одно деление, чтобы найти x_n , далее одно умножение, одно деление чтобы найти x_{n-1} и так далее, следовательно, получим:

$$M_o = n + \sum_{i=1}^{n-1} i = \frac{n(n+1)}{2}$$

А для сложения или вычитания:

$$\text{A}_o = \sum_{i=1}^{n-1} i = \frac{n(n-1)}{2}$$

Общее количество операция умножения/деления и сложения/вычитания:

$$M = M_{\text{п}} + M_{\text{о}} = \frac{n(n^2 + 3n - 1)}{n}$$

$$A = A_{\text{п}} + A_{\text{о}} = \frac{n(2n^2 + 3n - 5)}{6}$$

А общее количество всех операций:

$$T(n) = M + A = \frac{2n^3}{3} + \frac{3n^2}{2} + \frac{7n}{6}$$

Очевидно, что для больших n, следует, что:

$$T(n) \approx \frac{2n^3}{3} \Rightarrow O(n^3)$$

арифметических операций.

Один из основных недостатков обычного метода Гаусса связан с тем, что при его реализации накапливается вычислительная погрешность.

[Самарский, Гулин «Численные методы»] ((Подчеркнем, что основное время расчета затрачивается на осуществление прямого хода. Для больших n число действий умножения и деления в методе Гаусса близко к $\frac{n^3}{3}$. Это означает, что на вычисление одного неизвестного тратится в среднем $\frac{n^2}{3}$ действий. По затратам времени и необходимой машинной памяти метод

Гаусса пригоден для решения систем уравнений общего вида с числом неизвестных n порядка 100. Для того, чтобы уменьшить рост вычислительной погрешности применяются различные модификации метода Гаусса. Например, метод Гаусса с выбором главного элемента по столбцам, в этом случае на каждом этапе прямого хода строки матрицы переставляются таким образом, чтобы диагональный угловой элемент был максимальным. При исключении соответствующего неизвестного из других строк деление будет производиться на наибольший из возможных коэффициентов и следовательно относительная погрешность будет наименьшей))

1.4.4 Модифицированный метод Гаусса

Реализация:https://github.com/st1gmat/gauss_emp_analysis/blob/main/gauss_mod.py

Очевидным препятствием на пути применения этого метода может стать обнуление ведущего элемента на некотором шаге. Однако если исходная матрица системы есть матрица с диагональным преобладанием, то этого не случится. Если же матрица не обладает этим свойством, то применяют метод Гаусса с выбором главного элемента по столбцу. В данной модификации роль ведущего элемента на k-ом шаге играет максимальный по модулю элемент, расположенный в k-ом столбце в строках с номерами, большими k. При этом строка, в которой находится этот элемент, переставляется с k-ой строкой. Очевидно, что если исходная матрица СЛАУ неособая, то данная модификация свободна от отмеченного недостатка простого метода Гаусса.

1.4.5 Априорный анализ модифицированного метода Гаусса

Выбор ведущего элемента:

Для каждого шага прямого хода нужно найти максимальный элемент в текущем столбце. Всего шагов прямого хода: $O(n)$

На каждом шаге требуется сравнить элементы в столбце, что занимает $O(n)$ времени

Таким образом, общая сложность выбора ведущего элемента $O(n^2)$.

В итоге, для больших n аналогично сложность будет $O(n^3)$

1.5 Оценка затрачиваемой памяти

Исходные данные:

Матрица коэффициентов системы уравнений: $n * n$ элементов.

Вектор свободных членов: n элементов.

Общая память для исходных данных: $O(n^2)$.

Промежуточные данные:

В процессе работы алгоритма Гаусса может потребоваться хранить временные переменные, такие как коэффициенты для элементарных преобразований и индексы текущего столбца и строки. Эти переменные занимают постоянное количество памяти для каждой итерации, поэтому оценка памяти на этом этапе также может быть $O(1)$.

Следовательно, общая оценка занимающей память для алгоритма Гаусса: $O(n^2) + O(1) = O(n^2)$.

2 Описание входных данных и генератора

2.1 Входные данные

Матрица $A[1 \dots n, 1 \dots n]$, вектор $b[1 \dots n]$, $n = 1 \dots 320$

2.2 Генератор

Реализация:https://github.com/st1gmat/gauss_emp_analysis/blob/main/generators.py

Используется: generate_input_data

Генерирует случайную матрицу коэффициентов matrix размера (size, size) с элементами, равными случайным числам от 0 до 1; случайный вектор правой части b размера size.

Возвращает сгенерированную матрицу и вектор.

3 Эмпирический анализ

3.1 Вычислительный эксперимент и анализ результата

3.1.1 Обычный метод Гаусса

Размер матрицы n*n	Количество тестов	Среднее время(T(n))
10	20	0.00004980564117431641
20	20	0.00015000104904174804
40	20	0.00134509801864624023
80	20	0.01041834354400634696
160	20	0.07998282909393310269
320	20	0.65809830427169802025

Получаем, что

$$\frac{T(20)}{T(10)} = 3.0117$$

$$\frac{T(40)}{T(20)} = 8.9672$$

$$\frac{T(80)}{T(40)} = 7.74541$$

$$\frac{T(160)}{T(80)} = 7.6771$$

$$\frac{T(320)}{T(160)} = 8.2279$$

$T(n) = \frac{2n^3}{3} + \frac{3n^2}{2} + \frac{7n}{6}$ – проверим отношения временных характеристик

Поскольку данная функция представляет асимптотическую оценку, то отношение временных характеристик при увеличении размера входных данных будет стремиться к константе.

$\frac{T(2n)}{T(n)} = \frac{(2n)^3}{3} \div \frac{2n^3}{3} = 8.$

Все значения отношений близки к ожидаемому значению 8.

3.1.2 Модифицированный метод Гаусса

Размер матрицы n*n	Количество тестов	Среднее время($T(n)$)
10	20	0.00004981756210327148
20	20	0.00025198459625244143
40	20	0.00139602422714233407
80	20	0.01036237478256225517
160	20	0.08018145561218262274
320	20	0.66149204969406127930

$$\frac{T(20)}{T(10)} = 5.0581$$

$$\frac{T(40)}{T(20)} = 5.5401$$

$$\frac{T(80)}{T(40)} = 7.4227$$

$$\frac{T(160)}{T(80)} = 7.7377$$

$$\frac{T(320)}{T(160)} = 8.2499$$

Аналогично, все значения близки к теоретической оценке $\frac{T(2n)}{T(n)} = 8$.

4 Список литературы и источники

1. Левитин А. «Алгоритмы: введение в разработку и анализ»
2. Самарский, Гулин «Численные методы»
3. Иванов А.П., Олемской И.В.,Олемской Ю.В. «Численные методы»
4. MATH 488/688: «Numerical Analysis», North Dakota State University
5. «Numerical Analysis Lecture Notes», Peter J. Oliver

5 Характеристики вычислительной среды и оборудования

Вычислительная среда: интерпретатор python

Оборудование:

Процессор: 11th Gen Intel(R) Core(TM) i5-11400H @ 2.70GHz 2.69 GHz

ОЗУ: 16,0 ГБ