МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

имени М.В.Ломоносова

Факультет вычислительной математики и кибернетики

Компьютерный практикум по курсу

«ВВЕДЕНИЕ В ЧИСЛЕННЫЕ МЕТОДЫ»

ЗАДАНИЕ № 1

ОТЧЁТ

о выполненном задании

студента 205 учебной группы факультета ВМК МГУ

Феофилактова Андрея Дмитриевича

Москва, 2016

Содержание

1. Цель работы

Изучить классический метод Гаусса решения системы линейных алгебраических уравнений.

2. Постановка задачи

Дана система уравнений $Ax = f$ порядка $n \times n$ 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 её правой части как во входном файле данных, так и путём задания специальных формул.

3. Цели и задачи практической работы

1) Научиться решать заданную СЛАУ методом Гаусса и методом Гаусса с выбором главного элемента;
2) Вычислить определитель матрицы $det(A)$;
3) Вычислить обратную матрицу A^{-1};
4) Исследовать вопрос вычислительной устойчивости метода Гаусса (для матриц высоких порядков);
5) Правильность решения СЛАУ подтвердить системой тестов.

4. Описание алгоритма решения

Для решения поставленных задач будет использоваться метод Гаусса – один из классических методов решения системы линейных алгебраических уравнений. Метод подразумевает решение системы $Ax = f$ путём приведения A – матрицы системы к ступенчатому виду (прямой ход) и выражения значений переменных (обратный ход).

Итак, пусть A – матрица порядка $n \times n$, f – вектор-столбец свободных членов системы порядка n.

Прямой ход состоит в послдовательном повторении трёх шагов для всех $i = 1 \dots n$:

- Из строчек A с i по n выберем такую, что $A_{ii} \neq 0$ (в случае классичесского метода Гаусса) и такую, что $A_{ii} = \max\limits_{j=i \dots n} A_{ji}$ (в случае метода Гаусса с выбором главного элемента по столбцу) и поменяем её местами с i-той. В случае, если после этого $A_{ii} = 0$, отметим, что матрица является вырожденной (значит, либо система несовместна, либо многообразие решений невырожденно) и перейдём к $i = i + 1$.

- Поделим теперь все элементы i-той строчки A (а так же i-тую строчку правой части) на A_{ii}.

- Теперь из каждой j-той ($j = i + 1 \dots n$) строчки A и правой части вычтем i-тую, домноженную на A_{ji}. Теперь каждый элемент i-того столбца, стоящий ниже i-того равен нулю.

После выполнения прямого хода матрица приведена к верхне-ступенчатому виду. Причём если r – номер последней ненулевой строчки в матрице A, $r = rank(A)$. Теперь если найдётся $f_j \neq 0$ ($j = r + 1 \dots n$), то система несовместна. В ином случае многообразие решений имеет размерность, равную $n - r$. В частности, если $r = n$, решение единственно. В дальнейшем будем считать, что мы работаем именно с невырожденными матрицами, то есть решение единственно и матрица имеет верхнетреугольный вид.

Обратный ход включает повторение единственного шага для всех $i = r \dots 1$:

- Из каждой j-той строчки ($j < i$) вычитаем i-тую, домноженную на A_{ji}, не забыв проделать то же с правой частью.

После завершения обратного хода невырожденная матрица A станет единичной.

Заметим, что от правой части уравнения требуется только поддерживать т.н. "элементарные преобразования":

1) обмена двух рядов;

2) умножения ряда на число;

3) прибавления к одному ряду другого, домноженного на число.

В случае, если матрица невырожденная, а правая часть является вектором, как было описано выше, в результате работы алгоритма в правой части окажется искомый корень уравнения. Если же в качестве правой части использовать единичную матрицу, то в результате работы алгоритма справа получим матрицу A^{-1}.

Осталось научиться вычислять определитель матрицы. Для этого нужно отметить, что определитель меняется при проведении первых двух из перечисленных преобразований – меняет знак при обмене двух рядов и, при умножении ряда на число a, делится на a.

5. Описание программы

Программа написана на C++11.
Краткое содержание реализации (подробнее – см. код):

- Чисто виртуальный класс BaseElementGenerator, предоставляющий интерфейс для задания системы уравнений;
- Класс ElementGenerator, реализующий заданные в условии формулы для получения очередного элемента матрицы и вектора;
- Класс EquationsSystem, описывающий систему уравнений. В качестве параметра конструктора принимает имя файла, содержащего систему или объект типа BaseElementGenerator. Имеет методы Solve(), Determinant() и Inverse(), осуществляющие решение системы, нахождение определителя матрицы коэффициентов и обращение матрицы коэффициентов. Поддерживает неявный параметр, определяющий, использовать или нет модификацию метода Гаусса.
- Чисто виртуальный класс BaseRightEquationsSystemPart, предоставляющий интерфес для задания правой части уравнения $Ax = f$ (В частности с помощью него реализован подсчёт числа операций);
- Классы Matrix и Vector, представляющие из себя рализации квадратной матрицы и вектора соответственно. Оба класса отнаследованы от BaseRightEquationsSystemPart, а также представляют стандартный набор операций для этих алгебраических объектов (перегружены операторы $*, *=, +, +=, -, -=$)
- Также реализованы вспомогательные функции Equal(double, double) и Zero(double), проверяющие числа на равенства и на равенство нулю, соответственно и класс OperationsCounter, который позволяет посчитать количество операций при выполнении наивно реализованного алгоритма.

6. Тесты

Проверка проводилась с помощью приложенного скрпита на python3.4, с использованием библиотеки numpy

6.1. Небольшие тесты

6.1.1. Тест 1

$$\begin{cases} 2x_1 + 5x_2 + 4x_3 + x_4 = 20, \\ x_1 + 3x_2 + 2x_3 + x_4 = 11, \\ 2x_1 + 10x_2 + 9x_3 + 7x_4 = 40, \\ 3x_1 + 8x_2 + 9x_3 + 2x_4 = 37. \end{cases}$$

Результат: Рассчитано программой:

$$\begin{pmatrix} 1 \\ 2 \\ 2 \\ -0 \end{pmatrix}$$

Рассчитано при помощи пакета numpy:

$$\begin{pmatrix} 1.0 \\ 2.0 \\ 2.0 \\ -3.7 \times 10^{-15} \end{pmatrix}$$

6.1.2. Тест 2

$$\begin{cases} 6x_1 + 4x_2 + 5x_3 + 2x_4 = 1, \\ 3x_1 + 2x_2 + 4x_3 + x_4 = 3, \\ 3x_1 + 2x_2 - 2x_3 + x_4 = -7, \\ 9x_1 + 6x_2 + x_3 + 3x_4 = 2. \end{cases}$$

Результаты: Матрица является вырожденной.

6.1.3. Тест 3

$$\begin{cases} 2x_1 + x_2 + x_3 = 2, \\ x_1 + 3x_2 + x_3 + x_4 = 5, \\ x_1 + x_2 + 5x_3 = -7, \\ 2x_1 + 3x_2 - 3x_3 - 10x_4 = 14. \end{cases}$$

Результат: Рассчитано программой:

$$\begin{pmatrix} 1 \\ 2 \\ -2 \\ -0 \end{pmatrix}$$

Рассчитано при помощи пакета numpy:

$$\begin{pmatrix} 1.0 \\ 2.0 \\ -2.0 \\ -0.0 \end{pmatrix}$$

6.2. Объёмные тесты

Тестирование проводилось для матриц порядка $n = 100$, заданных формулой

$$A_{ij} = \begin{cases} q_M^{i+j} + 0.1 \cdot (j - i), i \neq j, \\ (q_M - 1)^{i+j}, i = j. \end{cases}$$

где $q_M = 1.001 - 2 * M * 10^{-3}$, $i, j = 1, \dots n$. M было принято равным 6(по условию). Элементы вектора свободных коэффициентов задавались формулой:

$$x \cdot \exp \frac{x}{i} \cdot \cos \frac{x}{i}$$

Программа выполнялась для значений X из промежутка с 0 до 15 с шагом 0.5. Для проверки решений использовался приложенный скрипт check.py, в котором считалась евклидова норма вектора невязки. При увеличении x это значение растёт как для обычного метода Гаусса, так и для его усовершенствованного варианта. Причём выбор главного элемента даёт выигрыш до нескольких порядков (см. таблицу).

x	Без модификации	С модификацией	Разность
0.0	0	0	0
0.5	6.8153e-11	4.70669e-14	-6.81059e-11
1.0	2.48026e-10	1.48061e-13	-2.47878e-10
1.5	3.1454e-10	1.16844e-13	-3.14423e-10
2.0	2.92344e-10	2.49085e-13	-2.92095e-10
2.5	1.79998e-09	1.05803e-12	-1.79892e-09
3.0	5.69084e-09	3.3937e-12	-5.68745e-09
3.5	1.05627e-08	6.3643e-12	-1.05563e-08
4.0	1.60849e-08	4.44234e-12	-1.60804e-08
4.5	1.0672e-08	2.01469e-12	-1.067e-08
5.0	1.74529e-08	1.17426e-11	-1.74412e-08
5.5	8.85728e-08	4.26294e-11	-8.85301e-08
6.0	2.33569e-07	3.68016e-11	-2.33532e-07
6.5	4.15609e-07	1.26384e-10	-4.15483e-07
7.0	5.02028e-07	1.90374e-10	-5.01838e-07
7.5	4.25878e-07	1.48411e-10	-4.25729e-07
8.0	3.85135e-07	1.35395e-10	-3.84999e-07
8.5	2.64374e-06	1.35466e-09	-2.64239e-06
9.0	6.69058e-06	2.61472e-09	-6.68797e-06
9.5	1.28617e-05	3.03868e-09	-1.28587e-05
10.0	1.861e-05	2.94096e-09	-1.86071e-05
10.5	1.85298e-05	1.42146e-08	-1.85156e-05
11.0	4.33969e-07	2.79863e-10	-4.3369e-07
11.5	5.94858e-05	5.69602e-08	-5.94288e-05
12.0	0.000164127	7.69133e-08	-0.00016405
12.5	0.000334385	1.47586e-07	-0.000334238
13.0	0.000495664	3.49012e-07	-0.000495315
13.5	0.000619153	2.65897e-07	-0.000618887
14.0	0.000248584	8.04283e-08	-0.000248504
14.5	0.000958917	6.01043e-07	-0.000958316
15.0	0.00407568	1.69442e-06	-0.00407399

7. Выводы

Приведённые результаты тестов показывают, что уже при $n = 100$ нетрудно найти систему уравнений с n переменными, для которой простой метод Гаусса будет давать заметную (относительно исходных данных) ошибку. Таким образом, метод вычислительно устойчивым не является.

Приведённая модификация метода Гаусса – метод Гаусса с выбором главного элемента показывает лучшую на несколько порядков точность, но видно, что если увеличить n, и этой точности станет недостаточно.

Для решения этой проблемы можно выбирать максимальный элемент не из столбца, а из всей необработанной подматрицы. Это позволит увеличить точность, но добавит асимптотической сложности и так не оптимальному алгоритму, поэтому подобное решение будет зависеть уже от конкретных целей в соблюдении баланса "время выполнения - точность".