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
Долгушева Глеба Дмитриевича

гор. Москва
2023

Содержание

Подвариант 1

Постановка задачи

Пусть дана система уравнений $Ax = f$ порядка nxn с некоторой матрицей A. Написать программу, решающую (в случае, если матрица невырожденная) систему линейных алгебраических уравнений заданного пользователем размера (n – параметр программы) методом Гаусса и методом Гаусса с выбором главного элемента. Кроме того, в этой программе реализовать функции для вычисления вспомогательных объектов: определителя матрицы, матрицы, обратной данной, и числа обусловленности матрицы.

Цели и задачи практической работы

- Изучить метод Гаусса и метод Гаусса с выбором главного элемента, изучить алгоритмы вычисления определителя, обратной матрицы и числа обусловленности
- С помощью дополнительных библиотек или самостоятельно реализовать вычисление вспомогательных функций, таких как норма вектора, матрицы и т.п.
- Реализовать данные алгоритмы на любом языке программирования (Для этой задачи был выбран язык программирования Python 3.10)
- Провести тестирование написанной программы на предоставленных в задании системах, подобрать собственные системы для тестирования, если это необходимо
- Исследовать вопрос вычислительной устойчивости реализованных алгоритмов решения системы линейных алгебраических уравнений с невырожденной матрицей коэффициентов в зависимости от значения числа обусловленности матрицы

Описание алгоритма решения

Пусть задана система линейных алгебраических уравнений:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + ... + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + ... + a_{2n}x_n = b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + ... + a_{3n}x_n = b_3 \\ \dots \\ a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + ... + a_{nn}x_n = b_n \end{cases}$$

Запишем ее для удобства в матричной форме:

$$Ax = b \tag{1}$$

Пусть, к тому же, матрица A является невырожденной. Опишем метод Гаусса для решения данной системы линейных алгебраических уравнений.

Метод Гаусса

В основе метода Гаусса, как, впрочем, и многих других методов решения систем линейных алгебраических уравнений (1) лежит следующее утверждение. Пусть матрица B невырождена. Тогда система уравнений

$$BAx = Bb \tag{2}$$

эквивалентна системе (1), т. е. решение системы (1) — решение системы (2) и, наоборот, решение системы (2) — решение системы (1). Матрица B выбирается так, чтобы матрица BA была проще матрицы A и решение системы (2) находилось легче, чем решение системы (1). В методе Гаусса матрица B конструируется при помощи элементарных нижних треугольных матриц так, чтобы матрица BA была верхней треугольной. Тогда решение системы (2) становится тривиальной задачей. Приведем необходимые расчетные формулы. Для удобства изложения положим

$$A^{(1)} = A, b^{(1)} = b \tag{3}$$

и запишем исходную систему в индексной форме:

$$\begin{cases} a_{11}^{(1)}x_1 + a_{12}^{(1)}x_2 + a_{13}^{(1)}x_3 + \ldots + a_{1n}^{(1)}x_n = b_1^{(1)} \\ a_{21}^{(1)}x_1 + a_{22}^{(1)}x_2 + a_{23}^{(1)}x_3 + \ldots + a_{2n}^{(1)}x_n = b_2^{(1)} \\ a_{31}^{(1)}x_1 + a_{32}^{(1)}x_2 + a_{33}^{(1)}x_3 + \ldots + a_{3n}^{(1)}x_n = b_3^{(1)} \\ \dots \\ a_{n1}^{(1)}x_1 + a_{n2}^{(1)}x_2 + a_{n3}^{(1)}x_3 + \ldots + a_{nn}^{(1)}x_n = b_n^{(1)} \end{cases} \tag{4}$$

Предположим, что $a_{11}^{(1)} \neq 0$ и введем множители $l_{i1} = a_{i1}^{(1)}/a_{11}^{(1)}; i = 2, \ldots, n$ Для каждого $i = 2, \ldots, n$, умножим обе части первого уравнения в (4) на l_{i1} и вычтем полученное равенство из i-го уравнения. Придем к новой (эквивалентной) системе

$$A^{(2)}x = b^{(2)}$$

вида

$$\begin{cases} a_{11}^{(1)}x_1 + a_{12}^{(1)}x_2 + a_{13}^{(1)}x_3 + \ldots + a_{1n}^{(1)}x_n = b_1^{(1)} \\ a_{22}^{(2)}x_2 + a_{23}^{(2)}x_3 + \ldots + a_{2n}^{(2)}x_n = b_2^{(2)} \\ \ldots \\ a_{n2}^{(2)}x_2 + a_{n3}^{(2)}x_3 + \ldots + a_{nn}^{(2)}x_n = b_n^{(2)} \end{cases} \tag{5}$$

Согласно описанию, данному выше, новые элементы матрицы и правой части вычисляются по формулам

$$\begin{aligned} a_{ij}^{(2)} &= a_{ij}^{(1)} - l_{i1}a_{1j}^{(1)}, i, j = 2, \ldots, n \\ b_i^{(2)} &= b_i^{(1)} - l_{i1}b_1^{(1)}, i = 2, \ldots, n \end{aligned} \tag{6}$$

На втором шаге проделаем аналогичные вычисления с подсистемой 6, включающей уравнения с номерами $2, \ldots, n$, и приведем матрицу системы к верхней треугольной форме во втором столбце. Это можно сделать, если $a_{22}^{(2)} \neq 0$. Повторяя аналогичные действия, на n-1 шаге получим систему $A^{(n)} = b^{(n)}$ с верхнетреугольной матрицей. Приведем формулы для расчета коэффициентов матрицы $A_{(k)}$ на k-ом шаге, где $k = 1, \ldots, n-1$

$$\begin{aligned} l_{ik+1} &= a_{ik}^{(k)}/a_{kk}^{(k)}; i = k+1, \ldots, n \\ a_{ij}^{(k+1)} &= a_{ij}^{(k)} - l_{i1}a_{kj}^{(k)}, i, j = k+1, \ldots, n \\ b_i^{(k+1)} &= b_i^{(k)} - l_{ik}b_1^{(k)}, i = k+1, \ldots, n \end{aligned} \tag{7}$$

После описанных выше преобразований получим систему

$$\begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & \ldots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} & \ldots & a_{2n}^{(2)}x_n \\ 0 & 0 & a_{33}^{(3)} & \ldots & a_{3n}^{(3)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & a_{nn}^{(n)} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \ldots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1^{(1)} \\ b_2^{(2)} \\ b_3^{(3)} \\ \ldots \\ b_n{}^{(n)} \end{pmatrix}$$

из которой неизвестные найдем по формулам

$$\begin{cases} x_1 = b_1^{(1)} - \sum_{i=2}^{n} a_{1i}^{(1)}x_i \\ \ldots \\ x_{n-2} = b_{n-2}^{(n-2)} - a_{n-2n-1}^{(n-2)}x_{n-1} - a_{n-2n}^{(n-2)}x_n \\ x_{n-1} = b_{n-1}^{(n-1)} - a_{n-1n}^{(n-1)}x_n \\ x_n = b_n^{(n)} \end{cases}$$

Метод Гаусса с выбором главного элемента

Описанный выше метод может быть применен лишь в том случае, когда все ведущие элементы метода Гаусса отличны от нуля. Для этого невырожденности матрицы недостаточно. Поэтому предложим модификацию этого, изученного

выше, метода, которая позволяет применять его для решения систем уравнений с любой невырожденной матрицей. Теперь на этапе приведения к верхнетреугольной форме на итерации под номером k следует заменять строку под номером k на такую строку, у которой элемент a_{kj} максимален, $j = k..n$. Кроме того, это улучшение помогает улучшить точность вычислений, что обусловлено особенностями реализации чисел с плавающей точкой в ЭВМ.

Нахождение Определителя

Отметим, что при приведении матрицы к треугольному виду мы легко получаем его определитель. Определитель треугольной матрицы с единицами на главной диагонали равен единице, а при делении строки на какое-то число, определитель делится на это число (при методе Гаусса с выбором опорного элемента также требуется запоминать количество перестановок строк и в случае нечетного их количества умножить полученное число на -1).

Нахождение обратной матрицы

Логическим развитием метода Гаусса является метод Жордана-Гаусса, позволяющий находить обратную матрицу. Опишем его идею. Все элементарные преобразования, примененные к матрице A применяются и к единичной матрице. Матрица A приводится к верхнему треугольному виду с единицами на главной диагонали, а потом уже треугольная матрица приводится к единичной. Тогда на месте матрицы I окажется A^{-1}.

Число обусловленности

Числом обусловленности матрицы A называется:

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

В данной работе мы используем бесконечную матричную норму. Число обусловленности является мерой чувствительности матрицы коэффициентов к погрешностям вектора правой части.

Тестирование программы

Точные решения были найдены с помощью вычислительных алгоритмов сайта `https://www.wolframalpha.com/`

Тест 1

Система:

$$\begin{cases} 2x_1 + -1x_2 + -6x_3 + 3x_4 = -1 \\ 1x_1 + 3x_2 + -6x_3 + 2x_4 = 3 \\ 3x_1 + -2x_2 + 2x_3 + -2x_4 = 8 \\ 2x_1 + -1x_2 + 2x_3 + = 4 \end{cases}$$

Точное решение:

$$x_1 = \frac{175}{69}, x_2 = \frac{5}{46}, x_3 = \frac{89}{69}, x_4 = -\frac{95}{69}$$

Метод Гаусса:

$$x_1 = 2.5362318841, x_2 = 1.2898550725, x_3 = 0.1086956522, x_4 = -1.3768115942$$

Метод Гаусса с выбором ведущего элемента:

$$x_1 = 2.5362318841, x_2 = 1.2898550725, x_3 = 0.1086956522, x_4 = -1.3768115942$$

Определитель матрицы: 138.0 (точное значение 138)
Число обусловленности: 8.0868424825
Обратная матрица:

$$\begin{pmatrix} -0.029 & 0.1594 & 0.1159 & 0.2754 \\ -0.2319 & 0.2754 & -0.0725 & 0.2029 \\ -0.087 & -0.0217 & -0.1522 & 0.3261 \\ 0.1014 & -0.058 & -0.4058 & 0.5362 \end{pmatrix}$$

Тест 2

Система:

$$\begin{cases} 1x_1 + 1x_2 + -3x_3 + 1x_4 = -1 \\ 2x_1 + 1x_2 + -2x_3 + -1x_4 = 1 \\ 1x_1 + 1x_2 + 1x_3 + -1x_4 = 3 \\ 2x_1 + 4x_2 + -6x_3 + 14x_4 = 12 \end{cases}$$

Точное решение:

$$x_1 = \frac{14}{5}, x_2 = -\frac{1}{5}, x_3 = \frac{8}{5}, x_4 = \frac{6}{5}$$

Метод Гаусса:

$$x_1 = 2.8, x_2 = -0.2, x_3 = 1.6, x_4 = 1.2$$

Метод Гаусса с выбором ведущего элемента:

$$x_1 = 2.8, x_2 = -0.2, x_3 = 1.6, x_4 = 1.2$$

Определитель матрицы: -40.0 (точное значение -40)
Число обусловленности: 47.7878363499
Обратная матрица:

$$\begin{pmatrix} -1.35 & 1.3 & -0.55 & 0.15 \\ 1.4 & -1.2 & 1.2 & -0.1 \\ -0.45 & 0.1 & 0.15 & 0.05 \\ -0.4 & 0.2 & -0.2 & 0.1 \end{pmatrix}$$

Тест 3

Система:

$$\begin{cases} 1x_1 + 4x_2 + 5x_3 + 2x_4 = 2 \\ 2x_1 + 9x_2 + 8x_3 + 3x_4 = 7 \\ 3x_1 + 7x_2 + 7x_3 + = 12 \\ 5x_1 + 7x_2 + 9x_3 + 2x_4 = 20 \end{cases}$$

Точное решение:

$$x_1 = \frac{31}{6}, x_2 = \frac{2}{3}, x_3 = -\frac{7}{6}, x_4 = 0$$

Метод Гаусса:

$$x_1 = 5.1666666667, x_2 = 0.6666666667, x_3 = -1.1666666667, x_4 = 0.0$$

Метод Гаусса с выбором ведущего элемента:

$$x_1 = 5.1666666667, x_2 = 0.6666666667, x_3 = -1.1666666667, x_4 = 0.0$$

Определитель матрицы: -84.00000000000003 (точное значение -84)
Число обусловленности: 30.8784184228
Обратная матрица:

$$\begin{pmatrix} -0.6667 & 0.1667 & -0.25 & 0.4167 \\ -0.5238 & 0.381 & -0.0 & -0.0476 \\ 0.8095 & -0.4524 & 0.25 & -0.131 \\ -0.1429 & 0.2857 & -0.5 & 0.2143 \end{pmatrix}$$

Тест 4

В тесте 4 матрица вводилась по формуле:

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

Правая часть по формуле:

$$b_i = m * i + n$$

Параметры: $n = 30, m = 20$.

Решение, полученное методом Гаусса:

$x_1 = 0.0942208563$, $x_2 = 0.1396646405$, $x_3 = 0.1850992682$, $x_4 = 0.2305247422$, $x_5 = 0.2759410654$, $x_6 = 0.3213482403$, $x_7 = 0.3667462699$, $x_8 = 0.4121351569$, $x_9 = 0.457514904$, $x_{10} = 0.5028855141$, $x_{11} = 0.5482469897$, $x_{12} = 0.5935993339$, $x_{13} = 0.6389425491$, $x_{14} = 0.6842766383$, $x_{15} = 0.7296016043$, $x_{16} = 0.7749174496$, $x_{17} = 0.8202241771$, $x_{18} = 0.8655217896$, $x_{19} = 0.9108102898$, $x_{20} = 0.9560896804$, $x_{21} = 1.0013599642$, $x_{22} = 1.046621144$, $x_{23} = 1.0918732225$, $x_{24} = 1.1371162023$, $x_{25} = 1.1823500864$, $x_{26} = 1.2275748774$, $x_{27} = 1.272790578$, $x_{28} = 1.317997191$, $x_{29} = 1.3631947192$, $x_{30} = 1.4083831653$

Решение, полученное методом Гаусса с выбором ведущего элемента:

$x_1 = 0.0942208563$, $x_2 = 0.1396646405$, $x_3 = 0.1850992682$, $x_4 = 0.2305247422$, $x_5 = 0.2759410654$, $x_6 = 0.3213482403$, $x_7 = 0.3667462699$, $x_8 = 0.4121351569$, $x_9 = 0.457514904$, $x_{10} = 0.5028855141$, $x_{11} = 0.5482469897$, $x_{12} = 0.5935993339$, $x_{13} = 0.6389425491$, $x_{14} = 0.6842766383$, $x_{15} = 0.7296016043$, $x_{16} = 0.7749174496$, $x_{17} = 0.8202241771$, $x_{18} = 0.8655217896$, $x_{19} = 0.9108102898$, $x_{20} = 0.9560896804$, $x_{21} = 1.0013599642$, $x_{22} = 1.046621144$, $x_{23} = 1.0918732225$, $x_{24} = 1.1371162023$, $x_{25} = 1.1823500864$, $x_{26} = 1.2275748774$, $x_{27} = 1.272790578$, $x_{28} = 1.317997191$, $x_{29} = 1.3631947192$, $x_{30} = 1.4083831653$

Полученные решения довольно близки к точному решению данной системы.

Определитель матрицы: $1.1032591757261959e79$
Число обусловленности: 1.0506854835

Обратную матрицу приводить в отчете не имеет смысла, так как читаемость такой информации неприемлема.

Выводы

Метод Гаусса с выбором ведущего элемента демонстрирует более высокую точность по сравнению с классическим методом Гаусса, поскольку в нем наблюдается меньшая накопленная ошибка. Однако, стоит отметить, что этот метод более высокую сложность, что увеличивает время выполнения алгоритма. Таким образом, учитывая относительно невысокий порядок матрицы, использование метода Гаусса является оптимальным решением. Однако, при увеличении размерности задачи, возникает необходимость рассмотрения альтернативных методов, о чем будет рассказано в следующем подварианте.

Подвариант 2

Постановка задачи

Рассмотрим систему линейных алгебраических уравнений с невырожденной квадратной матрицей коэффициентов, представленную в форме матрицы как $Ax = b$. Метод Гаусса, изученный ранее, требует порядка куба от порядка матрицы арифметических операций, что становится длительным при больших размерах матрицы. Это привело к разработке итерационных методов для решения системы линейных алгебраических уравнений, среди которых метод верхней релаксации. Его реализация будет являться целью данной работы.

Задачи практической работы

- Изучить метод верхней релаксации для решения системы линейных алгебраических уравнений для невырожденной положительно определенной матрицы
- Реализовать метод верхней релаксации для решения системы линейных алгебраических уравнений для невырожденной положительно определенной матрицы на одном из языков программирования (Для этой задачи был выбран язык программирования Python 3.10)
- Разработать критерий остановки итерационного процесса, гарантирующий получение приближенного решения исходной системы линейных алгебраических уравнений с заданной точностью
- Протестировать полученную реализацию на различных примерах
- Изучить скорость сходимости метода в зависимости от различных значений параметра ω

Описание алгоритма решения

Пусть задана система линейных алгебраических уравнений:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + ... + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + ... + a_{2n}x_n = b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + ... + a_{3n}x_n = b_3 \\ ... \\ a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + ... + a_{nn}x_n = b_n \end{cases}$$

Или в матричном виде:

$$Ax = b$$

Пусть, к тому же, матрица A является положительно определенной. Представим нашу матрицу в виде суммы:

$$A = L + D + R$$

Где матрица L - нижняя треугольная матрица с нулями на диагонали и совпадающая с A в остальных позициях. R - нижняя треугольная матрица с нулями на диагонали и совпадающая с A в остальных позициях. D - диагональная матрица, элементы которой совпадают с диагональю матрицы A. Тогда итерационный процесс записывается следующей формулой:

$$(D + \omega L)(x^{k+1} - x^k)/\omega + Ax^k = b$$

Где k - номер текущей итерации. ω - итерационный параметр. Метод релаксации сходится, если $\omega \in (0, 2)$. При этом значение параметра ω следует подбирать отдельно для каждой конкретной задачи. Этот процесс представляет отдельный интерес и, более того, для некоторых матриц (симметричных и положительно определенных) имеет однозначное решение. Теперь приведем более удобную для программной реализации формулу для итерационного процесса

$$x_i^{k+1} = (1 - \omega)x_i^k + \omega(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{k+1} - \sum_{j=i+1}^{i-1} a_{ij}x_j^k)/a_{ii}$$

В качестве критерия останова выберем следующее условие. Пусть требуется решить задачу с точностью ϵ, тогда итерационный процесс остановится, если

$$|x^{k+1} - x^k| < \epsilon$$

Тесты

Точные решения были найдены с помощью вычислительных алгоритмов сайта `https://www.wolframalpha.com/`

В связи с отсутствием положительно определенной матрицы коэффициентов среди приведенных вариантов, для тестирования были выбраны дополнительные системы.

Тест 1

Система:

$$\begin{cases} 5x_1 + 2x_2 + -1x_3 = 12 \\ -4x_1 + 7x_2 + 3x_3 = 24 \\ 2x_1 + -2x_2 + 4x_3 = 9 \end{cases}$$

Точное решение:

$$x_1 = \frac{381}{220}, x_2 = \frac{63}{20}, x_3 = \frac{651}{220}$$

Метод релаксации при $\omega = 0.2$, количество итераций $= 77$:

$$x_1 = 1.731818222, x_2 = 3.1499999189, x_3 = 2.9590908786$$

Метод релаксации при $\omega = 0.5$, количество итераций $= 26$:

$$x_1 = 1.7318182639, x_2 = 3.14999999, x_3 = 2.9590908443$$

Метод релаксации при $\omega = 1$, количество итераций $= 15$:

$$x_1 = 1.7318182249, x_2 = 3.1499999998, x_3 = 2.9590908875$$

Метод релаксации при $\omega = 1.7$, количество итераций $=$ None:
Сходимость отсутствует.

Тест 2

Система:

$$\begin{cases} 100x_1 + 13x_2 + 1x_3 + -7x_4 = 1 \\ 10x_1 + 105x_2 + 15x_3 + 1x_4 = 45 \\ 1x_1 + 3x_2 + -97x_3 + -13x_4 = 23 \\ 16x_1 + 1x_2 + 4x_3 + 88x_4 = 2 \end{cases}$$

Точное решение:

$$x_1 = -\frac{2043724}{44757755}, x_2 = \frac{20819206}{44757755}, x_3 = -\frac{10206438}{44757755}, x_4 = \frac{323231}{8951551}$$

Метод релаксации при $\omega = 0.5$, количество итераций $= 31$:

$$x_1 = -0.0456618925, x_2 = 0.4651530378, x_3 = -0.2280373045, x_4 = 0.0361089333$$

Метод релаксации при $\omega = 1$, количество итераций $= 8$:

$$x_1 = -0.0456618904, x_2 = 0.4651530475, x_3 = -0.2280373094, x_4 = 0.0361089368$$

Метод релаксации при $\omega = 1.7$, количество итераций $= 158$:

$$x_1 = -0.0456619028, x_2 = 0.4651530468, x_3 = -0.2280373049, x_4 = 0.0361089451$$

Тест 3

В тесте 3 матрица вводилась по формуле:

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

Правая часть по формуле:

$$b_i = m * i + n$$

Параметры: $n = 30, m = 20$.

Метод релаксации при $\omega = 0.5$, количество итераций $= 31$:

$x_1 = 0.0942208567$, $x_2 = 0.1396646409$, $x_3 = 0.1850992686$, $x_4 = 0.2305247426$, $x_5 = 0.2759410657$, $x_6 = 0.3213482406$, $x_7 = 0.3667462702$, $x_8 = 0.4121351572$, $x_9 = 0.4575149043$, $x_{10} = 0.5028855143$, $x_{11} = 0.5482469899$, $x_{12} = 0.593599334$, $x_{13} = 0.6389425493$, $x_{14} = 0.6842766385$, $x_{15} = 0.7296016044$, $x_{16} = 0.7749174497$, $x_{17} = 0.8202241772$, $x_{18} = 0.8655217897$, $x_{19} = 0.9108102898$, $x_{20} = 0.9560896804$, $x_{21} = 1.0013599642$, $x_{22} = 1.046621144$, $x_{23} = 1.0918732224$, $x_{24} = 1.1371162023$, $x_{25} = 1.1823500863$, $x_{26} = 1.2275748772$, $x_{27} = 1.2727905778$, $x_{28} = 1.3179971908$, $x_{29} = 1.363194719$, $x_{30} = 1.408383165$

Метод релаксации при $\omega = 1.0$, количество итераций $= 5$:

$x_1 = 0.0942208564$, $x_2 = 0.1396646405$, $x_3 = 0.1850992682$, $x_4 = 0.2305247423$, $x_5 = 0.2759410654$, $x_6 = 0.3213482404$, $x_7 = 0.36674627$, $x_8 = 0.412135157$, $x_9 = 0.4575149041$, $x_{10} = 0.5028855141$, $x_{11} = 0.5482469898$, $x_{12} = 0.5935993339$, $x_{13} = 0.6389425491$, $x_{14} = 0.6842766384$, $x_{15} = 0.7296016043$, $x_{16} = 0.7749174496$, $x_{17} = 0.8202241771$, $x_{18} = 0.8655217896$, $x_{19} = 0.9108102898$, $x_{20} = 0.9560896804$, $x_{21} = 1.0013599642$, $x_{22} = 1.046621144$, $x_{23} = 1.0918732224$, $x_{24} = 1.1371162023$, $x_{25} = 1.1823500864$, $x_{26} = 1.2275748774$, $x_{27} = 1.272790578$, $x_{28} = 1.317997191$, $x_{29} = 1.3631947192$, $x_{30} = 1.4083831653$

Метод релаксации при $\omega = 1.7$, количество итераций $= 59$:

$x_1 = 0.0942208562$, $x_2 = 0.1396646404$, $x_3 = 0.1850992682$, $x_4 = 0.2305247424$, $x_5 = 0.2759410656$, $x_6 = 0.3213482406$, $x_7 = 0.3667462703$, $x_8 = 0.4121351573$, $x_9 = 0.4575149045$, $x_{10} = 0.5028855146$, $x_{11} = 0.5482469903$, $x_{12} = 0.5935993344$, $x_{13} = 0.6389425496$, $x_{14} = 0.6842766388$, $x_{15} = 0.7296016047$, $x_{16} = 0.7749174499$,

$x_{17} = 0.8202241774$, $x_{18} = 0.8655217898$, $x_{19} = 0.9108102898$, $x_{20} = 0.9560896803$, $x_{21} = 1.0013599641$, $x_{22} = 1.0466211437$, $x_{23} = 1.0918732221$, $x_{24} = 1.1371162019$, $x_{25} = 1.1823500859$, $x_{26} = 1.2275748768$, $x_{27} = 1.2727905774$, $x_{28} = 1.3179971904$, $x_{29} = 1.3631947186$, $x_{30} = 1.4083831647$

Выводы

В ходе проделанной работы, было обнаружено, что метод верхней релаксации является более эффективным в сравнении с методом Гаусса при работе с матрицами большого порядка. Однако, следует отметить, что точность метода верхней релаксации ниже, чем точность метода Гаусса с выбором ведущего элемента. Кроме того, для применения метода верхней релаксации требуется дополнительная проверка на положительную определенность матрицы, так как данный метод применим только к таким матрицам. Однако, данная проверка является сложной задачей, поэтому на практике проще запустить метод и дождаться его результатов, ф уже по ним сделать выводы о сходимости. Такой подход требует дополнительных затрат времени при отсутствии гарантий результата. Несмотря на вышеупомянутые недостатки, метод верхней релаксации все же может быть применен в определенных задачах. При его использовании необходимо тщательно подходить к выбору параметра ω, так как скорость сходимости метода сильно зависит от этого параметра, что было подтверждено экспериментами.

Код программ

Использованный для всех реализаций язык - Python 3.10

Код для первого подварианта

Реализации всех вычислительных алгоритмов, которые использовались в первом подварианте для решения СЛАУ, вычисления определителя матрицы, нахождения обратной матрицы и вычисления числа обусловленности:

```
import numpy as np
import equations
from math import *
import matplotlib.pyplot as plt

def gauss(input_matrix :np.array, input_b: np.array) -> np.array:
    # Классический метод Гаусса
    matrix = input_matrix.copy()
    b = input_b.copy()
    size = matrix.shape[0]
    for i in range(size):
        if np.abs(matrix[i][i]) < 1.e-6:
            idx_arr = np.where(np.abs(matrix.T[i][i:]) < 1.e-6)
            if not len(idx_arr):
                print("ERROR")
                return
            idx = idx_arr[0][0]
            matrix[idx], matrix[i] = matrix[i], matrix[idx]

        for j in range(i + 1, size):
            b[j] = b[j] - b[i] / matrix[i][i] * matrix[j][i]
            matrix[j] = matrix[j] - matrix[i] / matrix[i][i] * matrix[j][i]

    return solve_right_triangular(matrix, b)

def gauss_modified(input_matrix :np.array, input_b: np.array) -> np.array:
    # Метод Гаусса с выбором главного элемента
    matrix = input_matrix.copy()
    b = input_b.copy()
    size = matrix.shape[0]
    for i in range(size):
        # Выбор строки
        idx = np.abs(matrix[i:, i]).argmax() + i
        matrix[idx], matrix[i] = matrix[i].copy(), matrix[idx].copy()
        b[idx], b[i] = b[i], b[idx]
        # Обниуление оставшихся строк
        for j in range(i + 1, size):
```

```
            b[j] = b[j] - b[i] / matrix[i][i] * matrix[j][i]
            matrix[j] = matrix[j] - matrix[i] / matrix[i][i] * matrix[j][i]

    return solve_right_triangular(matrix, b)

def solve_right_triangular(matrix: np.array, b: np.array) -> np.array:
    # Решение СЛАУ для верхнетреугольной матрицы
    length = matrix.shape[0]
    x = np.zeros_like(b, dtype=float)
    for i in range(length - 1, -1, -1):
        x[i] = (b[i] - np.sum(np.dot(matrix[i][i + 1:], x[i + 1:]))) \
        / matrix[i][i]
    return x

def cond(matrix: np.array):
    # Подсчет числа обусловленности
    return np.linalg.cond(matrix)


def decompose_to_lu(matrix: np.array) -> np.array:
    # Разложение матрицу коэффициентов на матрицы L и U.
    # Создаём пустую LU-матрицу
    lu_matrix = np.matrix(np.zeros_like(matrix))
    size = matrix.shape[0]

    for k in range(size):
        # Вычисляем все остаточные элементы k-ой строки
        for j in range(k, size):
            lu_matrix[k, j] = matrix[k, j] - lu_matrix[k, :k] *\
            lu_matrix[:k, j]
        # Вычисляем все остаточные элементы k-го столбца
        for i in range(k + 1, size):
            lu_matrix[i, k] = (matrix[i, k] - lu_matrix[i, : k] *\
            lu_matrix[: k, k]) / lu_matrix[k, k]

    return lu_matrix

def get_l(matrix: np.array):
    # Получение треугольной матрицы L из представления LU-матрицы
    l = matrix.copy()
    for i in range(l.shape[0]):
            l[i, i] = 1
            l[i, i + 1:] = 0
    return l
```

```
def get_u(matrix: np.array):
    # Получение треугольной матрицы U из представления LU-матрицы
    u = matrix.copy()
    for i in range(1, u.shape[0]):
        u[i, :i] = 0
    return u

def determinant(matrix: np.array):
    # Вычисление определителя
    u = get_u(decompose_to_lu(matrix))
    size = u.shape[0]
    det = 1
    for i in range(size):
        det *= u[i, i]
    return det

def inv(matrix: np.array):
    # Подсчет обратной матрицы
    size = matrix.shape[0]
    # Создание единичной матрицы
    identity_matrix = np.identity(size)
    inv_matrix = np.zeros_like(matrix)
    # Использование метода Гаусса
    for i in range(size):
        inv_matrix.T[i] = gauss(matrix, identity_matrix[i])

    return inv_matrix
```

Реализация интерфейса, позволяющего комфортно взаимодействовать с программой и предоставляющего вывод в удобной для пользователя форме (и в форме, удобной для написания отчета). Заметим, что в программе предусмотрено два варианта ввода систем: из файла и через матрицы внутри самой программы. Предполагается, что реализации всех необходимых функций, описанных ниже и выше находятся в пользовательской библиотеке **methods**.

```
import numpy as np
import methods
import equations

PRECISION = 10

def read_equation(file_name: str):
    with open(file_name, 'r', encoding="utf-8") as file:
        equation = np.array([list(map(float, line.split())) for line in file])

    return equation.T[:-1].T, equation.T[-1]

if __name__ == "__main__":
    mode = input("Введите (yes) для ввода из файла,
    (Enter) для подсчета по умолчанию: ")
    if mode == "yes":
        name = input("Введите имя файла, для выбора по умолчанию (Enter): ")
        a, b = read_equation("equations/test.txt")
    else:
        a, b = equations.option_5

    system = '\\\\\n'.join([' + ' '.join(
    ["{}x_{{{}}}".format(int(a[i, j]), j + 1) if a[i, j] else "" \
    for j in range(a.shape[1])]) + " = {}".format(int(b[i])) \
    for i in range(a.shape[0])])
    print(system)
    det = methods.determinant(a)
    print("DETERMINANT ({}): {}\n".format(
    np.isclose(det, np.linalg.det(a)), det) + "-" * 10)
    if np.isclose(det, 0):
        print("Вырожденная система, единственное решение отсутствует")
        # exit(0)

    print("CONDITION NUMBER: {:.10f}\n".format(methods.cond(a)) + "-" * 10)
    ans = methods.gauss(a, b).round(PRECISION)
    print("GAUSS ANS:\n{}\n".format(ans) + "-" * 10)
    format_ans = ', '.join(["$x_{{{}}} = {}$".format(i + 1, ans[i]) \
    for i in range(len(ans))])
    print("$$" + format_ans + "$$")
```

```
ans = methods.gauss_modified(a, b).round(PRECISION)
print("MODIFIED GAUSS ANS:\n{}\n".format(ans) + "-" * 10)
format_ans = ', '.join(["$x_{{{}}} = {}$".format(i + 1, ans[i]) \
for i in range(len(ans))])
print("$$" + format_ans + "$$")

inv = methods.inv(a)
print("INV MATRIX ({}):\n{}\n".format(
    np.allclose(inv, np.linalg.inv(a)),
    inv.round(PRECISION)) + "-" * 10)
inv = inv.round(4)
format_ans = '\\\\\n'.join([' & '.join(["{}".format(inv[i, j]) \
for j in range(inv.shape[1])]) for i in range(inv.shape[0])])
print(format_ans)

package = methods.relax(a, b, 1, 2500)
if package is not None:
    ans, iterations = package
    ans = ans.round(PRECISION)
    print("RELAX ANS:\n{}\nITERATIONS: {}\n".format(
    ans, iterations) + "-" * 10)
    format_ans = ', '.join(["x_{{{}}} = {}".format(
    i + 1, ans[i]) for i in range(len(ans))])
    print("$$" + format_ans + "$$")
```

Код для второго подварианта

Реализация метода верхней релаксации, который использовался во втором подварианте для решения СЛАУ:

```
import numpy as np
import equations

def relax(input_matrix :np.array, input_b: np.array, \
omiga: float, max_iter: int) -> np.array:
    # Реализация метода верхней релаксации
    matrix = input_matrix.copy()
    b = input_b.copy()
    x = np.ones_like(b)
    size = matrix.shape[0]

    # Преобразования матрицы
    for i in range(size):
        idx = np.argmax(np.abs(matrix[i:, i])) + i
        tmp = matrix[idx].copy()
        matrix[idx] = matrix[i].copy()
        matrix[i] = tmp

    err = 100.
    iterations = 0
    # Установка критериев останова
    while err > 1.e-6 and iterations < max_iter:
        # Вычисление новой итерации
        for i in range(size):
            if matrix[i, i] == 0:
                print("ERROR")
                return None
            x[i] = omiga * (b[i] - np.dot(matrix[i], x)) / matrix[i,i] + x[i]
        iterations += 1
        # Пересчет ошибки
        err = norm(matrix, b, x)
    return x, iterations


def norm(a, b, x):
    # Подсчет второй нормы разности b - Ax
    norm_value = np.linalg.norm(np.dot(a, x) - b, ord=2)
    return norm_value
```

Список литературы

[1] Костомаров Д. П., Фаворский А. П. Вводные лекции по численным методам: Учеб. Пособие. - М.: Университетская книга, Логос, 2006.

[2] Самарский А.А., Гулин А.В. Численные методы, 1989

[3] Тыртышников Е.Е. Конспект лекций по алгебре и геометрии для студентов первого курса факультета ВМК, 2020 (https://m.cs.msu.ru/index.php/s/N6FkcmFbxQkS8z9?path=