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

Гореленковой Анастасии Петровны

гор. Москва

2022 г.

Содержание

1 Подвариант 1

1.1 Постановка задачи

В данном задании необходимо реализовать 2 численных метода, позволяющих решать СЛАУ вида $Ax = b$, где A - невырожденная матрица порядка n, b - вектор-столбец размера $n \times 1$:

- Метод Гаусса;
- Модифицированный метод Гаусса.

Также программа должна вычислять:

- Определитель матрицы $det(A)$;
- Обратную матрицу A^{-1};
- Число обусловленности $M_A = ||A|| \times ||A^{-1}||$.

Кроме собственно реализации алгоритма целью работы является исследование вопроса вычислительной устойчивости метода Гаусса (при больших значениях n).

1.2 Описание алгоритма

Метод Гаусса [1] [2] [3] [4]

Метод состоит из 2 частей: прямого хода, когда матрица приводится к верхнетреугольному виду, и обратного хода, когда осуществляется последовательное отыскание неизвестных $x_1, x_2, ..., x_n$ этой треугольной системы.

Начнём с описания прямого хода. Не ограничивая общности, будем считать, что коэффициент a_{11} (ведущий элемент первого шага) отличен от нуля. В противном случае 1-ое уравнение (строку) поменяем местами с i-ым уравнением, таким что $a_{i1} \neq 0$. Такой номер заведомо найдётся, так как по условию система является невырожденной.

Разделим все члены 1-го уравнения на a_{11}. Далее вычтем из оставшихся уравнений системы преобразованное 1-ое уравнение, умноженное на соответствующие элементы первого столбца. Таким образом, система примет вид:

$$x1 + \frac{a_{12}}{a_{11}}x_2 + \frac{a_{13}}{a_{11}}x_3 + ... + \frac{a_{1n}}{a_{11}}x_n = \frac{b_1}{a_{11}}$$

$$(a_{22} - \frac{a_{12}}{a_{11}}a_{21})x_2 + (a_{23} - \frac{a_{13}}{a_{11}}a_{21})x_3 + ... + (a_{2n} - \frac{a_{1n}}{a_{11}}a_{21})x_n = b_2 - \frac{b_1}{a_{11}}a_{21}$$

...

$$(a_{n2} - \frac{a_{12}}{a_{11}}a_{n1})x_2 + (a_{n3} - \frac{a_{13}}{a_{11}}a_{n1})x_3 + ... + (a_{nn} - \frac{a_{1n}}{a_{11}}a_{n1})x_n = b_n - \frac{b_1}{a_{11}}a_{n1}$$

Заметим, что матрица A приведена к виду:

$$A = \begin{bmatrix} 1 & * & * & \cdots & * \\ 0 & * & * & \cdots & * \\ 0 & * & * & \cdots & * \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & * & * & \cdots & * \end{bmatrix}$$

Далее повторяем действия ещё $n-1$ раз, но уже не со всей матрицей, а с её частью порядка $n-1$ в нижнем правом углу.

По окончании прямого хода матрица коэффициентов будет приведена к верхнетреугольному виду, причём на главной диагонали будут находиться единицы. В процессе обратного хода матрица приводится к единичному виду, начиная с последней строки.

Модифицированный метод Гаусса [1] [2] [3]

Метод аналогичен методу Гаусса, за исключением того, что во время прямого хода очередная строка не сразу делится на свой ведущий элемент. Перед этим в столбце производится поиск максимального по модулю элемента. Соответствующая ему строка (при необходимости) меняется местами с ведущей.

Это позволяет нивелировать ошибки округления, которые только увеличиваются, если на i-ом шаге $|a_{i1}| \ll 1$.

Вычисление определителя матрицы [1] [2] [3]

Может быть произведено во время прямого хода метода Гаусса. Во время его вычисления учитываются следующие свойства определителя:

- Определитель матрицы не меняется, если к какой-то его строке или столбцу прибавить другую строку или столбец, умноженную на некоторое число;
- Определитель матрицы меняет знак на противоположный при перестановке строк или столбцов;
- Определитель матрицы увеличивается в k раз при домножении её строки или столбца на k.

Таким образом, $det(A) = (-1)^k * a_{11} * a_{22} * ... * a_{nn}$, где k - число перестановок, совершённых при прямом ходе метода Гаусса, a_{ii} - ведущий элемент на i-ом ходе.

Нахождение обратной матрицы

При использовании метода Гаусса и его модифицированной версии СЛАУ часто записывают в виде расширенной матрицы $A|b$ размера $n \times (n+1)$. Если вместо вектора b справа от матрицы A дописать единичную матрицу порядка n и выполнять все те же действия со строками, то в результате на месте единичной матрицы образуется обратная матрица. Это связано с тем, что все элементарные преобразования являются домножением матрицы на соответствующие матрицы элементарных преобразований [4], при перемножении которых и получится обратная матрица. Данный метод носит название Гаусса-Жордана.

Нахождение числа обусловленности матрицы [1] [2] [3] [4]

По определению число обусловленности матрицы $M_A = ||A|| \times ||A^{-1}||$, где $||A||$ - бесконечная норма матрицы, которая вычисляется как

$$\max_{1 \le x \le n} \sum_{j=1}^{n} |a_{ij}|$$

Число обусловленности позволяет оценить относительную погрешность решения через относительную погрешность возмущения правой части. Матрицы с большим числом обусловленности и соответствующие им СЛАУ называются плохо обусловленными.

1.3 Тестирование программы

Текст программы находится в Приложении А. Тестирование проводилось с помощью ресурсов online-системы
http://www.wolframalpha.com. Значения приводятся с точностью до 6 знаков после запятой.

1. **СЛАУ задана во входном файле (см. Приложение 1-2 задания)**

   (a) **Файл "gauss1.txt"**

$$A = \begin{pmatrix} 2 & 3 & 11 & 5 \\ 1 & 1 & 5 & 2 \\ 2 & 1 & 3 & 2 \\ 1 & 1 & 3 & 4 \end{pmatrix}, \qquad b = \begin{pmatrix} 2 \\ 1 \\ -3 \\ -3 \end{pmatrix}$$

	WolframAlpha	**Метод Гаусса**	**Модифицированный метод Гаусса**
$x =$	$\begin{bmatrix} -2 \\ 0 \\ 1 \\ -1 \end{bmatrix}$	$\begin{bmatrix} -1.999999 \\ 0.000000 \\ 0.999999 \\ -1.0 \end{bmatrix}$	$\begin{bmatrix} -1.999999 \\ 0.000000 \\ 0.999999 \\ -1.0 \end{bmatrix}$
$det(A) =$	14	14.0	14.0
$M_A =$	99	98.999999	98.999999

WolframAlpha:

$$A^{-1} = \begin{pmatrix} -0.285714 & 0.285714 & 0.714286 & -0.142857 \\ 1.28571 & -2.78571 & 0.285714 & -0.357143 \\ -0.142857 & 0.642857 & -0.142857 & -0.0714286 \\ -0.142857 & 0.142857 & -0.142857 & 0.428571 \end{pmatrix}$$

Метод Гаусса:

$$A^{-1} = \begin{pmatrix} -0.285714 & 0.285714 & 0.714285 & -0.142857 \\ 1.285714 & -2.785714 & 0.285714 & -0.357143 \\ -0.142857 & 0.642857 & -0.142857 & -0.071429 \\ -0.142857 & 0.142857 & -0.142857 & 0.428571 \end{pmatrix}$$

Модифицированный метод Гаусса:

$$A^{-1} = \begin{pmatrix} -0.285714 & 0.285714 & 0.714286 & -0.142857 \\ 1.285714 & -2.785714 & 0.285714 & -0.357143 \\ -0.142857 & 0.642857 & -0.142857 & -0.071429 \\ -0.142857 & 0.142857 & -0.142857 & 0.428571 \end{pmatrix}$$

(b) **Файл "gauss2.txt"**

$$A = \begin{pmatrix} 2 & -1 & 1 & 2 \\ 6 & -3 & 2 & 4 \\ 6 & -3 & 4 & 8 \\ 4 & -2 & 1 & 1 \end{pmatrix} = 1, \qquad b = \begin{pmatrix} 2 \\ 3 \\ 9 \\ 1 \end{pmatrix}$$

Матрица вырожденная ($det(A) = 0$) $\Rightarrow$ единственного решения нет. Программа корректно обрабатывает подобную ситуацию.

(c) **Файл "gauss3.txt"**

$$A = \begin{pmatrix} 1 & 1 & -3 & 1 \\ 2 & 1 & -2 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 2 & -3 & 7 \end{pmatrix} = 1, \qquad b = \begin{pmatrix} -1 \\ 1 \\ 3 \\ 1 \end{pmatrix}$$

	WolframAlpha	**Метод Гаусса**	**Модифицированный метод Гаусса**
$x =$	$\begin{bmatrix} 1.15 \\ 0.8 \\ 1.05 \\ 0.2 \end{bmatrix}$	$\begin{bmatrix} 1.150000 \\ 0.800000 \\ 1.05 \\ 0.2 \end{bmatrix}$	$\begin{bmatrix} 1.15 \\ 0.8 \\ 1.05 \\ 0.2 \end{bmatrix}$
$det(A) =$	-20	-20.0	-20.0
$M_A =$	52	52.0	52.0

WolframAlpha:

$$A^{-1} = \begin{pmatrix} -1.05 & 1.15 & -0.4 & 0.15 \\ 1.4 & -1.2 & 1.2 & -0.2 \\ -0.35 & 0.05 & 0.2 & 0.05 \\ -0.4 & 0.2 & -0.2 & 0.2 \end{pmatrix}$$

Метод Гаусса:

$$A^{-1} = \begin{pmatrix} -1.049999 & 1.15 & -0.400000 & 0.150000 \\ 1.4 & -1.2 & 1.200000 & -0.2 \\ -0.35 & 0.05 & 0.2 & 0.05 \\ -0.4 & 0.2 & -0.2 & 0.2 \end{pmatrix}$$

Модифицированный метод Гаусса:

$$A^{-1} = \begin{pmatrix} -1.049999 & 1.15 & -0.399999 & 0.149999 \\ 1.4 & -1.2 & 1.2 & -0.199999 \\ -0.35 & 0.0499999 & 0.2 & 0.0499999 \\ -0.4 & 0.2 & -0.2 & 0.2 \end{pmatrix}$$

2. **СЛАУ задана с помощью специальной функции (см. Приложение 2 (п.1-6) задания)**

$n = 25, m = 10$

$$A_{ij} = \begin{cases} \frac{i+j}{m+n} = \frac{i+j}{35} & , i \neq j \\ n + m^2 + \frac{j}{m} + \frac{i}{n} = 125 + \frac{j}{10} + \frac{i}{25} & , i = j \end{cases}, \qquad b_i = i^2 - n = i^2 - 25$$

	Метод Гаусса	**Модифицированный метод Гаусса**
$x =$	$\begin{bmatrix} -0.332676 \\ -0.330981 \\ -0.313308 \\ -0.279691 \\ -0.230161 \\ -0.164749 \\ -0.083486 \\ 0.013595 \\ 0.126464 \\ 0.255089 \\ 0.399439 \\ 0.559484 \\ 0.735191 \\ 0.926532 \\ 1.133473 \\ 1.355986 \\ 1.594036 \\ 1.847601 \\ 2.116642 \\ 2.401133 \\ 2.701042 \\ 3.016339 \\ 3.346993 \\ 3.692976 \\ 4.054257 \end{bmatrix}$	$\begin{bmatrix} -0.332676 \\ -0.330981 \\ -0.313308 \\ -0.279691 \\ -0.230161 \\ -0.164749 \\ -0.083486 \\ 0.013595 \\ 0.126463 \\ 0.255080 \\ 0.399439 \\ 0.559484 \\ 0.735191 \\ 0.926531 \\ 1.133473 \\ 1.355986 \\ 1.5940386 \\ 1.847601 \\ 2.116642 \\ 2.401133 \\ 2.701042 \\ 3.016336 \\ 3.346993 \\ 3.692976 \\ 4.054257 \end{bmatrix}$
$det(A) =$	-3.658170	-3.658170
$M_A =$	1.394566	1.394566

Обратная матрица, в силу размера (625 элементов), опущена.

1.4 Вывод

Согласно рассмотренным примерам модифицированный метод Гаусса показывает себя лучше классического метода Гаусса. Но из-за кубической сложности при большом порядке матрицы A он выполняется достаточно долго, значит не является оптимальным.

2 Подвариант 2

2.1 Постановка задачи

В данном задании необходимо реализовать метод верхней релаксации (и его частный случай - метод Зейделя), позволяющий решать СЛАУ вида $Ax = b$, где A - невырожденная матрица порядка n, x - вектор-столбец размера $n \times 1$. В частности, разработать критерий остановки итерационного процесса, который гарантирует получение решения СЛАУ с заданной точностью.

Кроме собственно реализации алгоритма целью работы является изучение скорости сходимости метода к точному решению задачи при различных значениях параметра ω (см. раздел 2.2).

2.2 Описание алгоритма

Итерационные методы [1] [2] [3]

Как уже было отмечено ранее, существуют плохо обусловленные матрицы. При малом возмущении правой части, решение соответствующего им СЛАУ приводит к значительным отклонениям, причём с ростом размера матрицы возрастает и шанс столкнуться с подобной ситуацией.

С такими проблемами успешно справляются итерационные методы решения СЛАУ. В таком случае ответ получается в процессе построения последовательных приближений (итераций), сходящихся к решению системы по евклидовой норме. Каноническое уравнение итерационного метода имеет вид:

$$B_{k+1}\frac{x_{k+1}-x_k}{\tau_{k+1}} + Ax_k = b, \qquad det(B_k) \neq 0, \qquad \tau_k > 0,$$

где B_k и τ_k - итерационные параметры (зависят от метода). Если итерационные параметры не зависят от индекса k, то итерационный процесс называется стационарным.

Метод Зейделя и обобщающий его метод верхней релаксации являются итерационными методами решения СЛАУ.

Метод Зейделя [1] [2] [3]

Представим матрицу коэффициентов в виде суммы:

$$A = D + T_B + T_H,$$

$$\text{где } D_{ij} = \begin{cases} 0 & ,i \neq j \\ a_{ii} & ,i = j \end{cases} \text{ - диагональная матрица,}$$

$$(T_B)_{ij} = \begin{cases} 0 & ,i \geq j \\ a_{ij} & ,i < j \end{cases} \text{ - верхнетреугольная матрица,}$$

$$(T_H)_{ij} = \begin{cases} a_{ij} & ,i > j \\ 0 & ,i \leq j \end{cases} \text{ - нижнетреугольная матрица.}$$

В каноническом уравнении итерационного метода примем $B_k = D + T_H$ и $\tau_k = 1$. Тогда формула примет вид:

$$(D + T_H)(x_{k+1} - x_k) + Ax_k = b \Leftrightarrow (D + T_H)x_{k+1} + T_B x_k = b$$

Метод верхней релаксации [1] [2] [3]

В каноническом уравнении итерационного метода примем $B_k = D + \omega T_H$ и $\tau_k = \omega$. Тогда формула примет вид:

$$(D + \omega T_H)\frac{x_{k+1} - x_k}{\omega} + Ax_k = b \Leftrightarrow (\frac{1}{\omega}D + T_H)x_{k+1} + ((1 - \frac{1}{\omega})D + T_B)x_k = b$$

Очевидно, что метод Зейделя является частным случаем при $\omega = 1$

Перейдём от векторной записи реккурентной формулы к построчной:

$$x_i^{k+1} = x_i^k + \frac{\omega}{a_{ii}}(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{k+1} - \sum_{j=i}^{n} a_{ij}x_j^k), \qquad i = 1..n.$$

Сходимость метода вариации постоянных [1] [2] [3]

Заметим, что метод сходится для положительно определённым матриц:

$$((B - \frac{\tau}{2}A)x, x) = (1 - \frac{\omega}{2})(Dx, x) > 0.$$

Если предположить, что матрица A - симметрическая положительно опердедённая, то можно вывести достаточное условие сходимости метода: $0 < \omega < 2$.

Поскольку на практике проверку на положительную определённость матрицы провести достаточно трудно, её часто опускают.

Критерий остановки итерационного процесса [1] [2] [3] [5]

Пусть ϵ - заданная точность. В качестве критерия остановки возьмём $||x_{k+1} - x_k|| < \epsilon$, где $||x|| = \sum\limits_{i=1}^{n} |x_i|$ - первая норма Гёльдера.

2.3 Тестирование программы

Текст программы находится в Приложении В. Тестирование проводилось с помощью ресурсов online-системы http://www.wolframalpha.com. Значения приводятся с точностью до 6 знаков после запятой.

Для исследования скорости сходимости метода верхней релаксации каждый из следующих примеров вычислялся при $\omega = 0.1; 0.2; ...; 1.9$. В соответствующих таблицах указаны значения для ω (не совпадающие с 1 - значением параметра в методе Зейделя), давшие наилучший и наихудший по количеству итераций результат.

1. **СЛАУ задана во входном файле**
   СЛАУ из Приложения 1-2 не сходятся за 100000 итераций (программа в таких случаях завершается) $\Rightarrow$ не являются положительно определёнными. Вместо них в указанных ниже файлах заданы подходящие матрицы.

   (a) **Файл "relax1.txt"**

$$A = \begin{pmatrix} 2 & 3 & 11 & 5 \\ 1 & 1 & 5 & 2 \\ 2 & 1 & 3 & 2 \\ 1 & 1 & 3 & 4 \end{pmatrix}, \qquad b = \begin{pmatrix} 2 \\ 1 \\ -3 \\ -3 \end{pmatrix}$$

	WolframAlpha	**Метод Зейделя**	**Метод верхней релаксации (лучший результат)**	**Метод верхней релаксации (худший результат)**
$x =$	$\begin{bmatrix} -7 \\ 3 \\ 10 \\ 1 \end{bmatrix}$	$\begin{bmatrix} -7.0 \\ 3.0 \\ 9.999999 \\ 1.0 \end{bmatrix}$	$\begin{bmatrix} -6.999999 \\ 2.999999 \\ 10.0 \\ 0.999999 \end{bmatrix}$	$\begin{bmatrix} -6.999999 \\ 2.999999 \\ 10.000000 \\ 0.999999 \end{bmatrix}$
Кол-во итераций		11	10	243
$\omega =$		1	1.1	0.1

(b) **Файл "relax2.txt"**

$$A = \begin{pmatrix} 7 & 1 & 0 & 0 \\ 1 & 7 & 1 & 0 \\ 0 & 1 & 7 & 1 \\ 0 & 0 & 1 & 7 \end{pmatrix} = 1, \qquad b = \begin{pmatrix} 9 \\ 18 \\ 27 \\ 31 \end{pmatrix}$$

	WolframAlpha	**Метод Зейделя**	**Метод верхней релаксации (лучший результат)**	**Метод верхней релаксации (худший результат)**
$x =$	$\begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}$	$\begin{bmatrix} 1.0 \\ 1.999999 \\ 3.0 \\ 3.999999 \end{bmatrix}$	$\begin{bmatrix} 1.0 \\ 2.0 \\ 2.999999 \\ 4.0 \end{bmatrix}$	$\begin{bmatrix} 1.000000 \\ 1.999999 \\ 3.000000 \\ 3.999999 \end{bmatrix}$
Кол-во итераций		9	10	170
$\omega =$		1	1.1	0.1

(c) **Файл "relax3.txt"**

$$A = \begin{pmatrix} 400 & 1 & 1 & 1 \\ 1 & 400 & 1 & 1 \\ 1 & 1 & 400 & 1 \\ 1 & 1 & 1 & 400 \end{pmatrix} = 1, \qquad b = \begin{pmatrix} 10 \\ 20 \\ 30 \\ 40 \end{pmatrix}$$

	WolframAlpha	**Метод Зейделя**	**Метод верхней релаксации (лучший результат)**	**Метод верхней релаксации (худший результат)**
$x =$	$\begin{bmatrix} 0.024441 \\ 0.049503 \\ 0.074566 \\ 0.099629 \end{bmatrix}$	$\begin{bmatrix} 0.0244406 \\ 0.049503 \\ 0.074566 \\ 0.099629 \end{bmatrix}$	$\begin{bmatrix} 0.024441 \\ 0.049503 \\ 0.074566 \\ 0.099629 \end{bmatrix}$	$\begin{bmatrix} 0.024441 \\ 0.049503 \\ 0.074566 \\ 0.099629 \end{bmatrix}$
Кол-во итераций		4	8	119
$\omega =$		1	0.9 и 1.1[1]	0.1

[1] В данном примере алгоритм, запущенный при $\omega = 0.9$ и $\omega = 1.1$, дал результаты, которые оказались равны вплоть до полученных значений.

2. **СЛАУ задана с помощью специальной функции (см. Приложение 2 (п.1-6) задания)**

$n = 25, m = 10$

$$A_{ij} = \begin{cases} \frac{i+j}{m+n} = \frac{i+j}{35} & , i \neq j \\ n + m^2 + \frac{j}{m} + \frac{i}{n} = 125 + \frac{j}{10} + \frac{i}{25} & , i = j \end{cases}, \qquad b_i = i^2 - n = i^2 - 25$$

Обратная матрица, в силу размера (625 элементов), опущена.

	Метод Зейделя	**Метод верхней релаксации (лучший результат)**	**Метод верхней релаксации (худший результат)**
$x =$	$\begin{bmatrix} -0.332676 \\ -0.330980 \\ -0.313308 \\ -0.279691 \\ -0.230161 \\ -0.164749 \\ -0.083486 \\ 0.013595 \\ 0.126464 \\ 0.255089 \\ 0.399439 \\ 0.559484 \\ 0.735191 \\ 0.926532 \\ 1.133473 \\ 1.355986 \\ 1.594039 \\ 1.847601 \\ 2.116642 \\ 2.401133 \\ 2.701042 \\ 3.016339 \\ 3.346993 \\ 3.692976 \\ 4.054257 \end{bmatrix}$	$\begin{bmatrix} -0.332676 \\ -0.330981 \\ -0.313308 \\ -0.279691 \\ -0.230161 \\ -0.164749 \\ -0.083486 \\ 0.013595 \\ 0.126464 \\ 0.255089 \\ 0.399439 \\ 0.559484 \\ 0.735191 \\ 0.926536 \\ 1.133473 \\ 1.355986 \\ 1.594039 \\ 1.847601 \\ 2.116642 \\ 2.401133 \\ 2.701042 \\ 3.016339 \\ 3.346994 \\ 3.692976 \\ 4.054257 \end{bmatrix}$	$\begin{bmatrix} -0.332676 \\ -0.330980 \\ -0.313308 \\ -0.279691 \\ -0.230161 \\ -0.164749 \\ -0.083486 \\ 0.013595 \\ 0.126464 \\ 0.255089 \\ 0.399439 \\ 0.559484 \\ 0.735191 \\ 0.926532 \\ 1.133473 \\ 1.355986 \\ 1.5940386 \\ 1.847601 \\ 2.116642 \\ 2.401133 \\ 2.701042 \\ 3.016338 \\ 3.346993 \\ 3.692976 \\ 4.054257 \end{bmatrix}$
Кол-во итераций	7	10	165
$\omega =$	1	0.9	0.1

2.4 Вывод

Согласно рассмотренным примерам метод верхних релаксаций при $n \gg 1$ оказывается быстрее метода Гаусса. Но учитывая ограничения на применимость метода верхнней релаксации, проблемы с проверкой положительной определённости матрицы и существование дополнительной задачи подбора наилучшего значения параметра ω, метод верхних релаксаций не является однозначно лучшей альтернативой методу Гаусса. Каждый из методов целесообразно выбирать в зависимости от ситуации.

Также отметим, что на симметричных относительно главной диагонали матрицах, метод Зейделя стабильно оказывается быстрее метода верхней релаксации на всех проверенных значениях ω.

A Приложение 1: метод Гаусса

Текст программы gauss_method.py на языке Python (специальные вспомогательные функции ввода и вывода опущены):

```
<...>

""" Вычисление бесконечной нормы марицы """
def matrix_norm(matrix):
    res = 0
    for i in range(len(matrix)):
        tmp = 0
        for j in range(len(matrix[0])):
            tmp += abs(matrix[i][j])
        if tmp > res:
            res = tmp
    return res

""" Вычисление числа обусловленности матрицы """

def conditional_number(matrix, method, det):
    return matrix_norm(matrix) * matrix_norm(inverse_matrix(matrix,
method, det))

""" Вычисление обратной матрицы """
def inverse_matrix(matrix, method, det):
    mtrx = method(augmented_matrix(matrix,
identity_matrix(len(matrix))), det)
    res = get_answer(mtrx)
    return res

""" Функция возвращает расширенную матрицу """
def augmented_matrix(matrix1, matrix2):
    res = []
    for i in range(len(matrix1)):
        res.append(matrix1[i] + matrix2[i])
    return res

""" Функция извлекает из расширенной матрицы ответ
(вектор при решении СЛАУ/матрицу при
поиске обратной матрицы) """
def get_answer(matrix):
    if matrix == -1:
        return -1
```

```
    ans = []
    ln = len(matrix)
    for i in range(ln):
        line = []
        for j in range(ln, len(matrix[0])):
            line.append(matrix[i][j])
        ans.append(line)
    return ans

""" Элементарные преобразования строчек матрицы """
def swap_lines(matrix, i, j): # функция меняет i-ую и j-ую строки
местами
    matrix[i], matrix[j] = matrix[j], matrix[i]

def mul_line(matrix, i, num): # функция умножает i-ую строку матрицы
на число num
    for j in range(len(matrix[0])):
        matrix[i][j] *= num

def add_lines(matrix, i1, i2, coef): # функция добавляет и i1-ой
строке i2-ую с коэффициентом coef
    for j in range(len(matrix[0])):
        matrix[i1][j] += coef * matrix[i2][j]

""" Метод Гаусса """
""" В зависимости от расширенной матрицы (матрица A +
вектор-столбец b / единичная матрица) вычисляет либо решение СЛАУ,
либо обратную матрицу """
def gauss_method(matrix, det):
    n = len(matrix)
    swap_counter = 0
    det[0] = 1

    # Прямой ход

    for j in range(n):
        for i in range(j, n):
            if matrix[i][j] != 0:
                swap_lines(matrix, j, i)
                swap_counter += 1
                break

        det[0] *= matrix[j][j]
        if matrix[j][j] == 0:
            print("\nМатрица - вырождена.
Это противоречит условию")
```

```
            return -1

        mul_line(matrix, j, 1 / matrix[j][j])
        for i in range(j + 1, n):
            coef = -matrix[i][j]
            add_lines(matrix, i, j, coef)

    det[0] *= (-1) ** (swap_counter

    # Обратный ход

    for j in range(n - 1, -1, -1):
        mul_line(matrix, j, 1 / matrix[j][j])
        for i in range(j - 1, -1, -1):
            coef = -matrix[i][j]
            add_lines(matrix, i, j, coef)
    return matrix

""" Метод Гаусса с выбором главного элемента """
""" В зависимости от расширенной матрицы (матрица A +
вектор-столбец b / единичная матрица) вычисляет либо решение СЛАУ,
либо обратную матрицу """
def modified_gauss_method(matrix, det):
    n = len(matrix)
    swap_counter = 0
    det[0] = 1

    # Прямой ход

    for j in range(n):
        leading_elem = abs(matrix[j][j])
        leading_elem_num = j
        for i in range(j, n):
            cur_elem = abs(matrix[i][j])
            if cur_elem > leading_elem:
                leading_elem = cur_elem
                leading_elem_num = i
        swap_counter += 1
        swap_lines(matrix, j, leading_elem_num)

        det[0] *= matrix[j][j]
        if matrix[j][j] == 0:
            print("\nМатрица - вырождена.
Это противоречит условию")
            return -1
```

```
            mul_line(matrix, j, 1 / matrix[j][j])
            for i in range(j + 1, n):
                coef = -matrix[i][j]
                add_lines(matrix, i, j, coef)

    det[0] *= (-1) ** (swap_counter

    # Обратный ход

    for j in range(n - 1, -1, -1):
            mul_line(matrix, j, 1 / matrix[j][j])
            for i in range(j - 1, -1, -1):
                coef = -matrix[i][j]
                add_lines(matrix, i, j, coef)

    return matrix

<...>
```

В Приложение 2: метод верхних релаксаций

Текст программы upper_relaxation_method.py на языке Python (специальные вспомогательные функции ввода и вывода опущены):

```
from copy import deepcopy

<...>
""" Вычисление нормы для условия окончания
метода верхней релаксации """
def norm(v1, v2):
    res = 0
    n = len(v1)
    for i in range(n):
        res += abs(v1[i] - v2[i])
    return res

""" Метод верхней релаксации """
""" (Метод Зейделя - частный случай метода верхней релаксации
при w = 1) """
def upper_relaxation_method(a, b, eps, w):
    n = len(a)
    cur_x = [0 for _ in range(n)]
    prev_x = [0 for _ in range(n)]
    iterations = 0
    flag = True # цикл должен выполниться хотя бы 1 раз
    while norm(cur_x, prev_x) > eps or flag:
        prev_x = deepcopy(cur_x)
        for i in range(n):
            tmp = 0
            for j in range(i):
                tmp += a[i][j] * cur_x[j]
            for j in range(i, n):
                tmp += a[i][j] * prev_x[j]
            cur_x[i] = prev_x[i] + w * (b[i] - tmp) / a[i][i]
        iterations += 1
        if iterations > 100000:
            print("Матрица не является положительно определённой")
            return -1, iterations
        flag = False
    for i in range(len(cur_x)):
        cur_x[i] = int(cur_x[i] * 100000000) /100000000
    return cur_x, iterations
<...>
```

Список литературы

[1] Костомаров Д. П., Фаворский А. П. Вводные лекции по численным методам. — Москва: Логос, 2004.

[2] Бахвалов Н. С., Жидков Н. П., Кобельков Г. М. Численные методы - 8-е изд. (эл.). — Москва: БИНОМ. Лаборатория знаний, 2015.

[3] Самарский А. А. Введение в численные методы. Учебное пособие для вузов - 3-е изд. — Санкт-Петербург: Лань, 2005.

[4] Тыртышников Е. Е. Матричный анализ и линейная алгебра. — Москва: Физматлит, 2007.

[5] Ильин В. А., Ким Г. Д. Линейная алгебра и аналитическая геометрия : учеб. - 3-е изд., перераб. и доп. — Москва: ТК Велби, Изд-во Проспект, 2007.