



Министерство науки и высшего образования Российской Федерации  
федеральное государственное бюджетное образовательное учреждение  
высшего профессионального образования  
«Московский государственный технический университет имени  
Н.Э. Баумана (национальный исследовательский университет)»  
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Робототехники и комплексной автоматизации»  
КАФЕДРА «Системы автоматизированного проектирования (РК-6)»

## ОТЧЕТ О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ №4

по дисциплине «Вычислительная математика»

Студент:	Киммель Анастасия Александровна
Группа:	РК6-55Б
Тип задания:	лабораторная работа
Тема:	Устойчивость прямых методов решения СЛАУ

Студент

\_\_\_\_\_  
подпись, дата

Киммель А.А.  
Фамилия, И.О.

Преподаватель

\_\_\_\_\_  
подпись, дата

\_\_\_\_\_  
Фамилия, И.О.

Москва, 2023

## Содержание

<b>Устойчивость прямых методов решения СЛАУ</b>	<b>3</b>
Задание . . . . .	3
Цель выполнения лабораторной работы . . . . .	4
1    Базовая часть . . . . .	5
2    Продвинутая часть . . . . .	13
Заключение . . . . .	20

# Устойчивость прямых методов решения СЛАУ

## Задание

Решение систем линейных алгебраических уравнений (СЛАУ) - ключевой этап огромного множества задач вычислительной математики и анализа данных. В случае плотных матриц сравнительно небольшой размерности, для нахождения решения СЛАУ часто применяются прямые методы, такие как метод Гаусса, метод прогонки или разложение Холецкого. В то же время известно, что многие прямые методы обладают вычислительной неустойчивостью и могут приводить к некорректному решению для некоторых матриц коэффициентов. В этой лабораторной работе рассматриваются матрицы нескольких видов и с помощью генерации большого количества случайных матриц демонстрируется наличие или отсутствие вычислительной неустойчивости у метода Гаусса, метода прогонки и разложения Холецкого.

### Требуется (базовая часть):

1. Написать функцию `gauss(A, b, pivoting)`, которая возвращает решение СЛАУ  $Ax=b$ , полученное с помощью метода Гаусса. Если параметр `pivoting=True`, то решение должно находиться с частичным выбором главного элемента. Если `pivoting=False`, то выбора главного элемента происходить не должно.
2. Написать функцию `thomas(A, b)`, которая возвращает решение СЛАУ  $Ax=b$ , полученное с помощью метода прогонки.
3. Среди реализованных методов, включая два варианта метода Гаусса, выбрать тот, который минимизирует вычислительные погрешности для случая квадратных матриц общего вида. В рамках задания такой метод будем называть "универсальным".
4. Разработать и описать алгоритм генерации случайных невырожденных матриц размерности  $6 \times 6$  с элементами  $a_{ij} \in \mathbb{R}, |a_{ij}| < 1$  общего и 3-х диагонального вида.
5. Сгенерировав 1000 случайных матриц  $A^{(j)}$  каждого типа с 32-битными float-представлением элементов, необходимо провести следующий эксперимент:
  - (a) Выбрать "специальный" вычислительно-эффективный метод по типу матрицы.
  - (b) Для каждой СЛАУ  $A^{(j)}x = [1, 1, 1, 1]^T$  найти решение с помощью "универсального" и "специального" методов, а затем найти относительную погрешность вычислений с помощью среднеквадратичной и супремум-нормы. Вывести на экран распределения погрешностей в виде гистограмм.
  - (c) Является ли выбранный "специальный" метод вычислительно устойчивым? Почему?

### Требуется (продвинутая часть):

1. Расширить генератор из задания 4 для получения положительно-определенных матриц.
2. Написать функцию `cholesky(A,b)`, которая возвращает решение СЛАУ  $Ax = b$ , полученное с помощью разложения Холецкого.
3. Провести требуемый в задании 5 анализ, учитывая метод Холецкого (сравнить с "универсальным" методом).
4. Для всех рассмотренных ранее матриц вывести на экран распределение спектральных радиусов и распределение чисел обусловленности в виде гистограмм. Сделать вывод.
5. Влияет ли значение спектрального радиуса матрицы на вычислительную устойчивость рассмотренных алгоритмов? Если да, то как?
6. Влияет ли отношение максимального по модулю собственного числа к минимальному по модулю собственному числу на вычислительную устойчивость алгоритма? Если да, то как?
7. Влияет ли число обусловленности на вычислительную устойчивость алгоритма? Если да, то как?

### Цель выполнения лабораторной работы

Цель работы: изучение методов решения СЛАУ, а именно: два метода Гаусса, прогонки и Холецкого. Также необходимо вычислить относительную погрешность методов и провести анализ результатов.

## 1 Базовая часть

В базовой части лабораторной работы необходимо разработать три метода решения СЛАУ и написать алгоритмы генерации матриц общего и трехдиагонального видов.

### Метод Гаусса

Пусть дано СЛАУ в виде:  $Ax = b$ . В дальнейшем на протяжении всей лабораторной работы будем рассматривать СЛАУ в данном виде.

Метод Гаусса - это один из методов решения СЛАУ для матрицы коэффициентов  $A$  общего вида. Метод Гаусса заключается в том, чтобы привести матрицу  $A$  к треугольному виду с помощью элементарных преобразований, после чего решение  $x$  находится довольно просто. Рассмотрим расширенную матрицу СЛАУ:

$$\tilde{A} = \left( \begin{array}{cccc|c} a_{11} & a_{12} & \dots & a_{1n} & b_1 \\ a_{21} & a_{22} & \dots & a_{2n} & b_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} & b_n \end{array} \right)$$

Необходимо привести матрицу  $\tilde{A}$  к верхнему треугольному виду, указанном в формуле (1):

$$\tilde{A} = \left( \begin{array}{cccc|c} a_{11} & a_{12} & \dots & \dots & a_{1n} & b_1 \\ 0 & a_{22}^{(1)} & \dots & \dots & a_{2n}^{(1)} & b_2^{(1)} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & a_{n-1,n-1}^{(n-2)} & a_{n-1,n}^{(n-2)} & b_n^{(n-2)} \\ 0 & 0 & \dots & 0 & a_{nn}^{(n-1)} & b_n^{(n-1)} \end{array} \right) \quad (1)$$

Коэффициенты  $a$  и  $b$  получаются в ходе элементарных преобразований расширенной матрицы, общая формула для нахождения коэффициентов представлена в формуле (2):

$$\begin{aligned} a_{ij}^{(k)} &= a_{ij} - \frac{a_{ik}}{a_{kk}} a_{kj} \\ b_i^{(k)} &= b_i - \frac{a_{ik}}{a_{kk}} b_k, \end{aligned} \quad (2)$$

где  $k \in [1; n-1]$ .

Далее приведем способ нахождения корней СЛАУ  $x$  с помощью обратного хода

метода Гаусса (формула (3)):

$$\begin{aligned}
 x_n &= \frac{b_n^{(n-1)}}{a_{nn}^{(n-1)}} \\
 x_{n-1} &= \frac{b_{n-1}^{(n-2)} - a_{n-1,n}^{(n-2)} x_n}{a_{n-1,n-1}^{(n-2)}}, \\
 &\dots \\
 x_1 &= \frac{b_1 - \sum_{i=2}^n a_{1i} x_i}{a_{11}}
 \end{aligned} \tag{3}$$

Также необходимо рассмотреть ситуацию, когда  $a_{kk}^{(k)} = 0$ , т.е. при поиске **коэффициентов** происходит деление на ноль. Эта проблема решается соответствующим переставлением строк: строки матрицы переставляются так, что диагональным элементом становится наибольший по модулю элемент  $k$ -го столбца:

$$|a_{kk}^{(k)}| = \max_{k \leq i \leq n} |a_{ik}^{(k)}|$$

Реализация двух вариаций метода Гаусса представлена в **Листинге 1**. Аргумент `pivoting` отвечает за выбор метода: если `pivoting=True`, то решение находится с выбором главного элемента, иначе выбор главного элемента происходит.

Листинг 1. Функция `gauss`

---

```

1 def gauss(A, b, pivoting):
2     n = len(A)
3     if pivoting:
4         for i in range(n - 1):
5             column_max = i + np.argmax(np.abs(A[i:, i]), axis=0)
6             A_temp = np.array([A[i], A[column_max]])
7             A[i] = A_temp[1]
8             A[column_max] = A_temp[0]
9
10    for row in range(n):
11        for i in range(row + 1, n):
12            frac = A[i][row] / A[row][row]
13            for j in range(n):
14                A[i][j] = A[i][j] - frac * A[row][j]
15            b[i] = b[i] - frac * b[row]
16
17    x = np.empty(n)
18    for i in range(n - 1, -1, -1):
19        summ = 0
20        for j in range(i + 1, n):
21            summ += A[i][j] * x[j]
22        x[i] = (b[i] - summ) / A[i][i]
23    return x

```

---

## Метод прогонки

Метод прогонки - это метод решения СЛАУ, имеющих трехдиагональную матрицу коэффициентов  $A$ :

$$A = \begin{bmatrix} a_{11} & a_{12} & 0 & \dots & \dots & \dots & 0 \\ a_{21} & a_{22} & a_{23} & \ddots & & & \vdots \\ 0 & a_{32} & a_{33} & a_{34} & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & & \ddots & a_{n-1,n-2} & a_{n-1,n-1} & a_{n-1,n} \\ 0 & \dots & \dots & \dots & 0 & a_{n,n-1} & a_{nn} \end{bmatrix}$$

Метод прогонки заключается в том, чтобы преобразовать трехдиагональную матрицу к верхней треугольной форме

Введем коэффициенты  $\gamma$  и  $\beta$ :

$$\gamma_{i+1} = \frac{-a_{i,i+1}}{a_{i,i-1}\gamma_i + a_{ii}}$$

$$\beta_{i+1} = \frac{b_i - a_{i,i-1}\beta_i}{a_{i,i-1}\gamma_i + a_{ii}},$$

Тогда решение СЛАУ находится следующим образом:

$$x_{i-1} = \gamma_i x_i + \beta_i$$

Реализация метода прогонки представлена в Листинге 2:

Листинг 2. Функция thomas

---

```

1 def thomas(A, b):
2     n = len(A)
3     gamma = np.empty(n, dtype=np.float32)
4     beta = np.empty(n, dtype=np.float32)
5     x = np.empty(n, dtype=np.float32)
6     gamma[0] = beta[0] = 0
7     for i in range(n - 1):
8         gamma[i + 1] = -A[i][i + 1] / (A[i][i - 1] * gamma[i] + A[i][i])
9         beta[i + 1] = (b[i] - A[i][i - 1] * beta[i]) / (A[i][i - 1] * gamma[i] + A[i][i])
10    x[n - 1] = (b[n - 1] - A[n - 1][n - 2] * beta[n - 1]) / (A[n - 1][n - 1] + A[n - 1][n
        - 2] * gamma[n - 1])
11    for i in range(n - 1, 0, -1):
12        x[i - 1] = gamma[i] * x[i] + beta[i]
13    return x

```

---

## Выбор универсального метода

Был выбран метод Гаусса с частичным выбором главного элемента в качестве "универсального" метода. В отличие от метода прогонки, который используется для решения систем линейных уравнений с трёхдиагональной матрицей коэффициентов, метод Гаусса применим к произвольным матрицам. Частичный выбор главного элемента в методе Гаусса решает проблему возникновения нулей на диагонали и деления на малые значения, что помогает уменьшить накопленные погрешности.

## Генерация матриц

Для применения разработанных методов Гаусса и прогонки необходимо реализовать алгоритм генерации невырожденных матриц общего вида и трехдиагональных матриц. Также по условию требуется, чтобы матрицы были с элементами  $a_{ij} \in \mathbb{R}, |a_{ij}| < 1$ .

По определению невырожденной матрицы - это такая матрицы, определитель которой не равен 0. Так как вероятность того, что у сгенерированной матрицы определитель будет равен 0 очень маленькая, то можем просто добавить в алгоритм проверку определителя. Для нахождения определителя будем использовать встроенную функцию библиотеки numpy `np.linalg.det()`.

Генерацию трехдиагональных матриц будем осуществлять построчно относительно её главной диагонали, т.е. индекс столбца будет варьироваться в пределах  $[i - 1; i + 1]$ ., кроме первой и последней строки.

Для генерации случайных чисел будем использовать функцию `np.random.uniform`, аргументы которой задают границы генерации чисел. Но так как нижняя граница применяется включительно, то введем очень маленькое число, близкое к -1.

Алгоритмы генерации двух типов матриц представлены в Листинге 3.

Листинг 3. Функции генерации матриц

```

1 def matrix_generation(n):
2     matrix = np.random.uniform(-1.000001, 1, (n, n)).astype(np.float32)
3     while np.linalg.det(matrix) == 0:
4         matrix = np.random.uniform(-1.000001, 1, (n, n)).astype(np.float32)
5     return matrix
6
7
8 def three_diag(n):
9     matrix = np.zeros((n, n), dtype=np.float32)
10    matrix[0][0] = np.random.uniform(-1.000001, 1)
11    matrix[0][1] = np.random.uniform(-1.000001, 1)
12    for i in range(1, n - 1):
13        for j in range(i - 1, i + 2):
14            matrix[i][j] = np.random.uniform(-1.000001, 1)
15    matrix[n - 1][n - 2] = np.random.uniform(-1.000001, 1)

```



```

16     matrix[n - 1][n - 1] = np.random.uniform(-1.000001, 1)
17     return matrix

```

---

### Относительная погрешность методов

Далее необходимо выбрать "специальный" метод для каждого типа матрицы и вычислить его относительную погрешность с помощью среднеквадратичной и супремум нормы. Для невырожденных матриц в качестве "специального" метода будем использовать метод Гаусса без частичного выбора главного элемента, а для трехдиагональных матриц - метод прогонки.

Приведем формулу (4) для нахождения относительной погрешности:

$$\delta = \frac{\|x_u - x_s\|}{\|x_u\|}, \quad (4)$$

где  $x_u$  - решение, полученное при помощи "универсального" метода,  $x_s$  - решение, полученное при помощи "специального" метода.

Для сравнения точности одного метода по отношению к другому использовались среднеквадратичная и супремум-нормы. Приведем их нахождение в формуле (5):

$$\begin{aligned} \|x\|_2 &= \sqrt{\sum_{i=1}^n x_i^2} \\ \|x\|_\infty &= \max_{i \in [1, \dots, n]} |x_i| \end{aligned} \quad (5)$$

В Листинге 4 представлено нахождение погрешности для метода Гаусса без частичного выбора главного элемента по отношению к аналогичному методу, но с выбором:

Листинг 4. Вычисление относительной погрешности

---

```

1 def gauss_error():
2     n = 6
3     count = 1000
4     square_gauss = np.empty(count, dtype=np.float32)
5     supremum_gauss = np.empty(count, dtype=np.float32)
6     for i in range(count):
7         A = matrix_generation(n)
8         A_copy1 = np.array(A)
9         A_copy2 = np.array(A)
10        x_gauss_element = gauss(A_copy1, [1, 1, 1, 1, 1, 1], True)
11        x_gauss = gauss(A_copy2, [1, 1, 1, 1, 1, 1], False)
12        square_gauss[i] = np.linalg.norm(x_gauss_element - x_gauss) /
            np.linalg.norm(x_gauss_element)

```

```

13     supremum_gauss[i] = np.linalg.norm((x_gauss_element - x_gauss), ord=np.inf) /
        np.linalg.norm(x_gauss_element,
14 plt.hist(square_gauss, np.linspace(0, 0.00001, 100), color='hotpink', edgecolor='black')
15 plt.xlabel('delta')
16 plt.title('Относительная погрешность для квадратичной нормы метод (Гаусса)')
17 plt.show()
18 plt.hist(supremum_gauss, np.linspace(0, 0.00001, 100), color='hotpink',
        edgecolor='black')
19 plt.xlabel('delta')
20 plt.title('Относительная погрешность для супремум нормы — метод (Гаусса)')
21 plt.show()

```

Для остальных методов относительная погрешность находится аналогично, поэтому их реализацию приводить не будем.

На рисунках 1, 2, 3, 4 осуществлен вывод относительной погрешности в виде гистограмм для двух методов и двух норм:

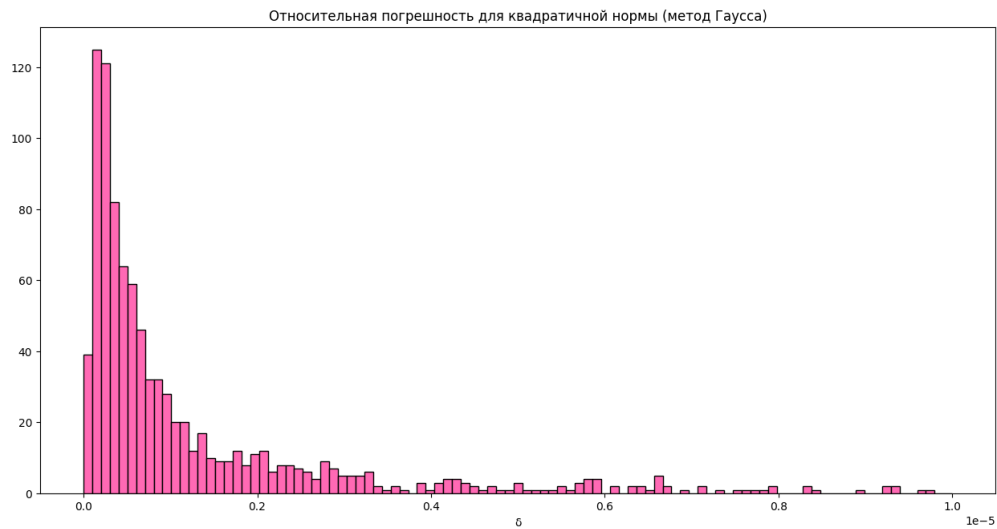


Рис. 1. График относительной погрешности метода Гаусса с частичным выбором главного элемента, вычисленной с помощью квадратичной нормы

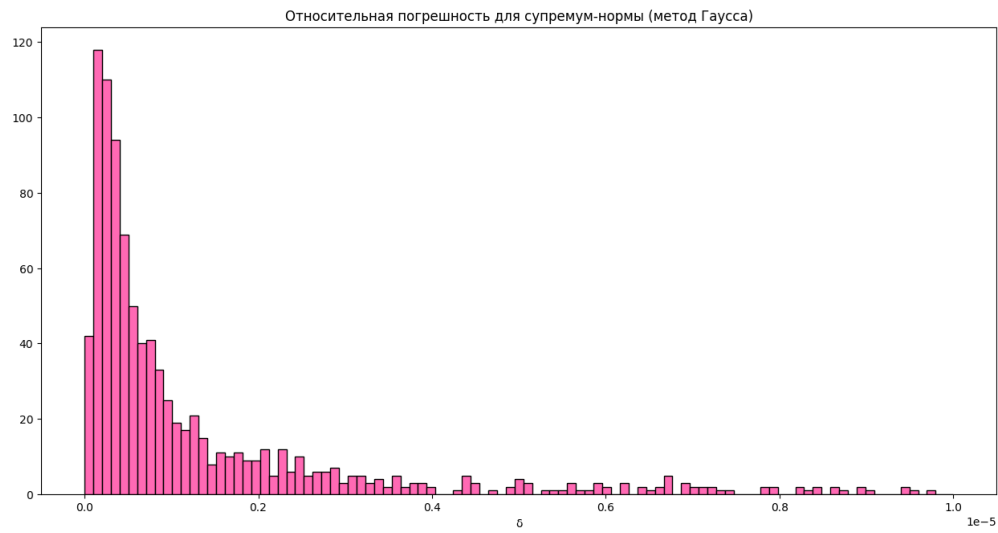


Рис. 2. График относительной погрешности метода Гаусса с частичным выбором главного элемента, вычисленной с помощью супремум-нормы

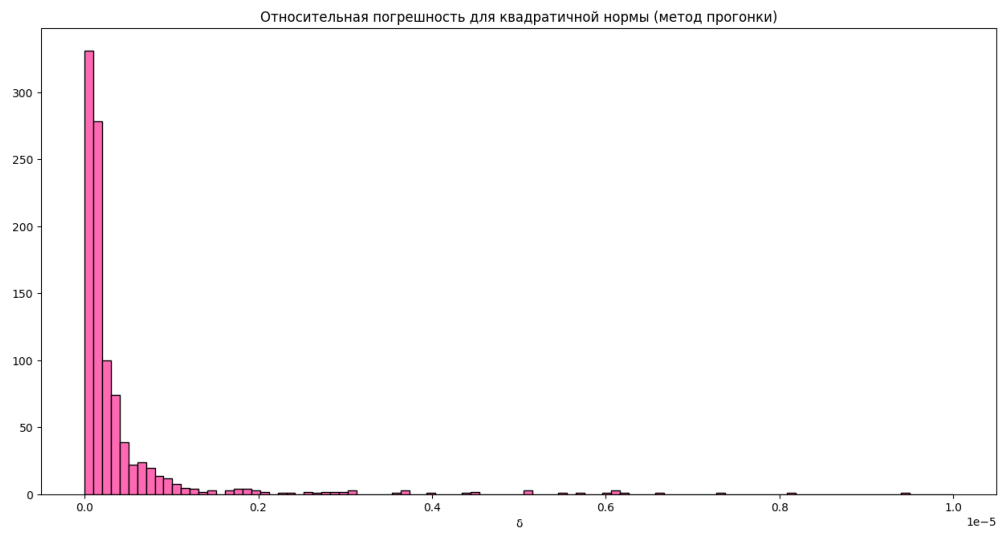


Рис. 3. График относительной погрешности метода прогонки, вычисленной с помощью квадратичной нормы

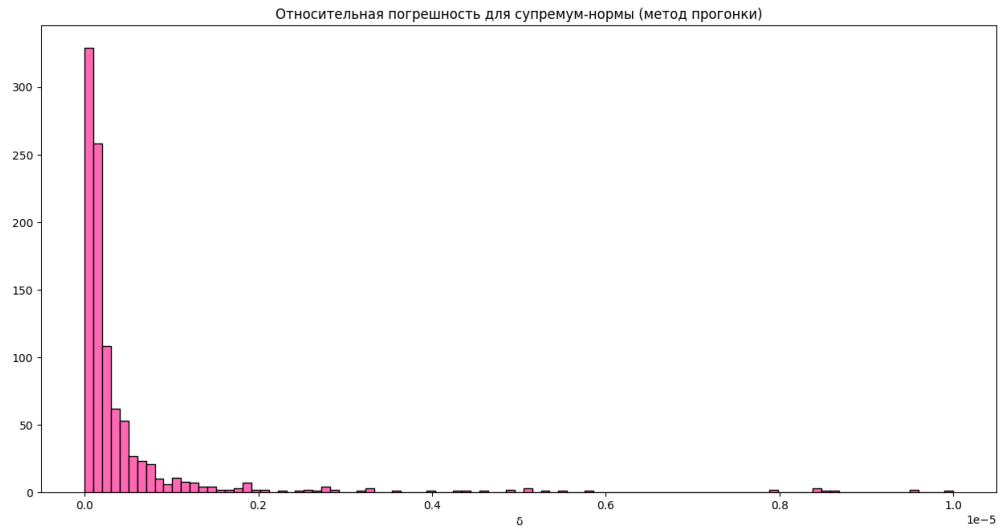


Рис. 4. График относительной погрешности метода прогонки, вычисленной с помощью супремум-нормы

По полученным графикам можем сделать вывод, что большинство значений относительной погрешности близки к нулю, что означает, что данные методы обладают высокой вычислительной устойчивостью. Сравнив два метода, можно увидеть, что метод прогонки дает меньшую погрешность, чем метод Гаусса без частичного выбора главного элемента.

## 2 Продвинутая часть

### Генерация положительно определённых матриц

Положительно определённая матрица - это такая матрица, собственные значения которой строго больше нуля. Матрица является положительно определённой только если присутствует разложение Холецкого вида  $A = LL^T$ . Чтобы получить положительно определённую матрицу, в работе применяются следующие шаги:

1. Создание нулевой матрицы  $n \times n$ .
2. Заполнение матрицы случайными значениями, оставляя элементы главной диагонали положительными.
3. Умножение полученной матрицы на её транспонированную версию.

Реализация функции генерации положительно определённых матриц представлена в листинге 5:

Листинг 5. Функция генерации положительно определённой матрицы

---

```

1 def positive_generation(n):
2     L = np.zeros((n, n), dtype=np.float32)
3     for i in range(n):
4         for j in range(i):
5             L[i][j] = np.random.uniform(-1.0001, 1)
6             L[i][i] = np.random.rand() # положительные элементы на диагонали
7     LT = np.transpose(L)
8     A = np.dot(L, LT)
9     print(np.all(np.linalg.eigvals(A) > 0)) # проверка на положительно определенность
10    return A

```

---

### Метод решения СЛАУ при помощи разложения Холецкого

Метод Холецкого применим к СЛАУ с положительно определённой матрицей коэффициентов. Разложение Холецкого - разложение вида  $A = LL^T$ , где  $L$  - нижнетреугольная матрица, не имеющая нулей на главной диагонали. Если матрица  $A$  положительно определённая и имеет вид:

$$\begin{bmatrix} a_{11} & \dots & a_{n1} \\ \vdots & \ddots & \vdots \\ a_{1n} & \dots & a_{nn} \end{bmatrix},$$

то элементы матрицы  $L$  будут вычисляться по следующим формулам:

$$l_{ii} = \sqrt{a_{ii} - \sum_{j=1}^{i-1} l_{ij}^2}, i \in [1, n],$$

$$l_{ij} = \frac{1}{l_{jj}}(a_{ij} - \sum_{k=1}^{j-1} l_{ik}l_{jk}), j < i$$

Содержание функции решения СЛАУ при помощи разложения Холецкого представлено в листинге 6:

Листинг 6. Решение СЛАУ при помощи разложения Холецкого

---

```

1 def cholesky(A, b):
2     L = np.zeros((n, n), dtype=np.float32)
3     for j in range(n):
4         L[j][j] = np.sqrt(A[j][j] - np.sum(L[j][:j] ** 2))
5         print(np.sum(L[j][:j] ** 2))
6         for i in range(j + 1, n):
7             L[i][j] = 1 / L[j][j] * (A[i][j] - np.sum(L[i][:j] * L[j][:j]))
8     print(L)
9     LT = np.transpose(L)
10    solution = np.zeros(n, dtype=np.float32)
11    solution1 = np.zeros(n, dtype=np.float32)
12    for i in range(n):
13        s = np.float32(0.0)
14        for j in range(i):
15            s += L[i][j] * solution[j]
16        solution[i] = (b[i] - s) / L[i][i]
17    for i in range(n - 1, -1, -1):
18        s = np.float32(0.0)
19        for j in range(n - 1, i, -1):
20            s += LT[i][j] * solution1[j]
21        solution1[i] = (solution[i] - s) / LT[i][i]
22    return solution1

```

---

## Проведение анализа метода Холецкого

Метод Холецкого является "специальным" методом для положительно определённых матриц. Было проведено сравнение с определённым ранее "универсальным" методом. Для того чтобы найти распределение погрешностей были использованы аналогичные нормы представленные в базовой части. Погрешности, вычисленные при помощи среднеквадратичной и супремум норм, отображены на рисунках 5 и 6. Полученные погрешности располагаются крайне близко к нулю, а значит метод Холецкого устойчив для положительно определённых матриц.

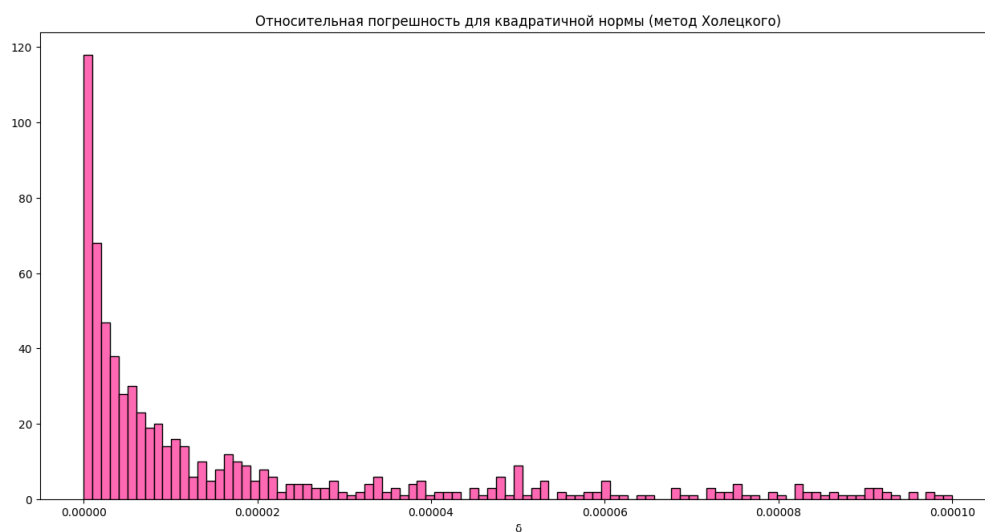


Рис. 5. График относительной погрешности метода Холецкого, вычисленной с помощью квадратичной нормы

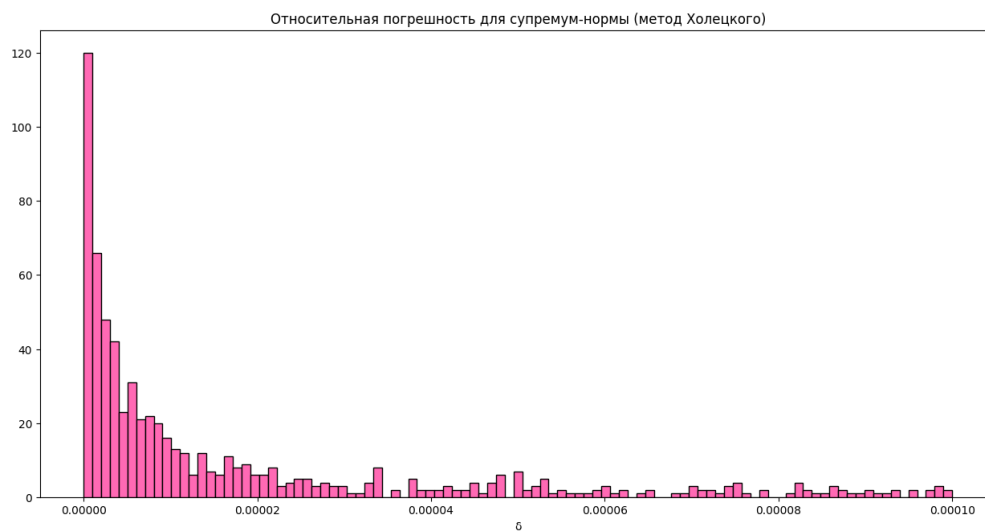


Рис. 6. График относительной погрешности метода Холецкого, вычисленной с помощью супремум-нормы

Из графиков можно сделать вывод, что метод Холецкого обладает наименьшей устойчивостью из трех рассматриваемых методов.

### Нахождение распределения для спектральных радиусов

Спектральный радиус матрицы - это максимальное значение среди модулей всех собственных значений данной матрицы. Собственные значения матрицы находились при помощи встроенной функции `numpy.linalg.eigvals()`. На рисунках 7, 8 и 9 показано распределение спектральных радиусов для произвольных, трёхдиагональных и положительно определённых матриц соответственно:

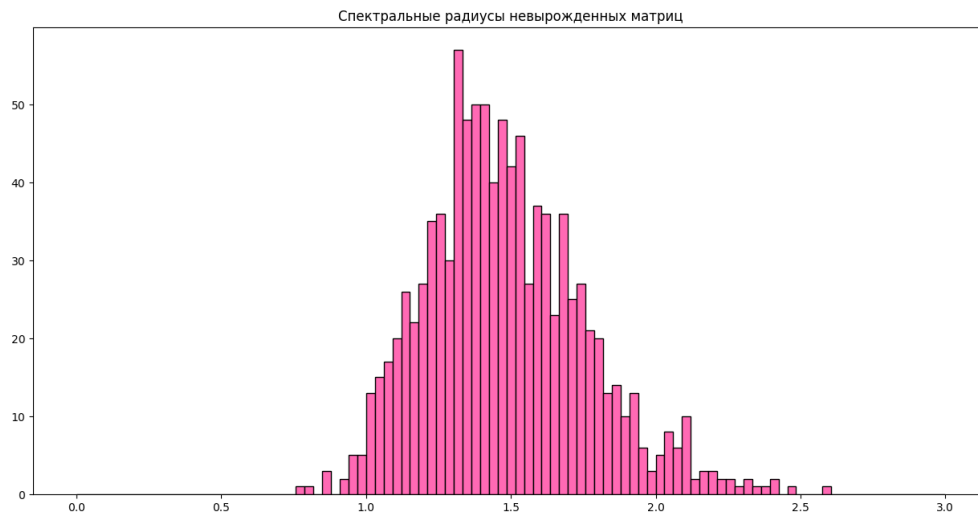


Рис. 7. Распределение спектральных радиусов для невырожденных произвольных матриц

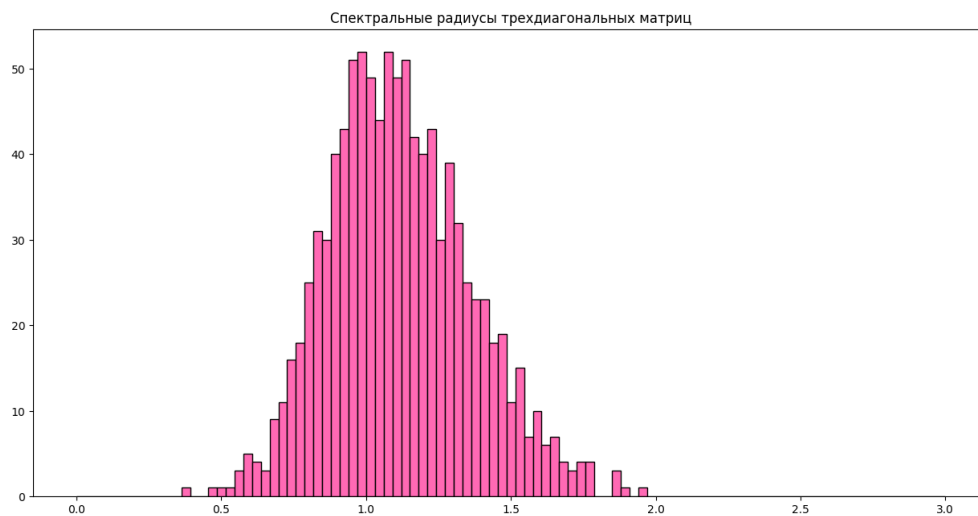


Рис. 8. Распределение спектральных радиусов для трёхдиагональных матриц



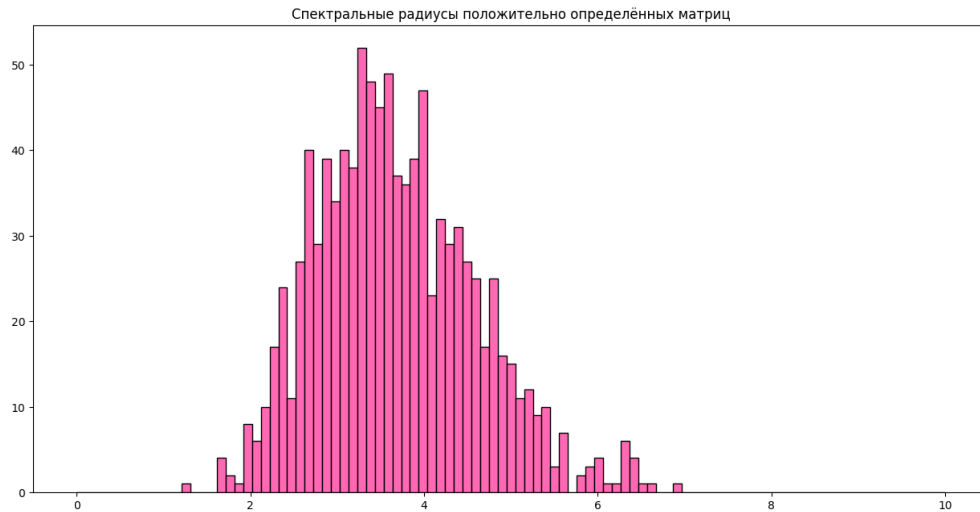


Рис. 9. Распределение спектральных радиусов для положительно определённых матриц

### Нахождение распределения для чисел обусловленности

Обусловленность - это качественная характеристика матрицы. Число обусловленности является показателем устойчивости решения СЛАУ к ее малым изменениям. Чем меньше число обусловленности  $K(A)$ , тем будет меньше относительная погрешность вычислений.

Число обусловленности находится следующим образом:

$$K(A) = \|A\| * \|A^{-1}\|$$

В работе числа обусловленности находились при помощи функции `numpy.linalg.cond()`. На рисунках 10, 11 и 12 показано распределение чисел обусловленности для произвольных, трёхдиагональных и положительно определённых матриц соответственно:

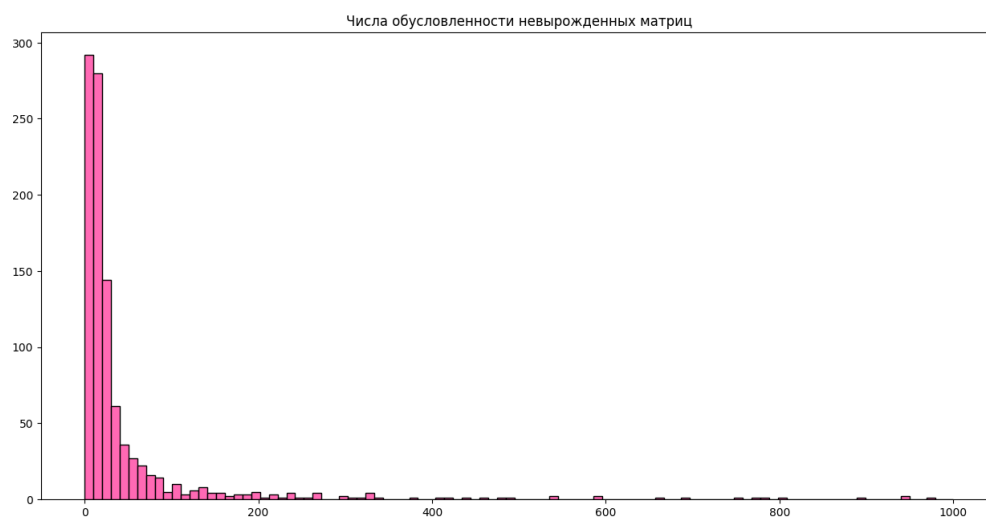


Рис. 10. Распределение чисел обусловленности для произвольных матриц

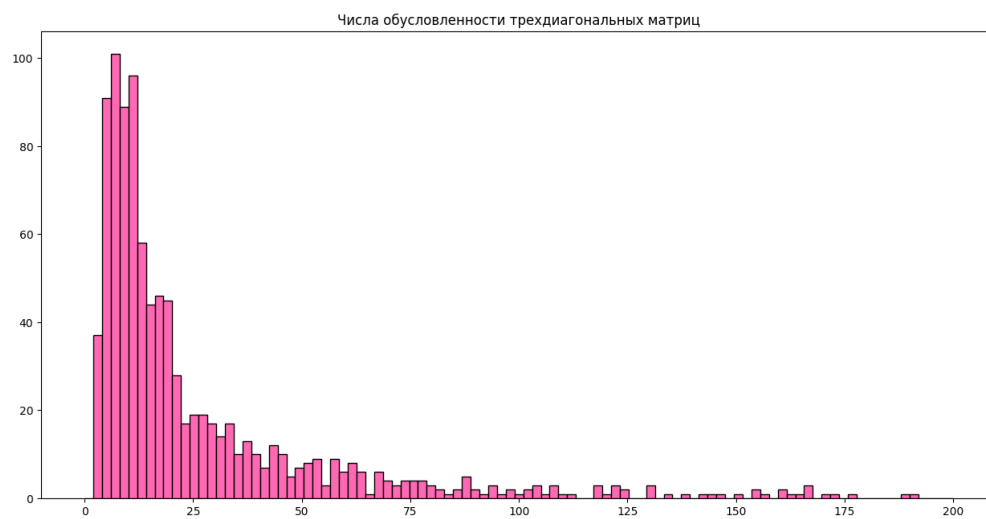


Рис. 11. Распределение чисел обусловленности для трёхдиагональных матриц

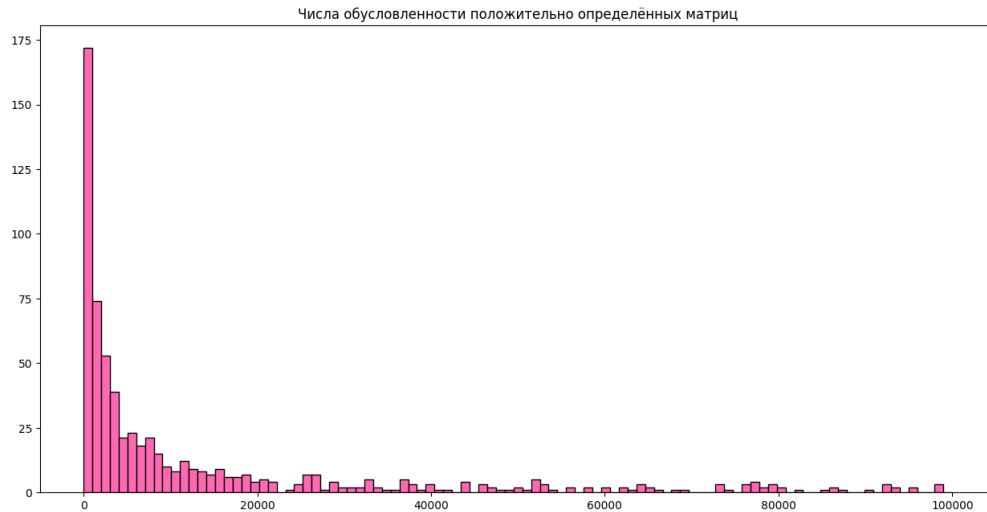


Рис. 12. Распределение чисел обусловленности для положительно определённых матриц

На графиках видно, что самые маленькие числа обусловленности у трехдиагональных матриц. Также в предыдущих пунктах был сделан вывод о том, что решение СЛАУ методом прогонки для трехдиагональных матриц обладает наименьшей относительной погрешностью (см. рисунки 3 и 4). Таким образом доказывается утверждение о том, что чем меньше число обусловленности матрицы, тем меньше погрешность при решении СЛАУ.

#### **Оценка влияния спектрального радиуса на вычислительную устойчивость**

Спектральный радиус не влияет напрямую на вычислительную устойчивость прямых методов решения СЛАУ. Но так как он связан с числом обусловленности, которое влияет на устойчивость, большое значение может негативно повлиять на погрешности.

#### **Оценка влияния отношения максимального и минимального собственного числа на вычислительную устойчивость**

Отношение максимального и минимального собственного числа непосредственно влияет на вычислительную устойчивость. Данное отношение показывает насколько близка матрица к вырожденной. Соответственно, чем ближе она к вырожденной, тем большее значение имеет это отношение. Близость матрицы к вырожденной отрицательно сказывается на устойчивости метода.

#### **Оценка влияния отношения числа обусловленности на вычислительную устойчивость**

**Теорема** Пусть  $x \in \mathbb{R}^n$  является приближением к точному решению СЛАУ  $Ax = b$ ,

$A$  – невырожденная матрица и  $r$  – вектор невязки. Тогда для любой индуцированной матричной нормы верно:

$$\|x - \tilde{x}\| \leq \|r\| \cdot \|A^{-1}\|, \quad (6)$$

и при  $x, b \neq 0$ :

$$\frac{\|x - \tilde{x}\|}{\|x\|} \leq K(A) \frac{\|r\|}{\|b\|}, \quad (7)$$

где  $K(A) = \|A\| \cdot \|A^{-1}\|$  – число обусловленности.

Из данного неравенства видно, что число обусловленности напрямую влияет на относительную погрешность. Чем больше число обусловленности, тем выше накопленные погрешности.

## Заключение



В ходе выполнения лабораторной работы были изучены прямые методы решения систем линейных алгебраических уравнений (СЛАУ), такие как метод Гаусса, метод прогонки и метод Холецкого. Был проведен анализ применимости каждого из этих методов к определенным типам матриц, а также изучены относительные погрешности, возникающие при использовании этих методов для решения СЛАУ. Кроме того, были рассмотрены спектральный радиус и число обусловленности, и описано их влияние на вычислительную устойчивость методов решения систем линейных уравнений.

## Список использованных источников

1. Першин А.Ю. Лекции по курсу «Вычислительная математика». Москва, 2018-2021. С. 140. URL: <https://archrk6.bmstu.ru/index.php/f/810046>.
2. Соколов, А.П. Инструкция по выполнению лабораторных работ (общая). Москва: Соколов, А.П., 2018-2021. С. 9. URL: <https://archrk6.bmstu.ru>. (облачный сервис кафедры РК6).
3. Соколов, А.П. Инструкция по выполнению заданий к семинарским занятиям (общая). Москва: Соколов, А.П., 2018-2022. С. 7. URL: <https://archrk6.bmstu.ru>. (облачный сервис кафедры РК6).
4. Першин А.Ю. Сборник задач семинарских занятий по курсу «Вычислительная математика»: Учебное пособие. / Под редакцией Соколова А.П. [Электронный ресурс]. Москва, 2018-2021. С. 20. URL: <https://archrk6.bmstu.ru>. (облачный сервис кафедры РК6).
5. Першин А.Ю., Соколов А.П. Сборник постановок задач на лабораторные работы по курсу «Вычислительная математика»: Учебное пособие. [Электронный ресурс]. Москва, 2021. С. 54. URL: <https://archrk6.bmstu.ru>. (облачный сервис кафедры РК6).

## Выходные данные

Киммель А.А. Отчет о выполнении лабораторной работы №4 по дисциплине «Вычислительная математика». [Электронный ресурс] — Москва: 2023. — 21 с. URL: <https://sa2systems.ru:88> (система контроля версий кафедры РК6)

Постановка:  доцент кафедры РК-6, PhD А.Ю. Першин  
Решение и вёрстка:  студент группы РК6-55Б, Киммель А.А.

2023, осенний семестр