



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 по дисциплине «Вычислительная математика»

| | |
|--------------|--|
| Студент: | Гапеев Дмитрий Андреевич |
| Группа: | РК6-51Б |
| Тип задания: | Лабораторная работа №4 |
| Тема: | Устойчивость прямых методов решения СЛАУ |

Студент

подпись, дата

Гапеев Д.А.

Фамилия, И.О.

Преподаватель

подпись, дата

Фамилия, И.О.

Москва, 2023

Содержание

| | |
|---|----------|
| Устойчивость прямых методов решения СЛАУ | 3 |
| Задание | 3 |
| Цель выполнения лабораторной работы | 4 |
| 1 Методы Гаусса и метод прогонки | 4 |
| 1.1 Реализация метода Гаусса | 4 |
| 1.2 Реализация метода прогонки | 6 |
| 1.3 Выбор “универсального” метода для матриц общего вида. | 7 |
| 1.4 Разработка алгоритма генерации случайных матриц | 8 |
| 1.5 Проведение эксперимента для матриц общего вида и трехдиагональ- | |
| ных матриц | 10 |
| 2 Разложение Холецкого и расширенный анализ вычислительной устойчи- | |
| вости реализованных методов | 15 |
| 2.1 Расширения генератора случайных матриц для получения положи- | |
| тельно определенных матриц | 15 |
| 2.2 Метод решения СЛАУ, полученного при помощи разложения Холецкого | 15 |
| 2.3 Анализ метода Холецкого | 17 |
| 2.4 Визуализация распределения спектральных радиусов и распределе- | |
| ния чисел обусловленности | 19 |
| 2.5 Влияние спектральных радиусов матриц на вычислительную устой- | |
| чивость алгоритмов | 24 |
| 2.6 Влияние отношения максимального по модулю собственного числа к | |
| минимальному по модулю собственному числу матриц на вычис- | |
| лительную устойчивость алгоритмов | 24 |
| 2.7 Влияние числа обусловленности матриц на вычислительную устой- | |
| чивость алгоритмов | 26 |
| Заключение | 27 |

Устойчивость прямых методов решения СЛАУ

Задание

Системы линейных алгебраических уравнений неизбежно появляются как промежуточный или конечный этап в нахождении численных решений в ряде методов 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 мы рассмотрим несколько видов матриц и с помощью генерации большого количества случайных матриц продемонстрируем 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 6×6 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-битным float-представлением элементов необходимо провести следующий эксперимент:
 - (a) Выбрать “специальный” вычислительно-эффективный метод по типу матрицы.
 - (b) Для каждой СЛАУ $A^{(j)}x = [1, 1, 1, 1, 1, 1]^T$ 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 значение отношения 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 выполнения лабораторной работы - изучение прямых методов решения СЛАУ. Эти методы включают: метод Гаусса с частичным выбором главного элемента, метод Гаусса без выбора главного элемента, метод прогонки и метод, решающий СЛАУ при помощи разложения Холецкого. Исследование вычислительной устойчивости метода Гаусса, метода прогонки и разложения Холецкого на примере нахождения решений СЛАУ с различными матрицами коэффициентов.

1 Методы Гаусса и метод прогонки

1.1 Реализация метода Гаусса

Метод Гаусса состоит в приведении матрицы коэффициентов A к верхнетреугольному виду с помощью элементарных преобразований, и последовательного решения полученных уравнений. Первый этап называется прямым ходом Гаусса и заключается в последовательном обнулении элементов, находящихся под главной диагональю, которое реализовано с помощью применения к расширенной матрице \tilde{A} элементарных преобразований вида[1]:

$$a_{ij}^{(1)} = a_{ij} - \frac{a_{i1}}{a_{11}} a_{1j} \quad (1)$$

$$b_i^{(1)} = b_i - \frac{a_{i1}}{a_{11}} b_1 \quad (2)$$

Реализация прямого хода подразумевает вычисление множителя $m_{ik} = \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}}$, последовательное вычисление коэффициентов в каждой строке на k -й итерации по формуле 1 и вычисление новых значений коэффициентов вектора b по формуле 2.

После приведения матрицы к треугольному виду решения находятся с помощью обратного хода метода Гаусса:

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

Если диагональный элемент равен нулю, вычисление множителя m_{ik} не представляется возможным. Помимо этого, диагональный элемент может быть крайне мал и может привести к вычислительной погрешности, которая в данном рассматриваемом методе на каждой итерации будет накапливаться. Более того, деление на малый элемент $a_{kk}^{(k)}$ происходит также во время обратного хода Гаусса и усиливает накопленные погрешности в числителе. Во избежание подобных проблем была реализована модификация метода Гаусса с частичным выбором главного элемента. Данный выбор подразумевает перестановку строк матрицы так, что диагональным элементом становится наибольший по модулю элемент k -го столбца следующим образом:

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

По итогу была реализована функция `gauss(A, b, pivoting)` (листинг 1) для решения СЛАУ вида $Ax = b$.

Листинг 1. Функция для решения СЛАУ методом Гаусса

```

1 def gauss(A, b, pivoting=True):
2     n = len(A)
3     if np.abs(np.linalg.det(A)) < epsilon: # Проверка на сингулярность
4         raise ValueError("No solution")
5     A_new = np.copy(A)
6     b_new = np.copy(b)
7     for k in range(n - 1):
8         if pivoting:
9             # Реализация метода Гаусса с частичным выбором главного элемента
10            index_max_row = abs(A_new[k:, k]).argmax() + k # Индекс строки
11            # Перестановка для метода Гаусса с частичным выбором главного элемента
12            if index_max_row != k: # Перестановка для строк
13                A_new[[k, index_max_row]] = A_new[[index_max_row, k]]
14                b_new[[k, index_max_row]] = b_new[[index_max_row, k]]
15            else:
16                if A_new[k, k] == 0: # Проверка на равенство нулю
17                    raise ValueError("Element on general diagonal is zero")
18            for row in range(k + 1, n):
19                coeff = A_new[row, k] / A_new[k, k]
20                A_new[row, k:] = A_new[row, k:] - coeff * A_new[k, k:]
21                b_new[row] = b_new[row] - coeff * b_new[k]
22            # Обратный ход
23            x = np.zeros((n, 1))
24            for k in range(n - 1, -1, -1):
25                x[k] = (b_new[k] - np.dot(A_new[k, k + 1:], x[k + 1:])) / A_new[k, k]
26            return x

```

1.2 Реализация метода прогонки

Метод прогонки является методом решения СЛАУ для трехдиагональных матриц, так как использует свойства данных матриц и уменьшает количество арифметических операций до $O(n)$ по сравнению с методом Гаусса, имеющим вычислительную сложность $O(n^3)$. Также если исходная матрица обладает свойством строгого диагонального преобладания, то метод прогонки является вычислительно устойчивым.

Метод прогонки требует вычисление коэффициентов γ_i и β_i при помощи следующих рекуррентных соотношений :

$$\gamma_{i+1} = -\frac{a_{i,i+1}}{a_{i,i-1}\gamma_i - a_{ii}}, \quad \beta_{i+1} = \frac{b_i - a_{i,i-1}\beta_i}{a_{i,i-1}\gamma_i + a_{ii}} \quad (3)$$

При чем $\gamma_1 = \beta_1 = 0$, ввиду особенностей трехдиагональных матриц. После последовательного вычисления коэффициентов γ_i и β_i (формула 3), вычисляются компоненты вектора решения СЛАУ следующим способом:

$$x_{i-1} = \gamma_i x_i + \beta_i \quad (4)$$

с начальным условием $x_n = \frac{b_n - a_{n,n-1}\beta_n}{a_{nn} + a_{n,n-1}\gamma_n}$.

По итогу была реализована функция `thomas(A, b)` (листинг 2) для решения СЛАУ вида $Ax = b$, в реализации которой используются формулы 3 и 4.

Листинг 2. Функция для решения СЛАУ методом прогонки

```

1 def thomas(A, b):
2     n = len(A)
3     gamma = np.zeros((n, 1)) # Коэффициенты гамма
4     beta = np.zeros((n, 1)) # Коэффициенты бета
5     for k in range(n - 1):
6         if k != 0:
7             A_k_previous = A[k, k - 1]
8         else:
9             A_k_previous = 0
10        # Расчет коэффициентов по средству рекуррентных соотношений
11        # Прямой проход
12        gamma[k + 1] = - A[k, k + 1] / (A_k_previous * gamma[k] + A[k, k])
13        beta[k + 1] = (b[k] - A_k_previous * beta[k]) / (A_k_previous * gamma[k] +
14        A[k, k])
15        x = np.zeros((n, 1))
16        # Начальное условие для обратного прохода
17        x[n - 1] = (b[n - 1] - A[n - 1, n - 2] * beta[n - 1]) / (A[n - 1, n - 1] + A[n - 1, n
18        - 2] * gamma[n - 1])
19        # Обратный проход
20        for i in range(n - 1, 0, -1):
21            x[i - 1] = gamma[i] * x[i] + beta[i]
22    return x

```

1.3 Выбор “универсального” метода для для матриц общего вида.

Были протестированы функции, реализующие методы для решения СЛАУ $Ax = b$. В качестве тестовых данных взяты следующие значения:

$$b = [3.52971, 0.333, 1.6666]^T$$

$$A = \begin{bmatrix} 1 & 2 & 0 \\ 2 & 6 & 5 \\ 0 & 5 & 13 \end{bmatrix}$$

Решения полученные функциями `gauss(A, b, pivoting)` и `thomas(A, b)` сравнивались с решением полученным при помощи метода `numpy.linalg.solve`. Исходя из результатов тестирования, продемонстрированных на рис. 1, сделан вывод что функции реализованы верно.

```

Gauss without pivoting:
[[195.08263]
 [-95.77646]
 [ 36.9653 ]]
Gauss with pivoting:
[[195.08263]
 [-95.77646]
 [ 36.9653 ]]
Thomas:
[[195.08263]
 [-95.77646]
 [ 36.9653 ]]
Answer:
[195.08263 -95.77646  36.9653 ]

```

Рис. 1. Результаты тестирования функций `gauss(A, b, pivoting)` и `thomas(A, b)`

Среди рассматриваемых методов, выбор “универсального” метода для матриц общего вида возможен только из метода Гаусса и метода Гаусса с частичным выбором главного элемента. Данные методы работают с матрицами общего вида, в то время как метод прогонки только для трехдиагональных матриц. Из возможных методов наиболее точным является метод Гаусса с частичным выбором главного элемента, поскольку по определению уменьшает накапливаемую погрешность вычисления, которая может присутствовать в методе Гаусса без выбора главного элемента. Таким образом, элементарные преобразования, присутствующие в методе Гаусса в реализации прямого хода являются более точными.

Таким образом, в качестве “универсального” метода для матриц общего вида был выбран метод Гаусса с частичным выбором главного элемента.

1.4 Разработка алгоритма генерации случайных матриц

Для генерации случайных матриц, принадлежащих определенному классу, была реализована функция `generate_random_matrix(n, matrix_type)`, где `n` - размерность генерируемой матрицы, `matrix_type` - тип генерируемой невырожденной матрицы, который может принимать следующие значения:

1. `default` - матрица общего вида;
2. `TD` - трехдиагональная матрица.

Генерация матрицы общего вида подразумевает генерацию случайного числа для каждого коэффициента матрицы в интервале $(-1; 1)$. Затем полученная матрица преобразуется в 32-битные числа с плавающей запятой. Далее происходит проверка является ли матрица близкой к сингулярной (определитель близок к нулю), то есть происходит сравнение с малой величиной $\epsilon = 0.5$. Генерация матрицы общего вида происходит до тех пор, пока сгенерированная матрица не пройдет проверку на сингулярность. Проверка на близость полученной матрицы к сингулярной необходима для обеспечения корректной работы методов решения СЛАУ, поскольку в ином случае могут возникать численные ошибки при выполнении метода Гаусса.

Трехдиагональная матрица - это особый тип квадратной матрицы, у которой ненулевые элементы расположены только на главной диагонали, а также на диагоналях непосредственно над и под главной диагональю. Все остальные элементы такой матрицы равны нулю. Генерация трехдиагональной матрицы подразумевает собой генерацию случайных чисел в интервале $(-1; 1)$ для трех диагоналей и последующее преобразование в 32-битные числа с плавающей запятой. Далее происходит аналогичная проверка на близость полученной матрицы к сингулярной. В случае несоответствия сгенерированной матрицы условию, происходит увеличение диагональных элементов. Сгенерированная матрица должна иметь значения коэффициентов $|a_{ij}| < 1$, тогда увеличение диагональных элементов может привести к нарушению ограничения на элементы матрицы. Поэтому, далее происходит деление всей матрицы на скаляр, равный максимальному элементу матрицы. В результате большинство сгенерированных трехдиагональных матриц обладают строгим диагональным преобладанием.

Реализация алгоритма генерации случайных матриц представлена в функции `generate_random_matrix(n, matrix_type)` (листинг 3).

Листинг 3. Функция для генерации случайных матриц общего и трехдиагонального видов

```

1 def generate_random_matrix(n, matrix_type="default"):
2     #Генерация матриц общего вида
3     gen_matrix = np.random.rand(n, n).astype(np.float32) * 2 - 1 # Генерация матрицы в
        интервале [-1, 1]
4     while np.abs(np.linalg.det(gen_matrix)) < epsilon:
5         gen_matrix = np.random.rand(n, n).astype(np.float32) * 2 - 1 # Генерация
        матрицы в интервале [-1, 1]
6     #Генерация трехдиагональных матриц
7     if matrix_type == "TD":
8         diagonal = np.random.uniform(-1, 1, n).astype(np.float32)
9         off_diagonal = np.random.uniform(-1, 1, n - 1).astype(np.float32)
10        gen_matrix = np.diag(diagonal) + \
11            np.diag(off_diagonal, k=1) + \
12            np.diag(off_diagonal, k=-1)
13        while np.abs(np.linalg.det(gen_matrix)) < epsilon:
14            gen_matrix += np.eye(n) * 0.1
15            max_elem = np.max(np.abs(gen_matrix))
16            if max_elem >= 1:
17                gen_matrix /= (max_elem + 0.001)

```

1.5 Проведение эксперимента для матриц общего вида и трехдиагональных матриц

Матрицы общего вида.

(а) В качестве “специального” вычислительно-эффективного метода был выбран метод Гаусса без выбора главного элемента. Так как для решения СЛАУ с матрицами общего вида подходят метод Гаусса и метод Гаусса с частичным выбором главного элемента, при этом частичный выбор главного элемента увеличивает количество арифметико-логических операций, значит метод Гаусса без выбора главного элемента более подходит под понятие “специальный”.

(б) Для анализа решений СЛАУ для матриц общего вида, а также для анализа характеристик таких матриц, с помощью функции `generate_random_matrix(n=6, matrix_type = default)` были сгенерированы 1000 случайных матриц $A^{(j)}$ общего вида. Затем найдены решения СЛАУ при помощи метода Гаусса с частичным выбором главного элемента (“универсальный”) и метода Гаусса без выбора главного элемента (“специальный”). СЛАУ имеет следующий вид:

$$Ax = b,$$

где $b = [1, 1, 1, 1, 1, 1]^T$. Относительная погрешность находилась при помощи среднеквадратичной и супремум-нормы. Среднеквадратичная и супремум нормы имеют соответственно формулы 4 и 5:

$$\|x\|_2 = \left(\sum_{i=1}^n x_i^2 \right)^{\frac{1}{2}}; \quad (4)$$

$$\|x\|_\infty = \max_{i \in [1;n]} |x_i|, \quad (5)$$

где $x = [x_1, \dots, x_n] \in \mathbb{R}^n$. Следовательно формула для вычисления относительной погрешности с помощью среднеквадратичной нормы (формула 4) выглядит следующим образом:

$$\frac{\|x - \tilde{x}\|_2}{\|x\|_2} = \sqrt{\frac{\sum_{i=1}^n (x_i - \tilde{x}_i)^2}{\sum_{i=1}^n x_i^2}}, \quad (6)$$

где x - решение, полученное при помощи “универсального” метода, а \tilde{x} - решение, полученное при помощи “специального” метода. Реализация алгоритма вычисления относительной нормы при помощи 6 представлена в функции `count_rms_error(accurate, approximate)` (листинг 4).

Листинг 4. Функция для вычисления относительной погрешности при помощи среднеквадратичной нормы

```

1 def count_rms_error(accurate, approximate):
2     #Вычисление относительной погрешности с помощью среднеквадратичной нормы
3     n = len(accurate)
4     absolute_error = np.sum([(accurate[i] - approximate[i]) ** 2 for i in range(n)])
5     normal = np.sum([(accurate[i] ** 2) for i in range(n)])
6     return np.sqrt(absolute_error) / np.sqrt(normal)

```

Формула для вычисления относительной погрешности с помощью супремум-нормы, исходя из формулы 5 выглядит следующим образом:

$$\frac{\|\mathbf{x} - \tilde{\mathbf{x}}\|_{\infty}}{\|\mathbf{x}\|_{\infty}} = \frac{\max_{i \in [1;n]} |x_i - \tilde{x}_i|}{\max_{i \in [1;n]} |x_i|}. \quad (7)$$

Реализация алгоритма вычисления относительной нормы при помощи 7 представлена в функции `count_supremum_error(accurate, approximate)` (листинг 5).

Листинг 5. Функция для вычисления относительной погрешности при помощи супремум-нормы

```

1 def count_supremum_error(accurate, approximate):
2     #Вычисление относительной погрешности с помощью супремум-нормы
3     return np.max(np.abs(accurate - approximate)) / np.max(np.abs(accurate))

```

Результаты визуализации распределения погрешностей в виде гистограмм продемонстрированы на рис. 2 и рис. 3.

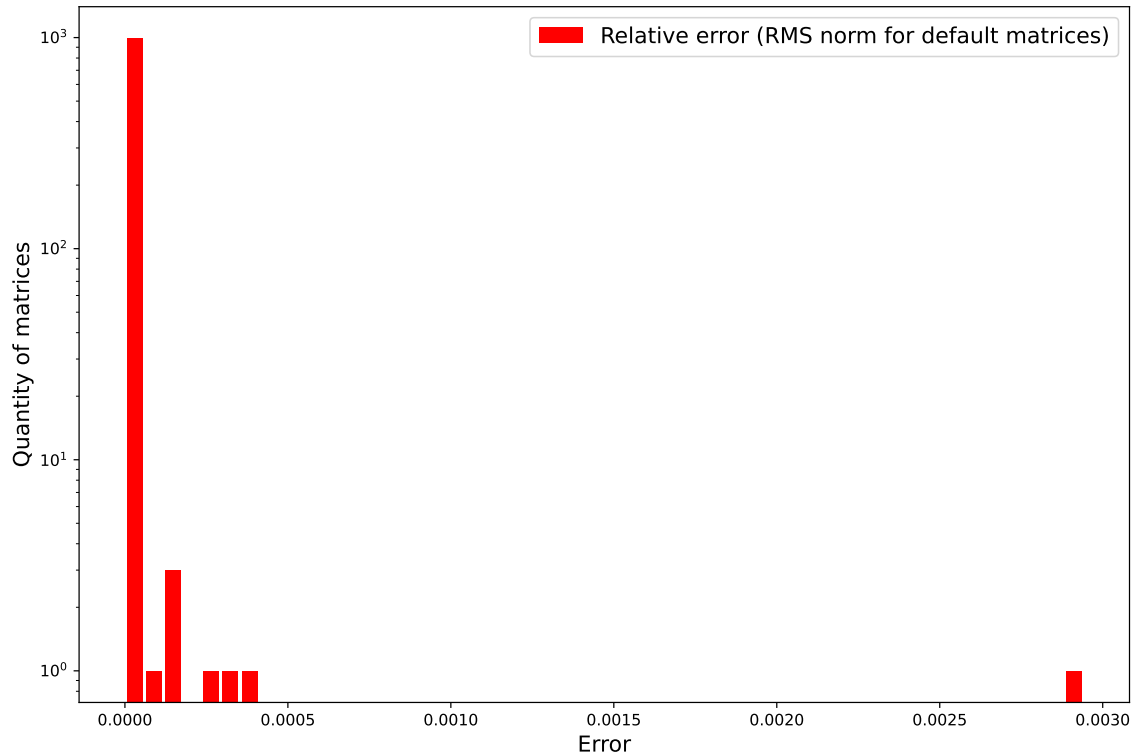


Рис. 2. Распределение относительных погрешностей, полученных с помощью среднеквадратичной нормы, для случайно сгенерированных матриц общего вида

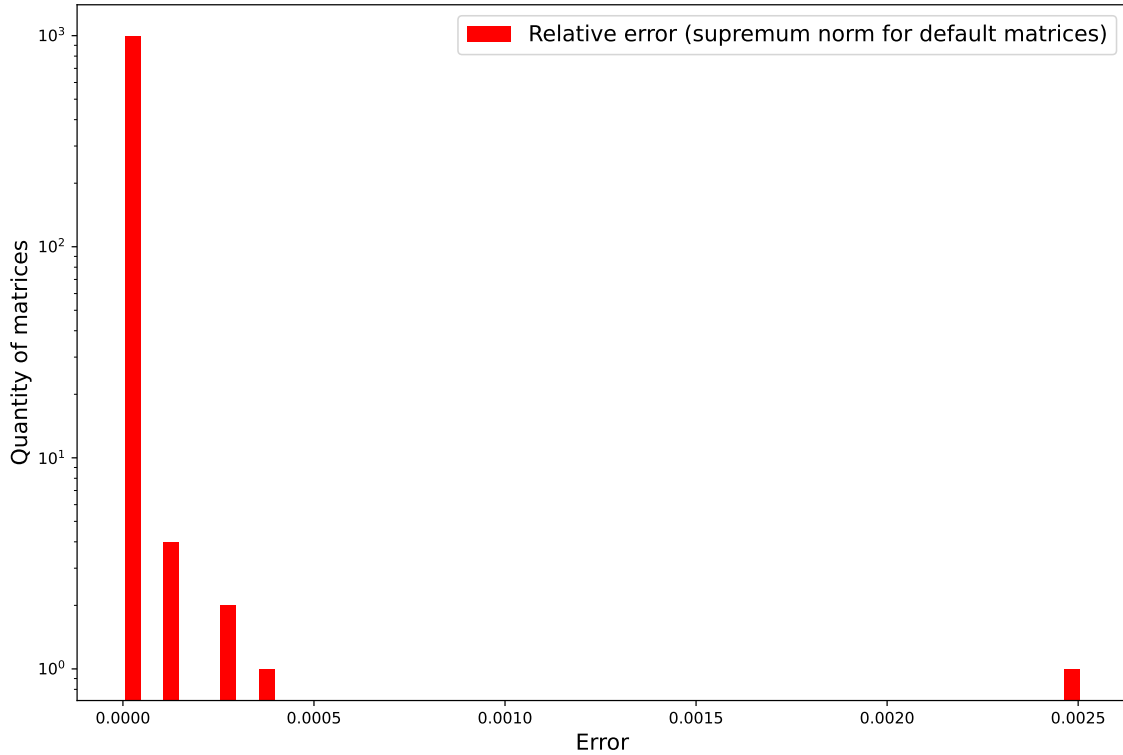


Рис. 3. Распределение относительных погрешностей, полученных с помощью супремум нормы, для случайно сгенерированных матриц общего вида

(с) Метод Гаусса без выбора главного элемента для матриц общего вида можно считать вычислительно неустойчивым, так как в ходе элементарных преобразований может возникнуть деление на очень малое число, которое увеличит ошибки округления и приведет к накоплению вычислительной погрешности, особенно для плохо обусловленных матриц. В этом случае, для некоторых матриц, в ходе элементарных преобразований которых возникло подобное деление, вычислительная погрешность будет больше, чем для остальных, что демонстрируется на [рис. 2](#) и [рис. 3](#). Большая часть относительных погрешностей сконцентрировано в левой части, что свидетельствует о том, что для некоторых матриц значение погрешностей значительно выше, чем для большинства.

Трехдиагональные матрицы.

(а) В качестве “специального” вычислительно-эффективного метода для трехдиагональных матриц был выбран метод прогонки. Для решения СЛАУ с трехдиагональными матрицами подходят метод Гаусса, метод Гаусса с частичным выбором главного элемента и метод прогонки, при этом метод прогонки обладает вычислительной сложностью $O(n)$, а метод Гаусса обладает сложностью $O(n^3)$. Исходя из этих соображений

метод прогонки обладает минимальным количеством арифметико-логических операций для решения СЛАУ с трехдиагональными матрицами.

(b) Для анализа решений СЛАУ с трехдиагональными матрицами, а также для анализа характеристик таких матриц, с помощью функции `generate_random_matrix(n=6, matrix_type = TD)` были сгенерированы 1000 случайных матриц $A^{(j)}$. Затем найдены решения СЛАУ при помощи метода Гаусса с частичным выбором главного элемента (“универсальный”) и метода прогонки (“специальный”). Нахождение относительных погрешностей вычислений выполнено при помощи ранее реализованных функций `count_rms_error(accurate, approximate)` (листинг 4) и `count_supremum_error(accurate, approximate)` (листинг 5). Результаты визуализации распределения погрешностей в виде гистограмм продемонстрированы на [рис. 4](#) и [рис. 5](#).

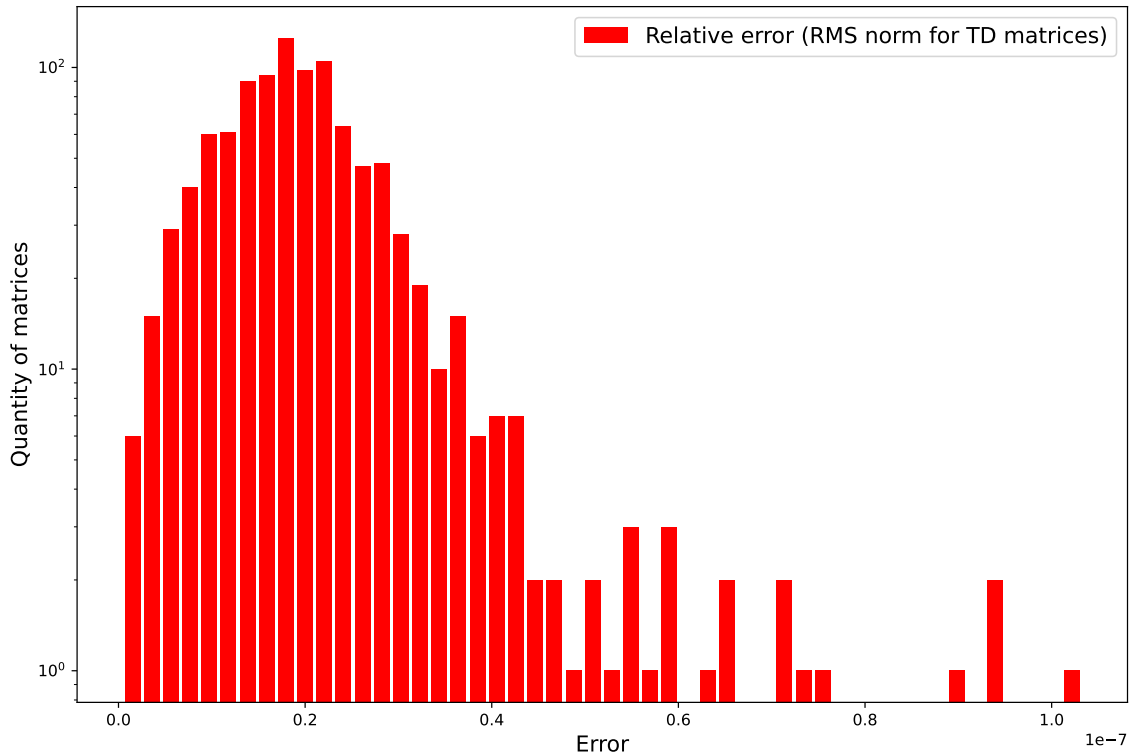


Рис. 4. Распределение относительных погрешностей, полученных с помощью среднеквадратичной нормы, для случайно сгенерированных трехдиагональных матриц

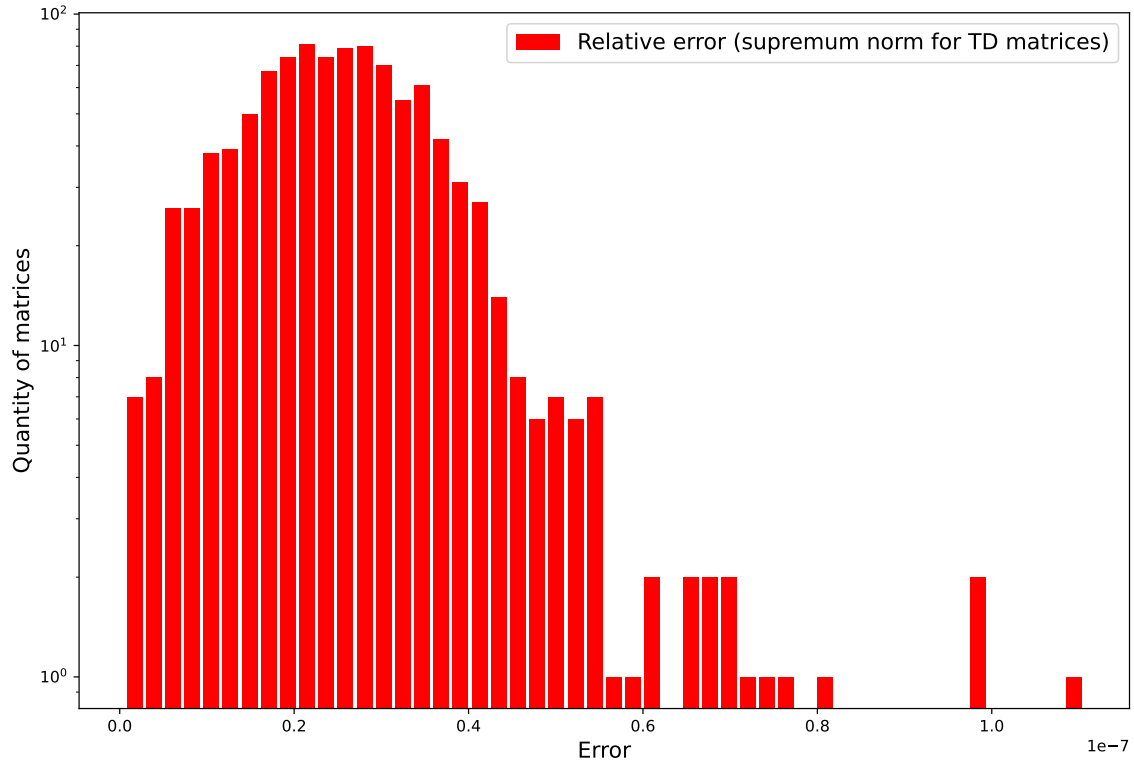


Рис. 5. Распределение относительных погрешностей, полученных с помощью супремум нормы, для случайно сгенерированных трехдиагональных матриц

(с) Метод прогонки можно считать вычислительно устойчивым для сгенерированных трехдиагональных матриц, поскольку результаты распределения погрешностей на [рис. 4](#) и [рис. 5](#) имеют достаточно равномерный характер. Если исходная матрица обладает свойством строгого диагонального преобладания, то метод прогонки является вычислительно устойчивым[1]. В связи с вышесказанным, был сделан вывод, что данный результат был получен в связи с особенностями генерации трехдиагональных матриц, поскольку большинство из них являются матрицами со строгим диагональным преобладанием.

Однако, решения СЛАУ для некоторых матриц все же обладают высокими значениями погрешностей, что может быть связано с конкретными характеристиками этих матриц, такими как плохая обусловленность или отсутствие свойства строго диагонального преобладания.

2 Разложение Холецкого и расширенный анализ вычислительной устойчивости реализованных методов

2.1 Расширения генератора случайных матриц для получения положительно определенных матриц

Матрица A называется положительно определенной, если она симметричная, и верным является неравенство $x^T A x > 0$ для любого вектора $x \neq 0$ подходящей размерности[1].

Множество положительно определенных матриц является подмножеством положительно полуопределенных матриц. Генерацию матрицы, принадлежащей множеству положительно полуопределенных матриц, можно реализовать путем умножения ее на саму себя транспонированную. Подмножество положительно определенных матриц отличается удовлетворением условию несингулярности. Таким образом, генерация положительно определенной матрицы заключается в перемножении матрицы общего вида на саму себя транспонированную и дальнейшее приведение к несингулярной. Алгоритм проверки матрицы на близость к сингулярной матрице аналогичен такому же алгоритму в генераторе трехдиагональных матриц.

Расширение функции `generate_random_matrix(n, matrix_type)` для генерации случайных положительно определенных матриц представлена в листинге 6. Для получения положительно определенной матрицы необходимо, чтобы параметр `matrix_type = PD`.

Листинг 6. Расширение функции `generate_random_matrix` для генерации случайных положительно определенных матриц

```

1  #Генерация положительно-определенных матриц
2  elif matrix_type == "PD":
3      symmetrical_matrix = np.dot(gen_matrix, gen_matrix.T)
4
5      max_elem = np.max(np.abs(symmetrical_matrix))
6      if max_elem >= 1:
7          symmetrical_matrix /= (max_elem + 0.001)
8
9      while np.abs(np.linalg.det(symmetrical_matrix)) < epsilon:
10         symmetrical_matrix += np.eye(n) * 0.1
11         max_elem = np.max(np.abs(symmetrical_matrix))
12         if max_elem >= 1:
13             symmetrical_matrix /= (max_elem + 0.001)
14
15     gen_matrix = symmetrical_matrix

```

2.2 Метод решения СЛАУ, полученного при помощи разложения Холецкого

Разложение Холецкого представляет собой разложение исходной матрицы коэффициентов A на нижнюю и верхнюю треугольные матрицы вида[1]:

$$A = LL^T, \quad (8)$$

где \mathbf{L} - нижняя треугольная матрица с ненулевыми элементами на диагонали. Известные элементы матрицы \mathbf{L} вычисляются следующим образом:

$$l_{ii} = \sqrt{a_{ii} - \sum_{j=1}^{i-1} l_{ij}^2}, \quad i = 1, \dots, n, \quad (9)$$

$$l_{ij} = \frac{1}{l_{jj}} \left(a_{ij} - \sum_{k=1}^{j-1} l_{ik} l_{jk} \right), \quad j < i, \quad (10)$$

После вычисления всех коэффициентов l_{ij} происходит вычисление компонентов вектора решения СЛАУ \mathbf{x} в соответствии с выражениями следующей системы:

$$\begin{cases} \mathbf{L}\mathbf{y} = \mathbf{b} \\ \mathbf{L}^\top \mathbf{x} = \mathbf{y} \end{cases} \quad (11)$$

Значения компонентов вектора \mathbf{y} из первого уравнения системы 11 определяются при помощи обратного хода Гаусса с прямой подстановкой. Компоненты вектора \mathbf{x} вычисляются с помощью обратного хода Гаусса с обратной подстановкой.

Весь описанный алгоритм для решения СЛАУ, с помощью разложения Холецкого, реализован в функции `cholesky(A, b)` (листинг 7), где также помимо непосредственного вычисления коэффициентов присутствуют проверки на то, что диагональные элементы матрицы \mathbf{L} ненулевые, что является условием разложения Холецкого, и проверку на то, что подкоренное выражение в формуле 9 положительно, так как в ином случае матрица переходит из действительного множества в комплексное.

Листинг 7. Функция для решения СЛАУ методом с использованием разложения Холецкого

```

1 def cholesky(A, b):
2     n = len(A)
3     L = np.zeros((n, n)) # Нижнетреугольная матрица в разложении Холецкого
4     for i in range(n): # По строкам
5         for j in range(i + 1): # По столбцам
6             l_sum = sum([L[i, p] * L[j, p] for p in range(j)]) # Сумма произведений
7                               # элементов предыдущих строк и столбцов матрицы L
8             if i == j: # Лежит ли элемент на главной диагонали
9                 radical_expression = A[i, i] - l_sum
10                if radical_expression < 0:
11                    raise ValueError("Negative under radical")
12                elif radical_expression == 0:
13                    raise ValueError("Matrix L is singular")
14                L[i, j] = np.sqrt(radical_expression)
15            else: # Вычисление недиагональных элементов
16                L[i, j] = ((A[i, j] - l_sum) / L[j, j])
17        y = np.zeros((n, 1))
18        for k in range(n):
19            y[k] = (b[k] - np.dot(L[k, :k], y[:k])) / L[k, k]
20        x = np.zeros((n, 1))
21        for k in range(n - 1, -1, -1):
22            x[k] = (y[k] - np.dot(L.T[k, k + 1:], x[k + 1:])) / L.T[k, k]
23    return x

```

2.3 Анализ метода Холецкого

Была протестирована функция, реализующая метод Холецкого для решения СЛАУ $Ax = b$. В качестве тестовых данных взяты те же самые значения:

$$b = [3.52971, 0.333, 1.6666]^T; \quad A = \begin{bmatrix} 1 & 2 & 0 \\ 2 & 6 & 5 \\ 0 & 5 & 13 \end{bmatrix}$$

Результаты тестирования продемонстрированы на [рис. 6](#).

```

Cholesky:
[[195.08263]
 [-95.77646]
 [ 36.9653 ]]
Answer:
[195.08263 -95.77646  36.9653 ]

```

Рис. 6. Результат тестирования функции cholesky(A, b)

Решение, полученное функцией `cholesky(A, b)`, сравнивалось с решением полученным при помощи метода `numpy.linalg.solve`. Исходя из результатов тестирования, продемонстрированных на [рис. 6](#), сделан вывод что функция работает верно.

Далее был проведен анализ аналогичный п 1.5, проведенный для матриц общего вида и трехдиагональных матриц.

(a) Метод Холецкого был выбран в качестве “специального” вычислительно-эффективного метода для решения СЛАУ с положительно определенными матрицами.

(b) Для анализа решений СЛАУ для положительно определенных матриц с помощью функций `generate_random_matrix(n=6, matrix_type = PD)` были сгенерированы 1000 случайных положительно определенных матриц $A^{(j)}$. Затем найдены решения СЛАУ при помощи метода Гаусса с частичным выбором главного элемента (“универсальный”) и метода Холецкого (“специальный”). Нахождение относительных погрешностей вычислений выполнено при помощи ранее реализованных функций `count_rms_error` и `count_supremum_error`. Результаты визуализации распределения погрешностей в виде гистограмм продемонстрированы на [рис. 7](#) и [рис. 8](#).

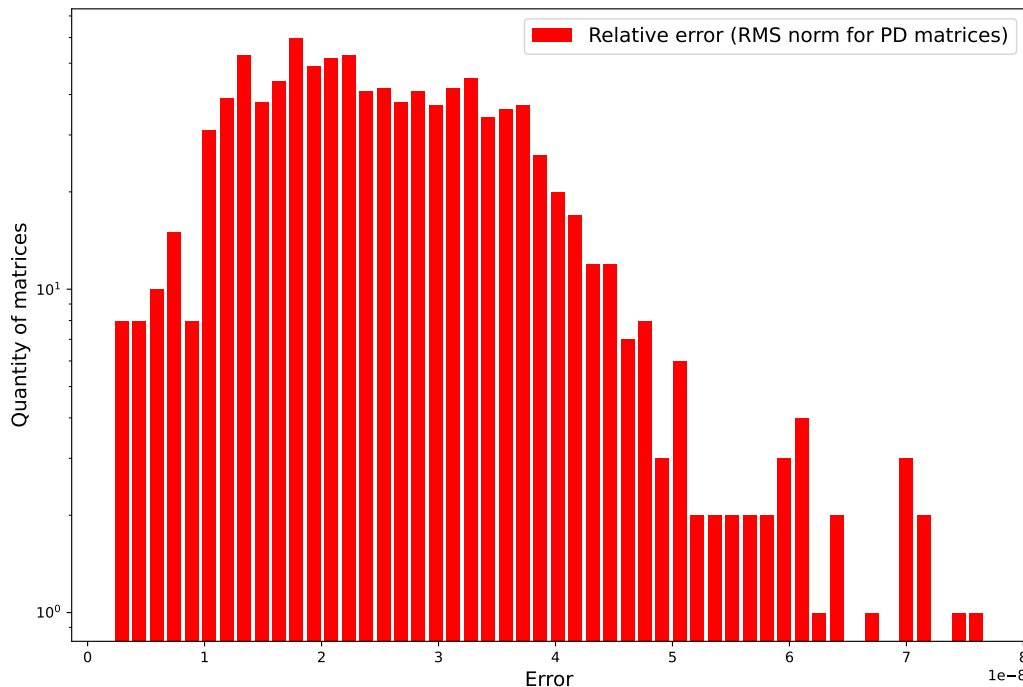


Рис. 7. Распределение относительных погрешностей, полученных с помощью среднеквадратичной нормы, для положительно определенных матриц

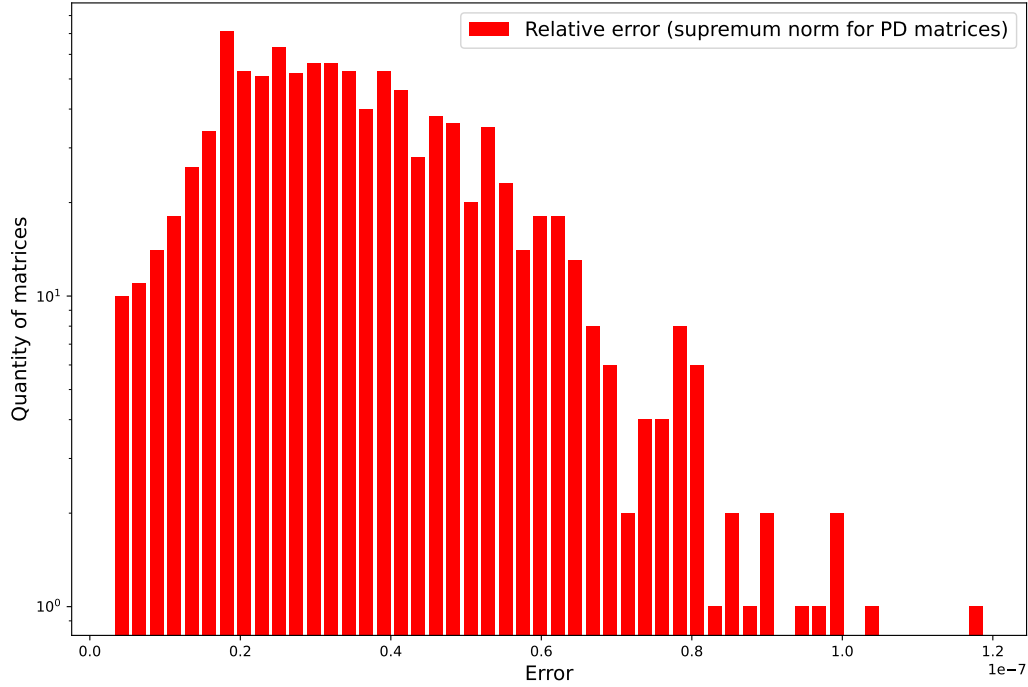


Рис. 8. Распределение относительных погрешностей, полученных с помощью супремум нормы, для положительно определенных матриц

(с) Метод Холецкого достаточно вычислительно устойчив для рассматриваемого типа матриц, поскольку распределение относительных погрешностей на [рис. 7](#) и [рис. 8](#) имеют достаточно равномерный характер, с незначительными отклонениями от основного распределения. Вычислительную устойчивость метода Холецкого для положительно определенных матриц можно аргументировать тем, что все диагональные элементы матрицы L положительны и относительно большие (из-за положительной определённости матрицы A). Значит, метод Холецкого избегает деления на малые числа, что является частой причиной численных проблем в других методах решения СЛАУ.

2.4 Визуализация распределения спектральных радиусов и распределения чисел обусловленности

Спектральным радиусом матрицы A называется число $\rho(A) \in \mathbb{R}$ такое, что [\[1\]](#):

$$\rho(A) = \max_{i \in [1, m]} |\lambda_i|,$$

где λ_i - одно из m собственных чисел матрицы A .

Для нахождения спектральных радиусов, то есть для вычисления максимального собственного числа матриц использована функция `numpy.linalg.eigvals`. Результаты распределения спектральных радиусов для каждого типа матриц, представлены на [рис. 9](#), [рис. 10](#) и [рис. 11](#).

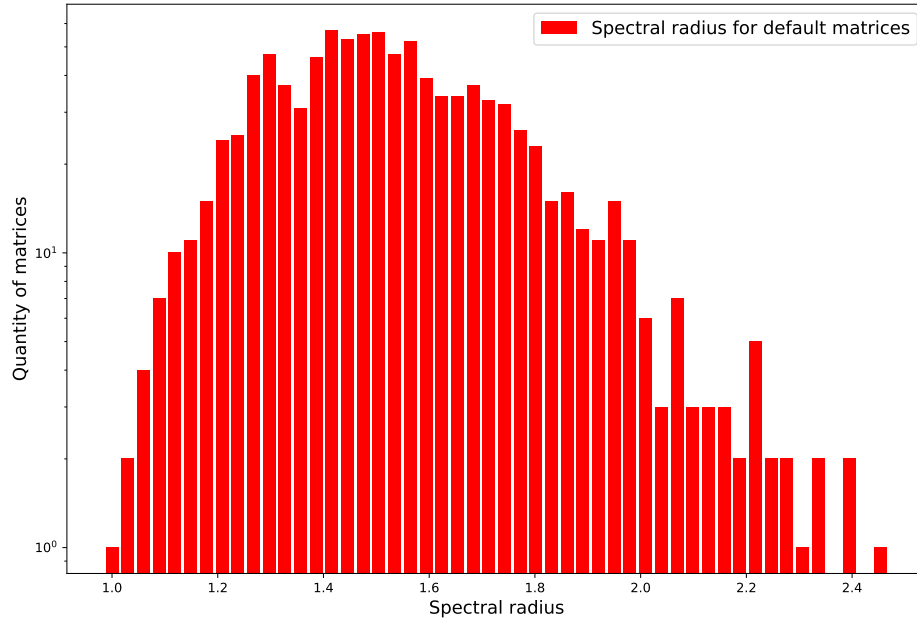


Рис. 9. Распределение спектральных радиусов для матриц общего вида.

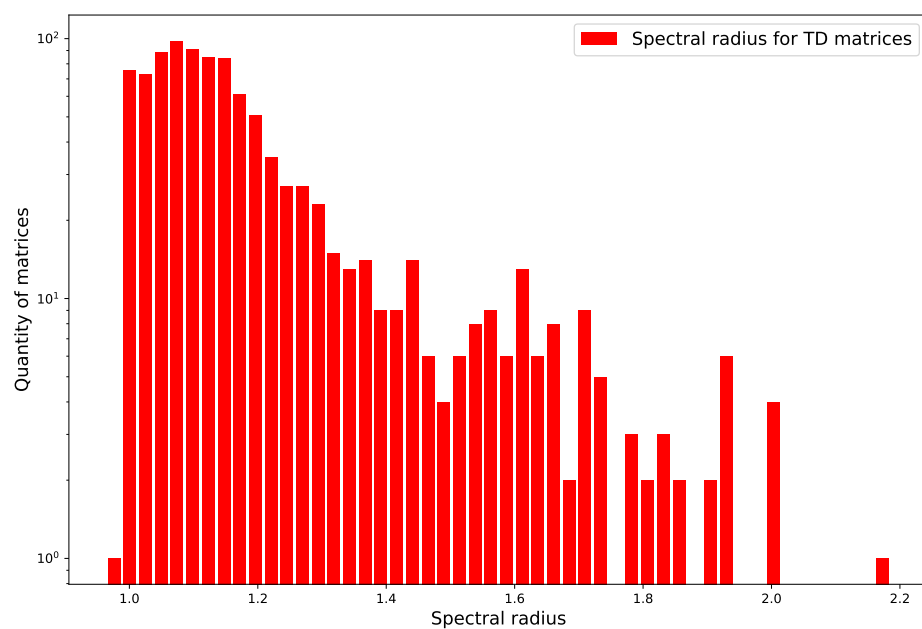


Рис. 10. Распределение спектральных радиусов для трехдиагональных матриц.

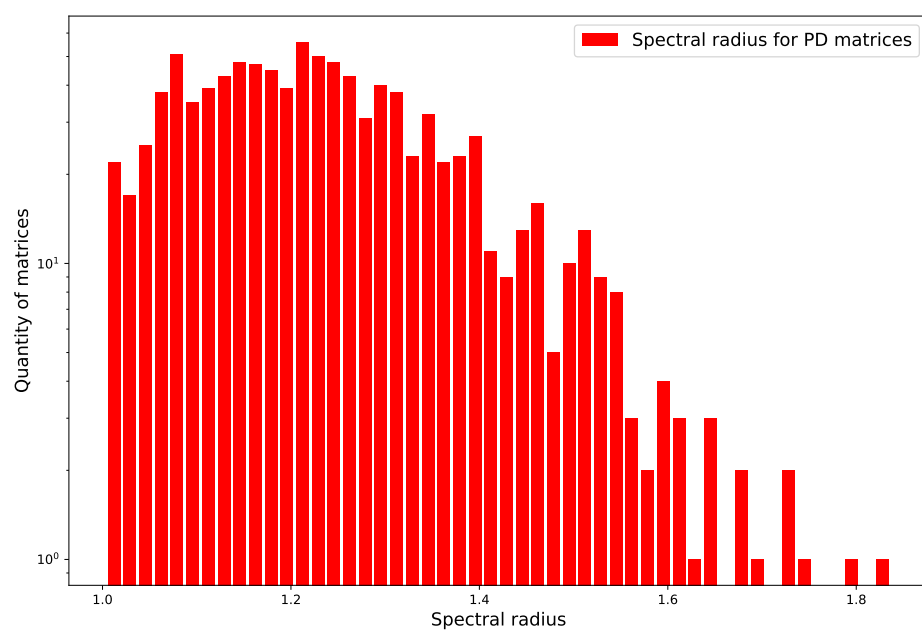


Рис. 11. Распределение спектральных радиусов для положительно определенных матриц.

Числом обусловленности называется следующая величина [1]:

$$K(\mathbf{A}) = \|\mathbf{A}\| \cdot \|\mathbf{A}^{-1}\| \quad (12)$$

Величина, представленная в формуле 13, связана с относительной погрешностью и вектором невязки следующим соотношением:

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

где \mathbf{x} - точное решение, $\tilde{\mathbf{x}}$ - приближенное решение, \mathbf{r} - вектор невязки, \mathbf{b} - вектор правой части СЛАУ. Для нахождения чисел обусловленности была использована функция `numpy.linalg.cond`. Результаты распределения чисел обусловленности для каждого типа матриц, представлены на рис. 12, рис. 13 и рис. 14.

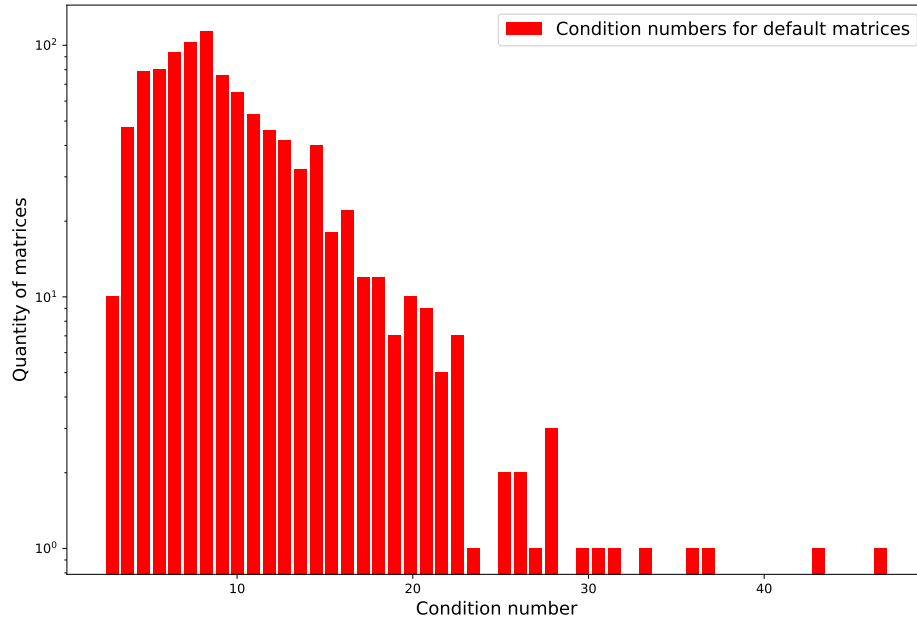


Рис. 12. Распределение чисел обусловленности для матриц общего вида.

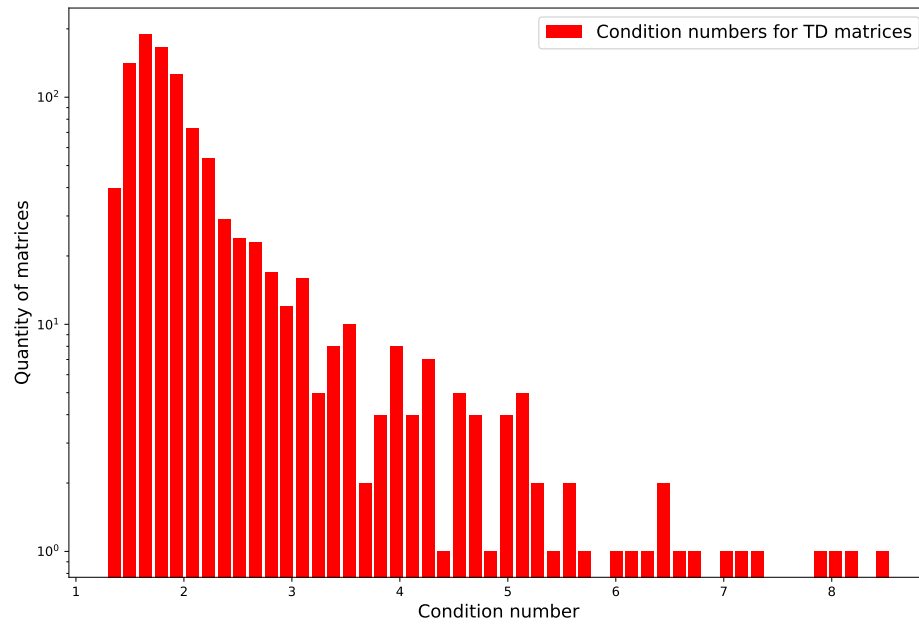


Рис. 13. Распределение чисел обусловленности для трехдиагональных матриц.

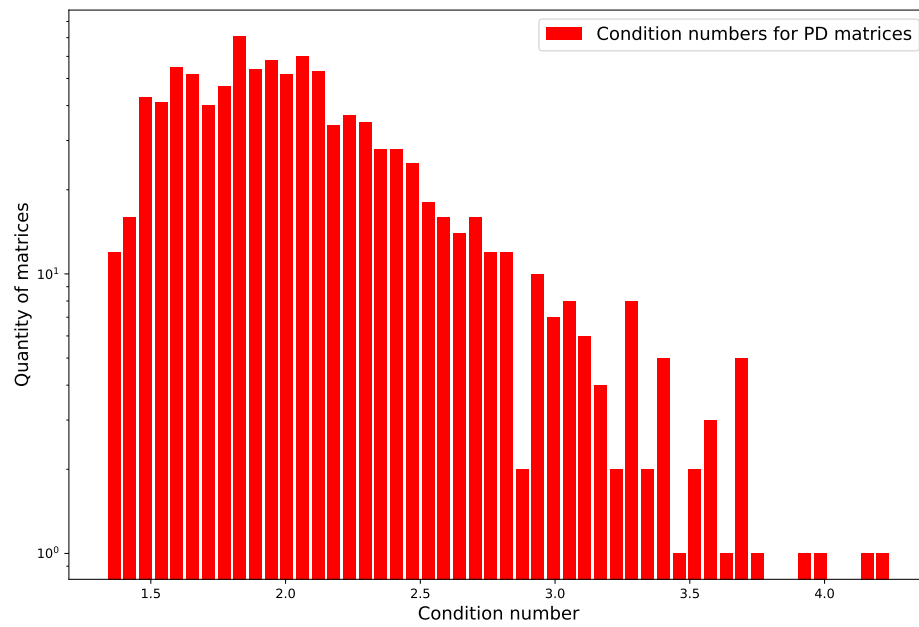


Рис. 14. Распределение чисел обусловленности для положительно определенных матриц.

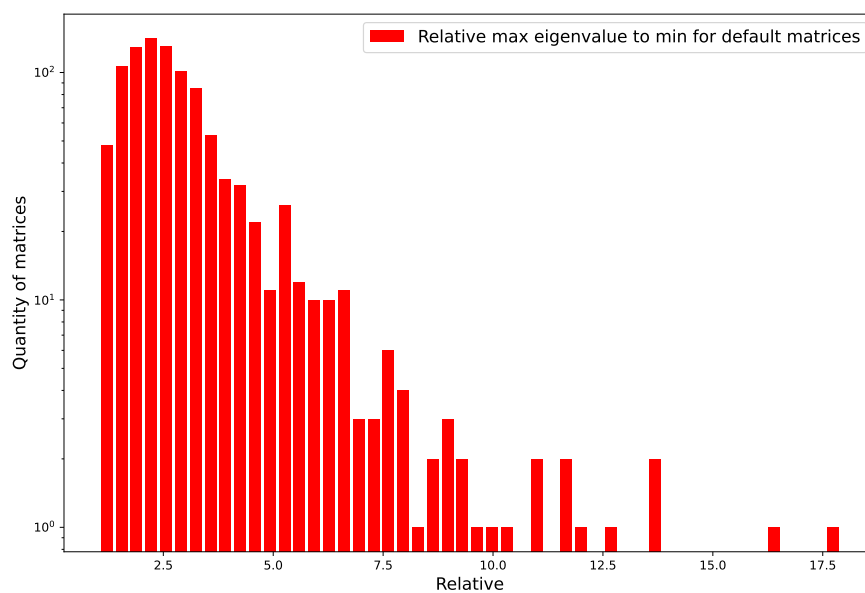


Рис. 15. Распределение отношений максимального по модулю собственного числа к минимальному для матриц общего вида.

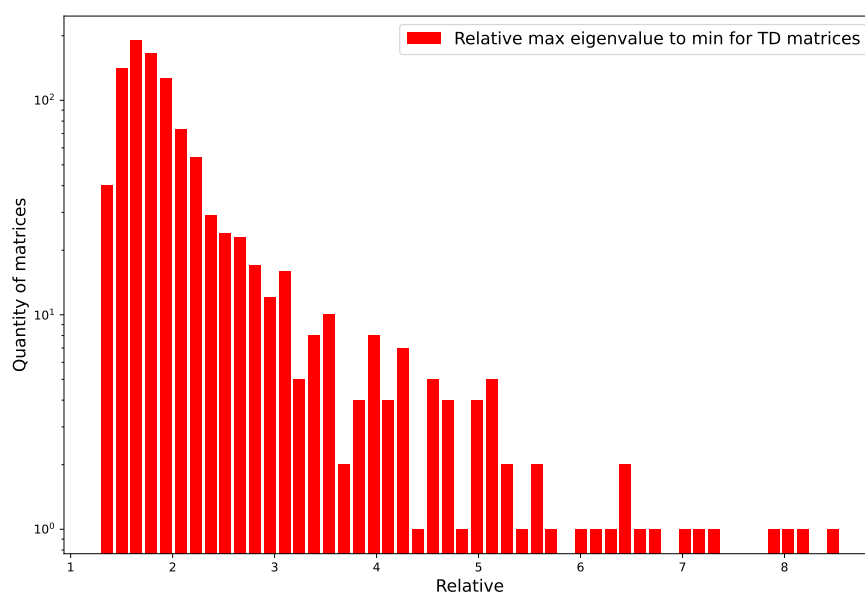


Рис. 16. Распределение отношений максимального по модулю собственного числа к минимальному для трехдиагональных матриц.

словами, число обусловленности позволяет оценить, сколько значимых цифр теряется в процессе вычислений[1]:

$$K(A) \approx \frac{\|x - \tilde{x}\|}{\|\tilde{x}\|} 10^t \Rightarrow \frac{\|x - \tilde{x}\|}{\|\tilde{x}\|} \approx 10^{-t} K(A).$$

На рис. 12 распределение чисел обусловленности происходит на намного большем диапазоне, чем на рис. 13 и рис. 14, при чем большая часть чисел обусловленности имеет значение сильно больше единицы. Это значит, что матрицы общего вида являются плохо обусловленными матрицами, вследствие чего метод Гаусса без выбора главного элемента является вычислительно неустойчивым. Вычислительная устойчивость метода прогонки и метода Холецкого подтверждается более узкими диапазонами значений чисел обусловленности, а также тем, что большая часть значений находятся вблизи $K(A) = 1$.

Заключение



В ходе лабораторной работы были изучены прямые методы решения СЛАУ: метод Гаусса с частичным выбором главного элемента и без него, метод прогонки и метод разложения Холецкого. Данные методы были проанализированы на предмет устойчивости и относительной погрешности для матриц общего вида, трехдиагональных матриц и положительно определенных матриц. Было проанализировано, каким образом спектральный радиус, число обусловленности и отношение максимального по модулю собственного числа к минимальному влияют на вычислительную устойчивость реализованных методов.

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. Першин А.Ю., Соколов А.П., Гудым А.В. Сборник постановок задач на лабораторные работы по курсу «Вычислительная математика»: Учебное пособие. [Электронный ресурс]. Москва, 2023. С. 47. URL: <https://arch.rk6.bmstu.ru>.

Выходные данные

Гапеев Д.А. Отчет о выполнении лабораторной работы по дисциплине «Вычислительная математика». [Электронный ресурс] — Москва: 2023. — 28 с. URL: <https://sa2systems.ru:88> (система контроля версий кафедры РК6)

Постановка:  ассистент кафедры РК-6, PhD Першин А.Ю.
Решение и вёрстка:  студент группы РК6-51Б, Галеев Д.А.

2023, осенний семестр