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:	Турунов Дмитрий Николаевич
Группа:	РК6-53Б
Тип задания:	Лабораторная работа
Тема:	Устойчивость прямых методов решения СЛАУ

Студент _____ Турунов Д.Н.
подпись, дата Фамилия, И.О.

Преподаватель _____ _____
подпись, дата Фамилия, И.О.

Москва, 2023

Содержание

Устойчивость прямых методов решения СЛАУ

Задание

Задача 4.3 (устойчивость прямых методов решения СЛАУ)

Требуется (базовая часть):

1. Написать функцию gauss(A, b, pivoting), которая возвращает решение СЛАУ $\boldsymbol{Ax} = \boldsymbol{b}$, 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 $\boldsymbol{Ax} = \boldsymbol{b}$, 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 $\boldsymbol{A}^{(j)}$ 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 $\boldsymbol{A}^{(j)}\boldsymbol{x} = [1, 1, 1, 1]^T$ 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 $\boldsymbol{Ax} = \boldsymbol{b}$, 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

Целью данной лабораторной работы является изучение и сравнение методов решения систем линейных алгебраических уравнений (СЛАУ) различными алгоритмами: методом Гаусса с и без выбора главного элемента, методом прогонки и методом Холецкого. Основной задачей является анализ вычислительной устойчивости и погрешностей этих методов на основе экспериментов с генерированными матрицами разного типа. Также предполагается оценка влияния спектрального радиуса, числа обусловленности и отношения максимального и минимального собственных значений на вычислительную устойчивость используемых алгоритмов.

1 Базовая часть

1. Метод Гаусса для решения СЛАУ

Описание:

Метод Гаусса применяется для решения систем линейных уравнений $\boldsymbol{Ax} = \boldsymbol{b}$. Основная идея состоит в приведении матрицы к верхнетреугольной форме и последующем решении системы обратной подстановкой. Вариант с частичным выбором главного элемента включает выбор наибольшего по модулю элемента в текущем столбце и перестановку строк для уменьшения вычислительных погрешностей.

Формулы:

- Прямой ход (без выбора главного элемента):

Для i-й строки, j-го столбца ($i, j \geq k$, где k - текущий шаг итерации):

$$a_{ij}^{(new)} = a_{ij} - \frac{a_{ik}}{a_{kk}} a_{kj},$$

$$b_i^{(new)} = b_i - \frac{a_{ik}}{a_{kk}} b_k.$$

- Частичный выбор главного элемента:

На k-м шаге итерации выбирается строка m так, что:

$$m = \arg\max_{i \geq k} |a_{ik}|.$$

Затем строки k и m меняются местами.

- Обратная подстановка:

После приведения к верхнетреугольной форме решение находится через обратную подстановку:

$$x_n = \frac{b_n}{a_{nn}},$$

$$x_i = \frac{1}{a_{ii}} \left(b_i - \sum_{j=i+1}^{n} a_{ij} x_j, \right), \quad i = n-1, n-2, \ldots, 1.$$

Код:

Реализована функция ‘gauss(A, b, pivoting)‘ (листинг 1). Она принимает на вход матрицу коэффициентов ‘A‘, вектор правых частей ‘b‘ и флаг ‘pivoting‘ для выбора метода с частичным выбором главного элемента, и возвращает вектор ‘x‘, содержащий решение системы линейных уравнений $\boldsymbol{Ax} = \boldsymbol{b}$.

Листинг 1. Метод Гаусса

```
def gauss(A, b, pivoting=False):
    A = A.copy()
    b = b.copy()
    n = len(b)

    for i in range(n):
        if pivoting:
            max_row = np.argmax(np.abs(A[i:, i])) + i
            A[[i, max_row]] = A[[max_row, i]]
            b[[i, max_row]] = b[[max_row, i]]

        for j in range(i+1, n):
            factor = A[j, i] / A[i, i]
            A[j, i:] -= factor * A[i, i:]
            b[j] -= factor * b[i]

    x = np.zeros_like(b)
    for i in range(n-1, -1, -1):
        x[i] = (b[i] - np.dot(A[i, i+1:], x[i+1:])) / A[i, i]

    return x
```

2. Метод Прогонки для трехдиагональных СЛАУ

Описание:

Метод прогонки применяется для эффективного решения трехдиагональных систем уравнений. Он включает два этапа: прямой ход для вычисления прогоночных коэффициентов и обратный ход для нахождения решения системы.

Формула:

В прямом ходе для i-го уравнения вычисляются коэффициенты:

$$\alpha_{i+1} = \frac{-c_i}{b_i + a_i\alpha_i},$$

$$\beta_{i+1} = \frac{d_i - a_i\beta_i}{b_i + a_i\alpha_i},$$

где a_i, b_i, c_i - элементы матрицы, d_i - элементы вектора правой части. В обратном ходе находим x_i:

$$x_i = \alpha_{i+1}x_{i+1} + \beta_{i+1}.$$

Код:

Реализована функция ‘thomas(A, b)‘ (листинг 2). Она принимает на вход трехдиагональную матрицу коэффициентов ‘A‘, вектор правых частей ‘b‘ и возвращает вектор ‘x‘, содержащий решение системы линейных уравнений $\boldsymbol{Ax} = \boldsymbol{b}$.

Листинг 2. Метод прогонки

```
def thomas(A, b):
    n = len(b)
    c_prime = np.zeros(n-1)
    d_prime = np.zeros(n)

    c_prime[0] = A[0, 1] / A[0, 0]
    d_prime[0] = b[0] / A[0, 0]

    for i in range(1, n):
        a = A[i, i-1]
        d = A[i, i]
        if i < n - 1:
            c = A[i, i+1]
            c_prime[i] = c / (d - a * c_prime[i-1])

        d_prime[i] = (b[i] - a * d_prime[i-1]) / (d - a * c_prime[i-1])
    x = np.zeros_like(b)
    x[-1] = d_prime[-1]
    for i in range(n-2, -1, -1):
        x[i] = d_prime[i] - c_prime[i] * x[i+1]

    return x
```

3. Определение универсального метода

Наиболее универсальным является метод Гаусса с частичным выбором главного элемента. Данный метод обладает преимуществом перед классическим методом Гаусса без выбора главного элемента и методом прогонки в контексте обработки квадратных матриц общего вида.

Основная причина предпочтения метода Гаусса с частичным выбором главного элемента заключается в его способности уменьшать ошибки округления, которые нередко возникают в процессе вычислений. В отличие от классического метода Гаусса, где последовательное исключение неизвестных может привести к значительным ошибкам при наличии малых коэффициентов, выбор главного элемента минимизирует этот эффект за счет выбора наибольшего по абсолютному значению элемента в текущем столбце матрицы.

Метод прогонки, хотя и эффективен для трехдиагональных матриц, не является подходящим для квадратных матриц общего вида. Он ограничен своей специализацией и не может быть применен в ситуациях, когда матрица системы не обладает трехдиагональной структурой.

Таким образом, для обеспечения минимальных вычислительных погрешностей в условиях обработки квадратных матриц общего вида метод Гаусса с частичным выбором главного элемента является предпочтительным выбором.

4. Генерация случайных невырожденных матриц

Формула:

Для общей матрицы:

$$a_{ij} = \text{rand}(-1, 1).$$

Для трехдиагональной:

$$a_{ij} = \begin{cases} \text{rand}(-1,1) & \text{если } |i-j| \le 1, \\ 0 & \text{иначе,} \end{cases}$$

где $\text{rand}(-1, 1)$ генерирует случайное число в диапазоне $[-1, 1]$.

Код:

Реализована функция ‘generate_random_matrices(n, tridiagonal)‘, представленная в листинге 3. Данная функция принимает два параметра: количество матриц ‘n‘ и логический флаг ‘tridiagonal‘, указывающий на необходимость генерации трехдиагональных матриц. В результате своей работы функция возвращает список матриц.

Листинг 3. Генератор матриц

```
1 def generate_random_matrices(n, tridiagonal=False):
2     matrices = []
3     while len(matrices) < n:
4         if tridiagonal:
5             a = np.diag(np.random.rand(6)*2 - 1)
```

```
            b = np.diag(np.random.rand(5)*2 - 1, k=1)
            c = np.diag(np.random.rand(5)*2 - 1, k=-1)
            mat = a + b + c
        else:
            mat = np.random.rand(6, 6)*2 - 1

        if np.linalg.det(mat) != 0:
            matrices.append(mat.astype(np.float32))

    return matrices
```

5. Эксперимент по сравнению методов

Формула:

Среднеквадратичная норма погрешности ϵ_{rms} и супремум-норма ϵ_{sup} рассчитываются как:

$$\epsilon_{rms} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{x}_i)^2},$$

$$\epsilon_{sup} = \max_i |x_i - \hat{x}_i|,$$

где x_i - истинное решение, $\hat{x}_i$ - найденное решение.

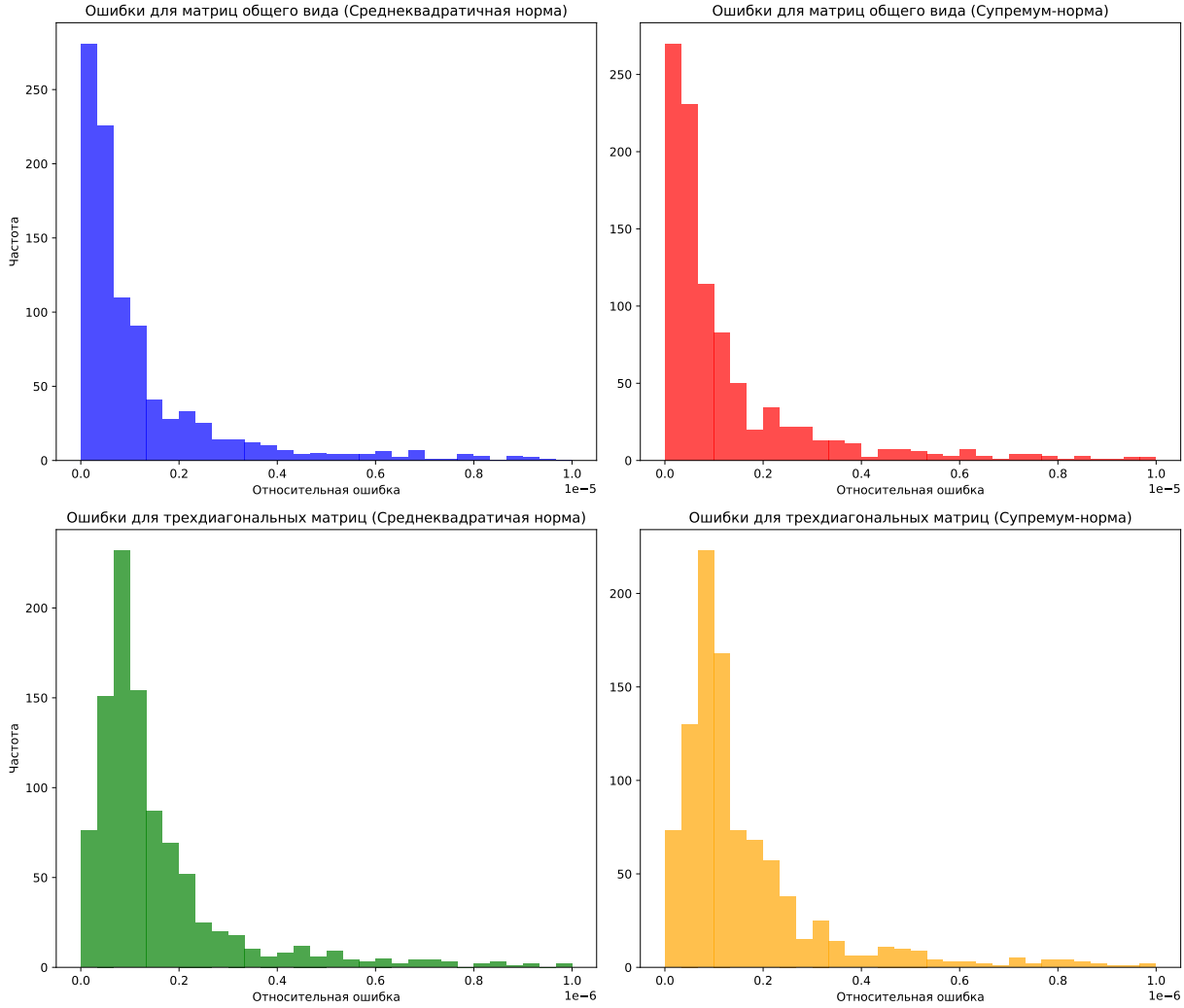

Рис. 1. Гистограммы ошибок.

На рисунке 1 можно видеть:

1. Ошибки для матриц общего вида:

   - Гистограмма ошибок в квадратичной норме показывает, что большинство относительных ошибок сосредоточено на меньших значениях, но при этом наблюдаются и значения с большими ошибками, что указывает на переменчивую устойчивость метода.
   - Гистограмма ошибок в супремум-норме отражает схожую картину: большая часть ошибок приходится на меньшие значения, однако имеются и значительные отклонения.

2. Ошибки для трёхдиагональных матриц :

- Гистограмма ошибок в квадратичной норме для трёхдиагональных матриц демонстрирует, что большинство ошибок находится на нижнем уровне относительной ошибки, с меньшим количеством больших ошибок, что свидетельствует о повышенной устойчивости метода прогонки.
- Гистограмма ошибок в супремум-норме показывает аналогичную картину: большая часть ошибок сосредоточена на меньших значениях, с незначительным количеством больших ошибок.

Вывод: Метод прогонки для трёхдиагональных матриц обладает более высокой вычислительной устойчивостью по сравнению с методом Гаусса для общих матриц. Это подтверждается распределением относительных ошибок, где большинство значений сосредоточено на меньших ошибках, с меньшим количеством крупных ошибок, что указывает на меньшие и более однородные ошибки в серии испытаний, являясь важным показателем вычислительной устойчивости.

2 Продвинутая часть

1. Генерация положительно-определенных матриц

Алгоритм:

- Сгенерировать симметричную матрицу A как $\frac{A+A^T}{2}$, где A — случайная матрица.
- Увеличить диагональные элементы матрицы A, добавив к ним n, где n — размер матрицы.
- Нормализовать A, поделив на максимальное абсолютное значение в A.
- Проверить положительную определенность: A принимается, если все собственные значения A строго больше нуля.

Код:

Функция generate_positive_definite_matrix (листинг 4) принимает на вход размер матрицы (по умолчанию 6) и возвращает положительно-определенную матрицу $\mathbf{A}$.

Листинг 4. Генератор положительно-определенных матриц

```
def generate_positive_definite_matrix(size=6):
    while True:
        A = np.random.rand(size, size) * 2 - 1
        A = (A + A.T) / 2

        A += np.eye(size) * size

        max_val = np.max(np.abs(A))
        A = A / max_val

```

```
11          if np.all(np.linalg.eigvals(A) > 0):
12              return A
```

2. Разложение Холецкого для решения СЛАУ

Алгоритм:

Разложение Холецкого матрицы A представляет собой разложение вида $A = L \cdot L^T$, где L — нижнетреугольная матрица. Для решения СЛАУ $Ax = b$ используются следующие шаги:

1. Найти L из $A = L \cdot L^T$.
2. Решить $L \cdot y = b$ для y методом прямой подстановки.
3. Решить $L^T \cdot x = y$ для x методом обратной подстановки.

Здесь y и x — промежуточные и конечные векторы решений соответственно.

Код:

Реализована функция ‘cholesky(A, b)‘ (листинг 5). Она принимает на вход матрицу коэффициентов ‘A‘, вектор правых частей ‘b‘ и возвращает вектор ‘x‘, содержащий решение системы линейных уравнений $\boldsymbol{Ax} = \boldsymbol{b}$.

Листинг 5. Разложение Холецкого

```
1 def cholesky(A, b):
2     L = np.linalg.cholesky(A)
3     y = np.linalg.solve(L, b)
4     x = np.linalg.solve(L.T, y)
5     return x
```

3. Анализ эффективности метода Холецкого

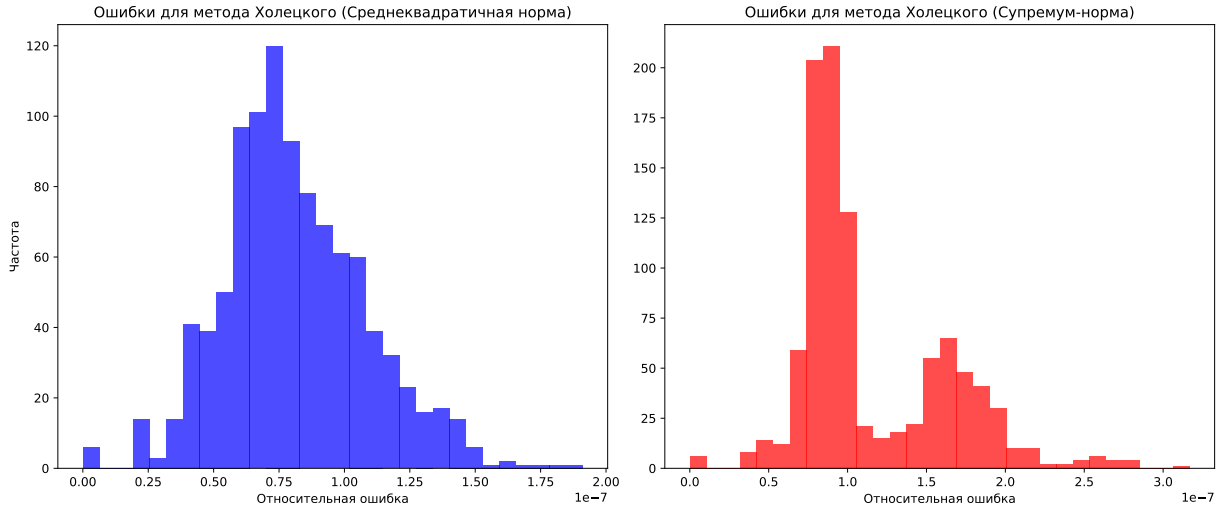

Рис. 2. Распределения ошибок для метода Холецкого.

На рисунке 2 можно видеть распределение ошибок для разложения Холецкого. При сравнении с методом Гаусса, разложение Холецкого демонстрирует более низкий уровень ошибок.

4. Анализ спектральных радиусов и чисел обусловленности матриц: Гистограммы и выводы

Спектральный радиус матрицы A определяется как максимальное абсолютное значение среди её собственных значений:

$$\rho(A) = \max |\lambda_i|$$

где λ_i — собственные значения A.

Число обусловленности матрицы A определяется как:

$$\kappa(A) = \|A\| \cdot \|A^{-1}\|$$

где $\|A\|$ — норма матрицы A, а A^{-1} — обратная матрица A.

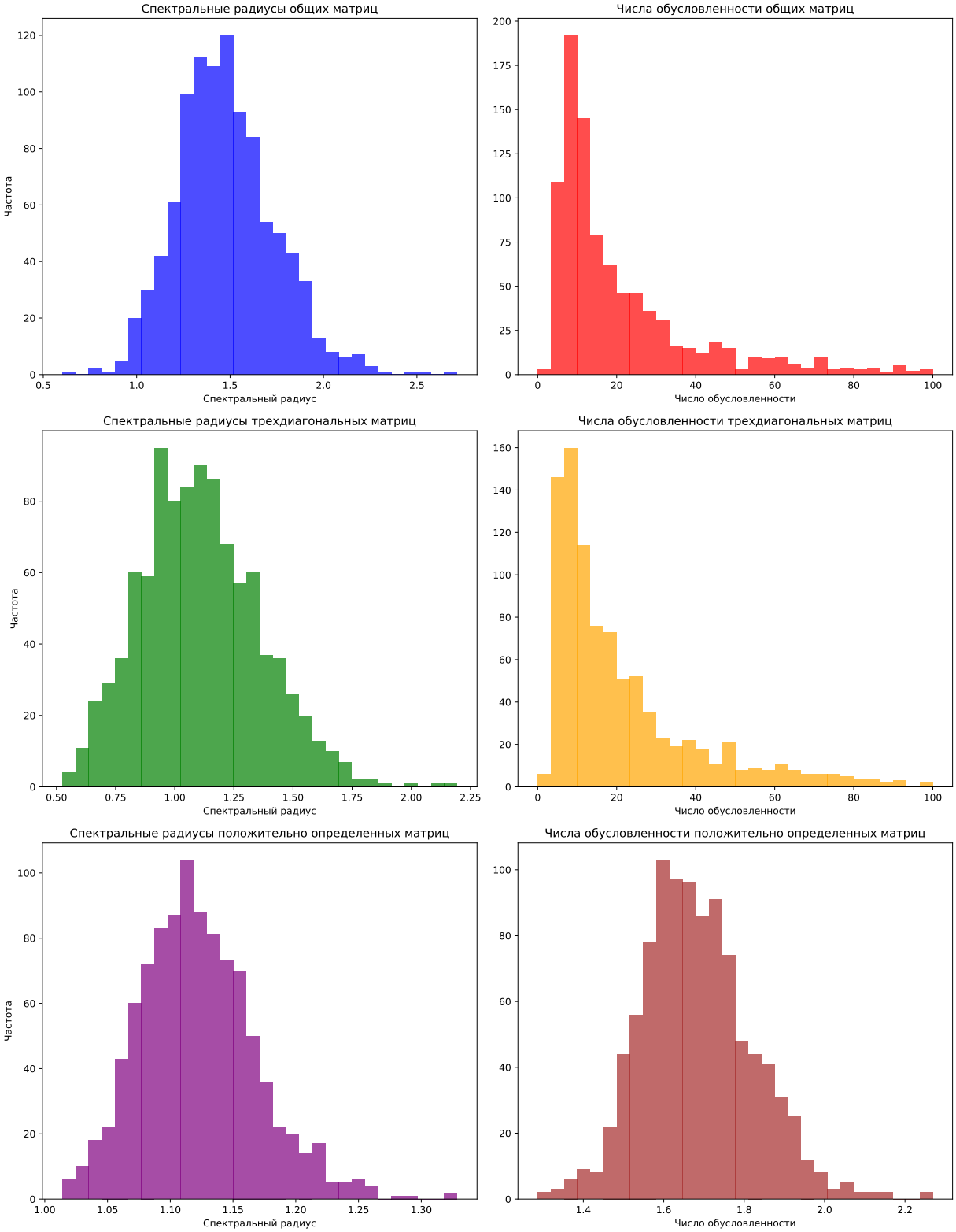

Рис. 3. Спектральные радиусы и числа обусловленности.

На Рисунке 3 представлены распределения спектральных радиусов и чисел обусловленности для всех категорий матриц, использованных в данной лабораторной работе. Для улучшения визуальной интерпретации графики чисел обусловленности были ограничены значением 100. Отмечается, что положительно определённые матрицы демонстрируют наиболее благоприятные показатели чисел обусловленности, что объясняется их специфическими свойствами, такими как отсутствие нулевых собственных значений и симметричность. Это приводит к более стабильному и предсказуемому поведению в контексте вычислительной устойчивости. В отношении спектральных радиусов, данные показывают нормальное распределение для всех трёх типов матриц, что указывает на отсутствие выраженного асимметричного смещения в их значениях и предполагает статистическую однородность в рамках каждого типа.

5. Влияние спектрального радиуса на устойчивость алгоритмов:

Спектральный радиус матрицы, определяемый как максимальное абсолютное значение собственных чисел, влияет на вычислительную устойчивость итерационных методов решения СЛАУ. Уменьшение спектрального радиуса часто ведет к ускорению сходимости метода и уменьшению числа итераций, необходимых для достижения заданной точности. Однако для прямых методов, таких как метод Гаусса или разложение Холецкого, спектральный радиус не играет столь значительной роли в устойчивости, поскольку они не используют итерационные процессы.

6. Влияние отношения собственных чисел на устойчивость:

Отношение максимального к минимальному по модулю собственного числа матрицы (частный случай числа обусловленности при использовании спектральной нормы) напрямую влияет на вычислительную устойчивость алгоритмов. Большое значение этого отношения указывает на то, что матрица плохо обусловлена, что может привести к значительным ошибкам из-за потери точности при вычислениях с плавающей запятой. В таких случаях маленькие изменения входных данных могут привести к большим изменениям в результатах, делая алгоритм менее устойчивым к ошибкам округления.

7. Влияние числа обусловленности на устойчивость:

Число обусловленности матрицы является мерой её чувствительности к ошибкам входных данных и ошибкам округления. Оно определяется как отношение максимального к минимальному сингулярному значению матрицы. Высокое число обусловленности указывает на то, что решение СЛАУ может быть чрезвычайно чувствительно к небольшим изменениям в коэффициентах матрицы или вектора правой части. Таким образом, при высоком числе обусловленности даже незначительные вычислительные ошибки могут привести к значительным отклонениям в решении, снижая вычислительную устойчивость алгоритма. В таких случаях предпочтительны алгоритмы с повышенной устойчивостью или использование предварительной обработки для улучшения числа обусловленности матрицы.

3 Заключение

1. Реализация Алгоритмов: Успешно реализованы алгоритмы для решения СЛАУ - метод Гаусса с частичным выбором главного элемента и без него, метод прогонки и метод Холецкого.

2. Анализ "Универсального"Метода: Определен метод, который минимизирует вычислительные погрешности для квадратных матриц общего вида.

3. Генерация Матриц: Разработан и реализован алгоритм для генерации случайных невырожденных матриц размерности 6×6 как общего, так и трехдиагонального вида, включая положительно-определенные матрицы.

4. Экспериментальный Анализ: Проведен эксперимент с использованием 1000 случайных матриц каждого типа. Сравнение "универсального"метода и "специального"метода показало различия в вычислительной устойчивости и погрешностях.

5. Анализ Вычислительной Устойчивости: Определено, насколько эффективен выбранный "специальный"метод с точки зрения вычислительной устойчивости.

6. Спектральный Радиус и Число Обусловленности: Проведен анализ влияния спектрального радиуса и числа обусловленности на вычислительную устойчивость алгоритмов. Это позволило оценить, как эти характеристики матрицы влияют на точность и стабильность решений СЛАУ.

7. Обобщенные Выводы: Сделаны выводы о том, как характеристики матриц и выбор метода решения влияют на точность и эффективность решения СЛАУ, что имеет важное значение для практического применения данных алгоритмов.

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

Турунов Д.Н.Отчет о выполнении лабораторной работы №4 по дисциплине «Вычислительная математика». [Электронный ресурс] — Москва: 2023. — 16 с. URL: https://sa2systems.ru:88 (система контроля версий кафедры РК6)

Постановка: *аспирант кафедры РК-6, Гудым А.В.*

Решение и вёрстка: *студент группы РК6-53Б, Турунов Д.Н.*

2023, осенний семестр