Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего 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	Шпиталев Даниил Олегович
Группа	РК6-56Б
Тип задания	лабораторная работа 4 Вариант 3
Тема лабораторной работы	Устойчивость прямых методов решения СЛАУ

Студент _____ **Шпиталев Д.О.**
подпись, дата *фамилия, и.о.*

Преподаватель _____ **Соколов А.П.**
подпись, дата *фамилия, и.о.*

Москва, 2023 г.

Оглавление

Задание на лабораторную работу

Устойчивость прямых методов решения СЛАУ.

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 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 $\boldsymbol{Ax} = \boldsymbol{b}$, 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 “точным”, предполагая, что он даёт точное решение СЛАУ.

4. Разработать и описать алгоритм генерации случайных невырожденных матриц размерности 6 × 6 с элементами $a_{ij} \in \mathrm{R}$, $|a_{ij}| < 1$ 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 $\boldsymbol{A}^{(j)}\boldsymbol{x} = [1, 1, 1, 1, 1, 1]^T$ 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. Написать функцию cholesky(A, b), которая возвращает решение СЛАУ $\boldsymbol{Ax} = \boldsymbol{b}$, полученное с помощью разложения Холецкого.

2. Провести требуемый в базовой части задания анализ, в том числе, учитывая метод Холецкого.

3. Вывести на экран распределение спектральных радиусов и распределение чисел обусловленности для сгенерированных матриц $\boldsymbol{A}^{(j)}$ в виде гистограмм. Сделать вывод.

4. Влияет ли значение спектрального радиуса матрицы на вычислительную устойчивость алгоритма? Если да, то как?

5. Влияет ли значение отношения максимального по модулю собственного числа к минимальному по модулю собственному числу на вычислительную устойчивость алгоритма? Если да, то как?

6. Влияет ли число обусловленности на вычислительную устойчивость алгоритма? Если да, то как?

Требуется (опциональное задание):

1. Написать функцию iter_refinement(A, b), которая возвращает решение СЛАУ $\boldsymbol{Ax} = \boldsymbol{b}$, полученное с помощью итерационного уточнения, полученного на основе решения методом Гаусса с частичным или полным выбором главного элемента.

2. Провести требуемый в базовой части задания анализ, в том числе, учитывая реализованную вариацию метода итерационного уточнения.

Цель лабораторной работы

Цель лабораторной работы – изучить и реализовать решение систем линейных алгебраических уравнений прямыми методами, в том числе методами, которые используются для определенных классов матриц. Требуется разработать функции для решения СЛАУ для класса матриц общего вида методом Гаусса с возможностью использования частичного выбора главного элемента, для класса трехдиагональных матриц методом прогонки, для класса положительно определенных матриц методом на основе разложения Холецкого. Требуется сделать выводы по устойчивости данных методов, взяв во внимание их реализацию, класс матриц, для которых они предназначены, а также спектральные радиусы и числа обусловленности матриц, характеризующих СЛАУ.

1. Решение СЛАУ с помощью метода Гаусса

Матричная форма СЛАУ имеет вид:

$$\begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \dots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix} \quad (1)$$

Или кратко:

$$\boldsymbol{Ax} = \boldsymbol{b}$$

Метод Гаусса состоит из прямого и обратного хода. Прямой ход включает в себя преобразование расширенной матрицы (2) к верхнему треугольному виду с помощью элементарных преобразований. Обратный ход включает в себя нахождение вектора решений с помощью рекурсивной подстановки.

Расширенная матрица имеет вид:

$$\overline{\boldsymbol{A}} = [\boldsymbol{A}, \boldsymbol{b}] = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} & b_1 \\ a_{21} & a_{22} & \dots & a_{2n} & b_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} & b_n \end{bmatrix} \quad (2)$$

Необходимо последовательно обнулять элементы под главной диагональю, для этого будем использовать следующее элементарное преобразование, позволяющее обнулить столбец под диагональным элементом a_{11}:

$$(i) \to (i) - \frac{a_{i1}}{a_{11}}(1), i = 2,3, \dots n \quad (3)$$

где (i) обозначает i-ую строку расширенной матрицы. Тогда, применив данное элементарное преобразование (3) к (2), получим (4):

$$\overline{\boldsymbol{A}} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} & b_1 \\ 0 & a_{22}^{(1)} & \dots & a_{2n}^{(1)} & b_2^{(1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & a_{n2}^{(1)} & \dots & a_{nn}^{(1)} & b_n^{(1)} \end{bmatrix} \quad (4)$$

где $a_{ij}^{(1)} = a_{ij} - \frac{a_{i1}}{a_{11}} a_{1j}$ и $b_i^{(1)} = b_j - \frac{a_{i1}}{a_{11}} b_1$.

Аналогично, для обнуления столбца под диагональным элементом $a_{22} \dots a_{n-2,n-2}$ воспользуемся:

$$(i) \to (i) - \frac{a_{i2}}{a_{22}}(2), i = 3,4, \dots n$$

$$\dots \quad (5)$$

$$(i) \to (i) - \frac{a_{i,n-2}}{a_{n-2,n-2}}(n-2), i = n-1, n$$

Применяя подобные преобразования (5) каскадно вплоть до последнего диагонального элемента получим верхнюю треугольную матрицу (6):

$$\overline{\boldsymbol{A}} = \begin{bmatrix} a_{11} & a_{12} & \dots & \dots & a_{1n}^{(1)} & b_1 \\ 0 & a_{22}^{(1)} & \dots & \dots & a_{2n}^{(1)} & b_2^{(1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & a_{n-1,n-1}^{(n-2)} & a_{n-1,n}^{(n-2)} & b_{n-1}^{(n-2)} \\ a_{n1}^{(1)} & a_{n2}^{(1)} & \dots & 0 & a_{nn}^{(n-1)} & b_n^{(n-1)} \end{bmatrix} \quad (6)$$

Получив верхнюю треугольную матрицу, найдём решение СЛАУ с помощью обратного хода метода Гаусса (7):

$$x_n = \frac{b_n^{(n-1)}}{a_{nn}^{(n-1)}}, x_{n-1} = \frac{b_{n-1}^{(n-2)} - a_{n-1,n}^{(n-2)} x_n}{a_{n-1,n-1}^{(n-2)}}, x_1 = \frac{b_1 - \sum_{i=2}^{n} a_{1i} x_i}{a_{11}} \quad (7)$$

В общем случае метод Гаусса неустойчивый, так как при прямом ходе на некоторой k-ой итерации происходит деление на диагональные элементы $a_{kk}^{(k-1)}$, которые могут быть малы по сравнению с элементами $a_{ik}^{(k-1)}$, что приводит к усилению погрешности округления. Кроме того, диагональный элемент может быть равен нулю, что приводит к невозможности вычислений. Простым способом решения данной проблемы является использование метода частичного выбора главного элемента. Данный метод заключается в перестановки строк матрицы так, чтобы диагональным элементом был

наибольший по модулю элемент k-го столбца на каждой итерации прямого хода метода Гаусса.

Функция, реализующая метод Гаусса, представлена в листинге 1. Она принимает на вход матрицы $\boldsymbol{A}$ и $\boldsymbol{b}$, а также булеву переменную для использования частичного выбора главного элемента. Алгоритм частичного выбора главного элемента работает следующим образом. На i-ой итерации прямого хода проверяется значение элементов i-ого столбца (под диагональным элементом). Если какой-то из элементов больше диагонального, то i-ая строка меняется местами со строкой, в которой был найден элемент с большим значением.

Листинг 1: функция метода Гаусса с частичным выбором главного элемента

```
def gauss(A, b, pivoting=False):
    n = len(A)
    extended_matrix = np.column_stack((A, b))
    for i in range(n):
        if pivoting:
            max_element_row = i
            for j in range(i+1, n):
               if abs(extended_matrix[j][i]) >
abs(extended_matrix[max_element_row][i]):
                    max_element_row = j
            extended_matrix[[i, max_element_row]] =
extended_matrix[[max_element_row, i]]
        for j in range(i + 1, n):
            factor = extended_matrix[j, i] / extended_matrix[i, i]
            extended_matrix[j, i:] -= factor * extended_matrix[i, i:]
    x = np.zeros(n+1, dtype=dtype)
    for i in range(n - 1, -1, -1):
        x[i] = (extended_matrix[i][-1] - np.dot(extended_matrix[i, i + 1:],
x[i + 1:])) / extended_matrix[i, i]
    return x[:-1]
```

2. Решение СЛАУ с помощью метода прогонки.

Метод прогонки – это специальный метод решения СЛАУ, имеющих трехдиагональную матрицу $\boldsymbol{A}$. Трехдиагональная матрица имеет вид (8):

$$\boldsymbol{A} = \begin{bmatrix} a_{11} & a_{12} & 0 & \dots & 0 & 0 \\ a_{21} & a_{22} & a_{23} & \dots & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & a_{n-1,n-1} & a_{n-1,n-1} & a_{n-1,n} \\ 0 & \dots & \dots & 0 & a_{n,n-1} & a_{nn} \end{bmatrix} \quad (8)$$

Трехдиагональная матрица имеет ненулевые элементы только на главной диагонали и двух соседних диагоналях рядом с ней. Все остальные элементы равны нулю.

Решение СЛАУ будем находить по формуле (9):

$$x_{i-1} = \gamma_i x_i + \beta_i, \tag{9}$$

где $\gamma_{i+1} = \frac{-a_{i,i+1}}{a_{i,i-1}\gamma_i + a_{ii}}, \beta_{i+1} = \frac{b_i - a_{i,i-1}\beta_i}{a_{i,i-1}\gamma_i + a_{ii}}, \gamma_1 = \beta_1 = 0, x_n = \frac{b_n - a_{n,n-1\beta_n}}{a_{n,n-1}\gamma_n}$

При использовании метода прогонки можно найти решение СЛАУ с помощью $O(n)$ операций, что делает данный метод более предпочтительным перед методом Гаусса, который требует $O(n^3)$.

В листинге 2 указана функция, реализующая метод прогонки. В функции копируются матрицы, переданные в качестве аргументов. Сначала рассчитываются все коэффициенты γ_i и β_i по формуле (9) итерационным образом, затем рассчитываются элементы вектора неизвестных $\boldsymbol{x}$, также с помощью (9).

Листинг 2: функция метода прогонки

```
def thomas(A, b):
    local_A = A.copy()
    local_b = b.copy()
    n = len(A)
    gamma = np.zeros(n)
    betta = np.zeros(n)
    x = np.zeros(n)
    for i in range(n-1):
        gamma[i+1] = (-local_A[i, i+1]) / (local_A[i, i-1]*gamma[i] +
local_A[i, i])
        betta[i+1] = (local_b[i] - local_A[i, i-1]*betta[i]) / (local_A[i, i-
1]*gamma[i] + local_A[i, i])
    x[-1] = (local_b[-1] - local_A[-1, -2] * betta[-1]) / (local_A[-1, -1] +
local_A[-1, -2] * gamma[-1])
    for i in range(n-1, 0, -1):
        x[i-1] = gamma[i]*x[i] + betta[i]
    return x
```

3. Метод, минимизирующий вычислительные погрешности для квадратичных матриц общего вида.

Метод прогонки используется специально для трехдиагональных матриц. Метод на основе разложения Холецкого, который будет рассмотрен позже, подходит только положительно-определенных матриц. Методом, способным решать СЛАУ для квадратичных матриц общего вида, является метод Гаусса (без и с использованием частичного выбора главного элемента). Как было сказано ранее, частичный выбор главного элемента позволяет избежать усиления погрешностей округления в некоторых случаях. Таким образом, методом, минимизирующим вычислительные погрешности для квадратичных матриц общего вида, является метод Гаусса с частичным

выбором главного элемента. В дальнейшем, в рамках лабораторной работы будем называть данный метод «универсальным».

4. Реализация алгоритмов генерации матриц

Рассмотрим квадратные матрицы общего вида (будем сравнивать методы Гаусса без и с использованием частичного выбора главного элемента), трехдиагональные матрицы (будем сравнивать метод Гаусса с частичным выбором главного элемента и метод прогонки).

Определим Алгоритм генерации квадратных матриц общего вида. Так как метод Гаусса в общем случае является неустойчивым, важно не допустить, чтобы на главной диагонали оказались элементы равные нулю. Также матрицы должны оказаться невырожденными.

Функция для генерации квадратных матриц общего вида указана в листинге 3. Функция принимает на вход размерность квадратной матрицы. С помощью random.uniform из модуля numpy матрица заполняется случайными рациональными числами в интервале от -1 до 1. Далее высчитывается определитель матрицы и проверяются элементы главной диагонали. В случае если определитель равен нулю или хотя бы один из элементов главной диагонали равен нулю, матрица заполняется полностью с нуля (вероятность получить нежелательную матрицу очень мала). Равенство определителя нулю означает, что матрица вырожденная, и ее строки не являются линейно-независимыми, тогда СЛАУ будет иметь бесконечное количество решений или не будет иметь решений.

Листинг 3: генерация квадратных матриц общего вида

```
def generate_square_matrix(n):
    while True:
        A = np.random.uniform(low=-1, high=1, size=(n, n))
        diagonal = np.diag(A)
        if np.any(diagonal == 0):
            continue
        if np.linalg.det(A) != 0:
            return A
```

Функция для генерации трехдиагональных матриц представлена в листинге 4. Она также принимает на вход размерность квадратной матрицы. Алгоритм работы функции включает в себя заполнение матрицы нулями, а затем заполнение центральных диагоналей случайными числами в интервале от -1 до 1 с помощью random.uniform из модуля numpy. Также проверяется проверка на невырожденность.

Листинг 4: генерация трехдиагональных матриц

```
def generate_tridioganal_matrix(n):
    while True:
        A = np.zeros((n, n), dtype=dtype)
        for i in range(n - 1):
```

```
            A[i, i + 1] = np.random.uniform(low=-1, high=1)
            A[i, i] = np.random.uniform(low=-1, high=1)
            A[i + 1, i] = np.random.uniform(low=-1, high=1)
        A[-1, -1] = np.random.uniform(low=-1, high=1)
        if np.linalg.det(A) != 0:
            return A
```

5. Нахождение относительных погрешностей

Найдем относительные погрешности вычислений. Будем рассматривать такие классы матриц, как квадратные матрицы общего вида, трехдиагональные матрицы, положительно определенные матрицы. «Точным» методом будет выступать метод Гаусса с частичным выбором главного элемента. Методом «по умолчанию» для квадратных матриц общего вида будет выступать метод Гаусса без использования частичного выбора главного элемента. Методом «по умолчанию» для трехдиагональных матриц будет выступать метод прогонки, для положительно определенных матриц – метод на основе разложения Холецкого.

Формула для нахождения относительной погрешности (10):

$$\delta(\tilde{\boldsymbol{x}}) = \frac{\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|}{\|\boldsymbol{x}\|}, \tag{10}$$

где $\boldsymbol{x} = [x_1, \dots x_n] \in \mathbb{R}^n$ вектор решения, полученный «универсальным» методом, $\tilde{\boldsymbol{x}} = \boldsymbol{x} = [\tilde{x}_1, \dots \tilde{x}_n] \in \mathbb{R}^n$ вектор решения, полученный методом «по умолчанию».

Использовать для вычисления относительной погрешности среднеквадратичную норму (11) и супремум-норму (12).

$$\|\boldsymbol{x}\|_2 = \left(\sum_{i=1}^{n} x_i^2\right)^2 \tag{11}$$

$$\|\boldsymbol{x}\|_\infty = \max_{i \in [1,n]} |x_i| \tag{12}$$

Функция нахождения относительных погрешностей представлена в листинге 5. Данная функция принимает на вход некоторую функцию алгоритма генерации матриц, функцию метода решения СЛАУ «по умолчанию», количество матриц, которые надо сгенерировать. Будем генерировать 1000 матриц размерности 6 × 6 с элементами $|a_{ij}| < 1$ каждого класса. Вектор свободных членов $\boldsymbol{b} = [1, 1, 1, 1, 1, 1]^T$ для всех СЛАУ. В цикле для каждого СЛАУ вычисляется решение с помощью метода Гаусса с частичным выбором главного элемента и метода, переданного в качестве аргумента функции. Вычисляется относительная погрешность по формуле (10), при этом нормы векторов вычисляются с помощью функции norm модуля numpy.linalg.

Распределение относительных погрешностей для класса квадратных матриц общего вида (при подстановке в формулу (10) результат метода Гаусса с частичным выбором считается «универсальным», результат метода Гаусса без частичного выбора считается методом «по умолчанию» для матриц общего вида) отображено на гистограмме рис. 1 (среднеквадратичная норма), рис. 2 (супремум-норма). Распределение относительных погрешностей для трехдиагональных матриц (методом «по умолчанию» считается метод прогонки, универсальным все также метод Гаусса с частичным выбором главного элемента) для среднеквадратичной и супремум-нормы показано на рис. 3 и рис. 4.

Листинг 5: нахождение относительных погрешностей методов

```
def find_relative_errors(generate_method, solve_method, amount):
    b = np.array([1, 1, 1, 1, 1, 1], dtype=dtype)
    errors_L2 = np.array([], dtype=dtype)
    errors_L_inf = np.array([], dtype=dtype)
    conds = np.array([], dtype=dtype)
    spectral_radiuses = np.array([], dtype=dtype)
    eig_ratios = np.array([], dtype=dtype)
    for i in range(amount):
        A = generate_method(n=6)
        conds = np.append(conds, np.linalg.cond(A))
        spectral_radiuses = np.append(spectral_radiuses,
np.max(np.abs(np.linalg.eigvals(A))))
        eig_ratios = np.append(eig_ratios,
np.max(np.abs(np.linalg.eigvals(A))) / np.min(np.abs(np.linalg.eigvals(A))))
        x_default = solve_method(A, b)
        x_universal = gauss(A, b, True)
        errors_L2 = np.append(errors_L2, np.linalg.norm(x_default-
x_universal) / np.linalg.norm(x_universal))
        errors_L_inf = np.append(errors_L_inf, np.linalg.norm(x_default -
x_universal, ord=np.inf) / np.linalg.norm(x_universal, ord=np.inf))
    return errors_L2, errors_L_inf, conds, spectral_radiuses, eig_ratios
```

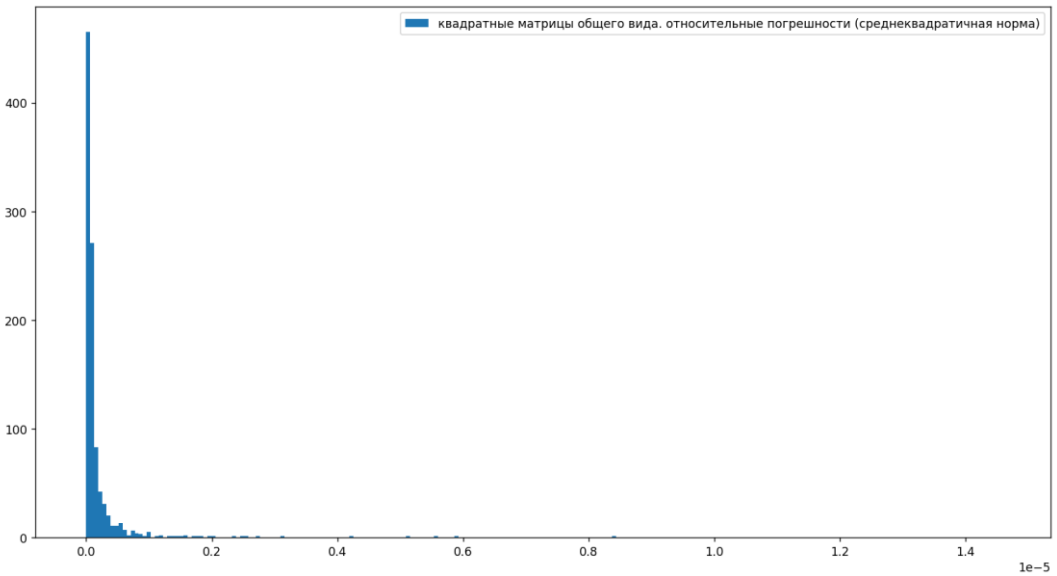

Рис. 1 – распределение относительных погрешностей для класса квадратных матриц общего вида (среднеквадратичная норма)

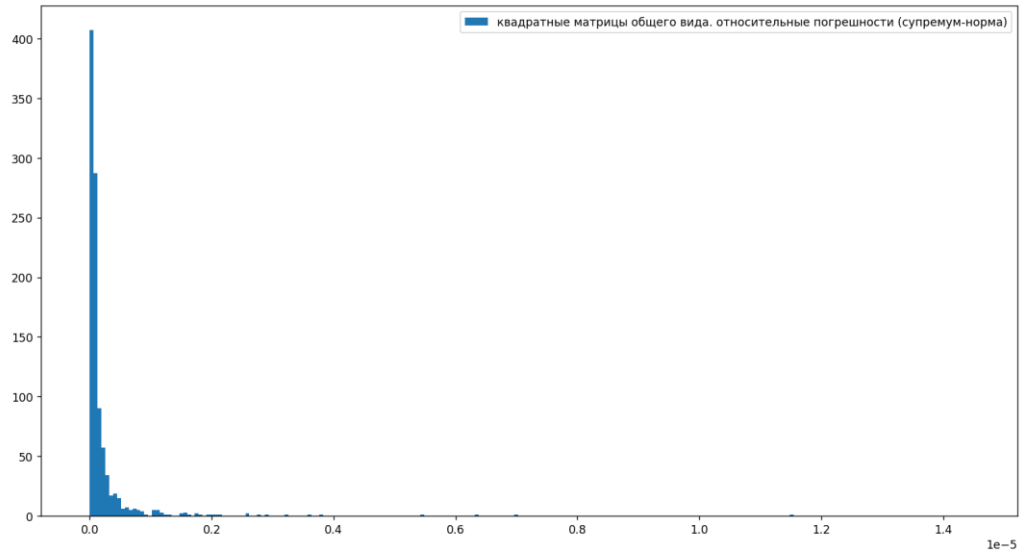

Рис. 2 – распределение относительных погрешностей для класса квадратных матриц общего вида (супремум-норма)

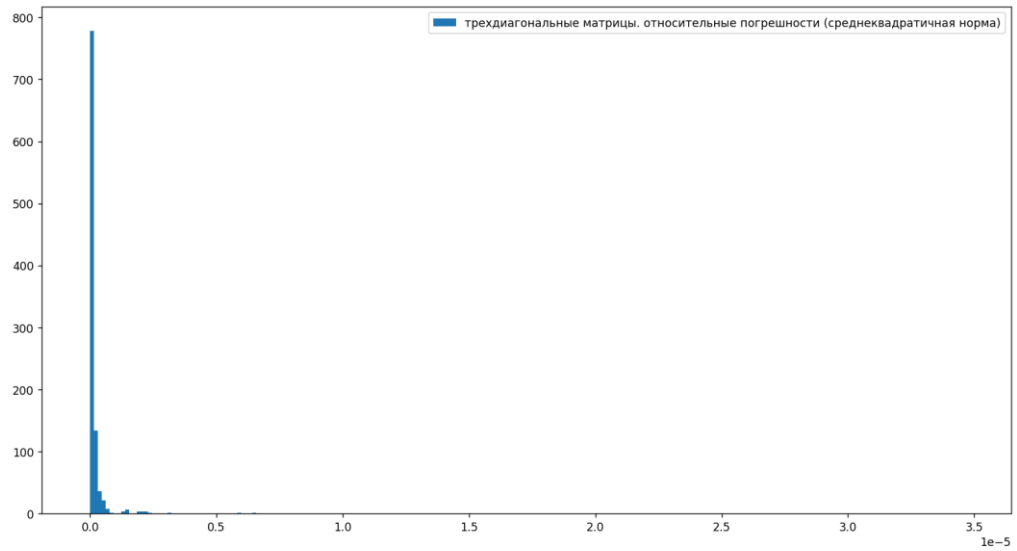

Рис. 3 – распределение относительных погрешностей для класса трехдиагональных матриц (среднеквадратичная норма)

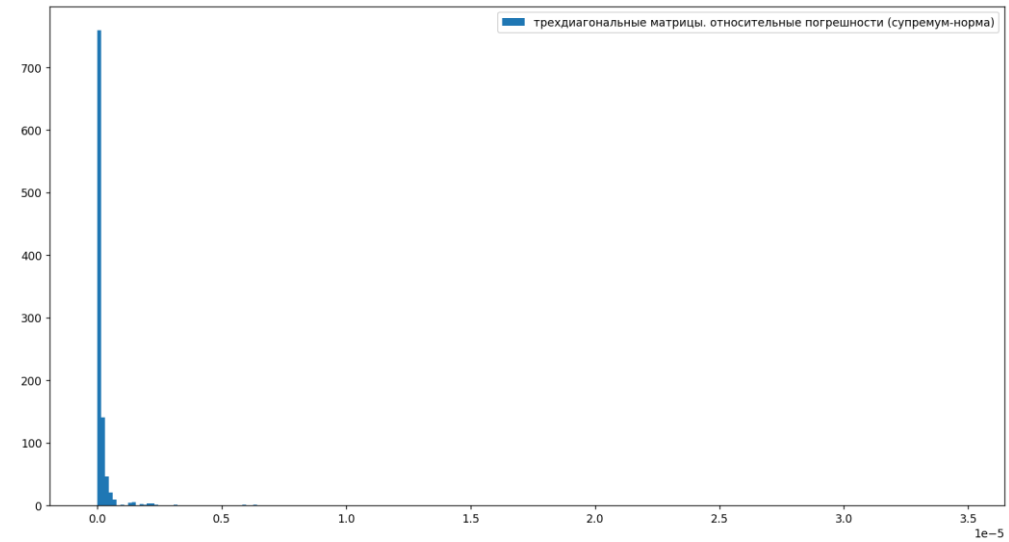

Рис. 4 – распределение относительных погрешностей для класса трехдиагональных матриц (супремум-норма)

Важно отметить, что рассмотренные методы, не являются устойчивыми. Частичный выбор главного элемента делает метод Гаусса более устойчивым, но не абсолютно устойчивым. Метод Гаусса без использования частичного выбора в общем случае не является устойчивым. Метод прогонки в общем случае не является устойчивым. Обоим методам для устойчивости необходимы матрицы со строгим диагональным преобладанием (13).

Для матрицы со строгим диагональным преобладанием верны соответствующие строгие неравенства:

$$|a_{ii}| > \sum_{\substack{j=1 \\ i \neq j}}^{n} |a_{ij}|, i = 1, \dots, n \quad (13)$$

Полученные гистограммы (рис. 1-4), матрицы для которых построены по ранее описанным алгоритмам и не предполагают строгого диагонального преобладания, демонстрируют некоторую неустойчивость методов. Небольшая часть результатов находится далеко от моды (мода на гистограмме представляет собой значение или интервал с наибольшей частотой или наибольшим количеством значений).

Продвинутая часть

1. Генерация положительно определенных матриц

Функция для генерации положительно определенных матриц представлена в листинге 6. Матрица заполняется случайными значениями, полученными с помощью ранее написанного алгоритма генерации случайных матриц общего вида, после чего складывается с собой же, но транспонированной, для соблюдения условия принадлежности значений интервалов диапазону от -1 до 1 полученная матрица умножается на 1/2. В результате получается симметричная матрица. Диагональ этой матрицы заполняется случайными положительными вещественными значениями от 0 до 1. При этом добавляется сумма по модулю элементов строки рассматриваемого диагонального элемента (значение суммы корректируется на значение диагонального элемента, заполненного ранее генератором матриц общего вида). Таким образом матрица становится со строгим диагональным преобладанием. Выполняется проверка. Если собственные числа матрицы больше нуля, то построенная матрица, действительно, является положительно определенной, иначе все выше описанное повторяется, пока не будет получена требуемая матрица.

Листинг 6: метод, использующий разложение Холецкого

```
def generate_pos_defined_matrix(n):
    while True:
        A = generate_square_matrix(n)
        A = (A + A.T) / 2
        for i in range(n):
            row_sum = sum(np.abs(A[i, j]) for j in range(n)) - np.abs(A[i,
i])
            A[i, i] = np.random.uniform(low=0, high=1) + row_sum
        if np.all(np.linalg.eigvals(A) > 0):
            return A
```

2. Решение СЛАУ с помощью разложения Холецкого

Матрица является положительно определенной, тогда и только тогда, когда существует разложение Холецкого для этой матрицы.

Разложение Холецкого имеет вид:

$$\boldsymbol{A = LL^T},$$

где $\boldsymbol{L}$ – нижняя треугольная матрица, $\boldsymbol{L^T}$ – транспонированная матрица $\boldsymbol{L}$.

Для матрицы 3×3 разложение Холецкого имеет вид:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix} \begin{bmatrix} l_{11} & l_{21} & l_{31} \\ 0 & l_{22} & l_{32} \\ 0 & 0 & l_{33} \end{bmatrix}$$

Общие формулы для расчета коэффициентов l имеют вид (14) и (15):

$$l_{ii} = \sqrt{a_{ii} - \sum_{j=1}^{i-1} l^2{}_{ij}}, i = 1, \dots, n \quad (14)$$

$$l_{ij} = \frac{1}{l_{jj}} \left(a_{ij} - \sum_{k=1}^{j-1} l_{ik} l_{jk} \right), j < i \quad (15)$$

Получив разложение Холецкого требуется решить СЛАУ (16), а затем (17):

$$\boldsymbol{Ly = b} \quad (16)$$

$$\boldsymbol{L^T x = y} \quad (17)$$

Итоговая функция для решения СЛАУ на основе разложения Холецкого представлена в листинге 7. В данной функции сначала находятся матрица $\boldsymbol{L}$ по формулам (14) и (15), находится $\boldsymbol{L^T}$ путем транспонирования $\boldsymbol{L}$ с помощью модуля numpy. Решаются СЛАУ (16) и (17) при помощи обратного хода метода Гаусса.

Листинг 7: функция решения СЛАУ на основе разложения Холецкого

```
def cholesky(A, b):
    n = len(A)
    L = np.zeros((n, n), dtype=dtype)
    L[0, 0] = np.sqrt(A[0, 0])
    for i in range(n):
        for j in range(0, i):
            L[i, j] = 1 / L[j, j] * (A[i, j] - sum(L[i, k] * L[j, k] for k in
```

```
range(j)))
        L[i, i] = np.sqrt(A[i, i] - sum(L[i, j] * L[i, j] for j in range(i)))
    L_T = np.transpose(L)
    y = np.zeros(n)
    y[0] = b[0] / L[0, 0]
    for i in range(n-1):
        y[i+1] = (b[i+1] - sum(L[i+1, j] * y[j] for j in range(i+1))) /
L[i+1, i+1]
    x = np.zeros(n)
    for i in range(n - 1, -1, -1):
        x[i] = (y[i] - np.dot(L_T[i, i + 1:], x[i + 1:])) / L_T[i, i]
    return x
```

3. Нахождение относительных погрешностей (метод Холецкого)

В пункте 5 базовой части уже были описаны формулы для нахождения относительной погрешности и функция, реализующая это.

Гистограммы распределений относительных погрешностей показаны на рис. 5-6 (метод «по умолчанию» - метод на основе разложения Холецкого, «универсальный» метод – метод Гаусса с частичным выбором главного элемента). Поскольку генерируются положительно определенные матрицы, которые являются матрицами со строгим диагональным преобладанием, метод показывает хорошую вычислительную устойчивость.

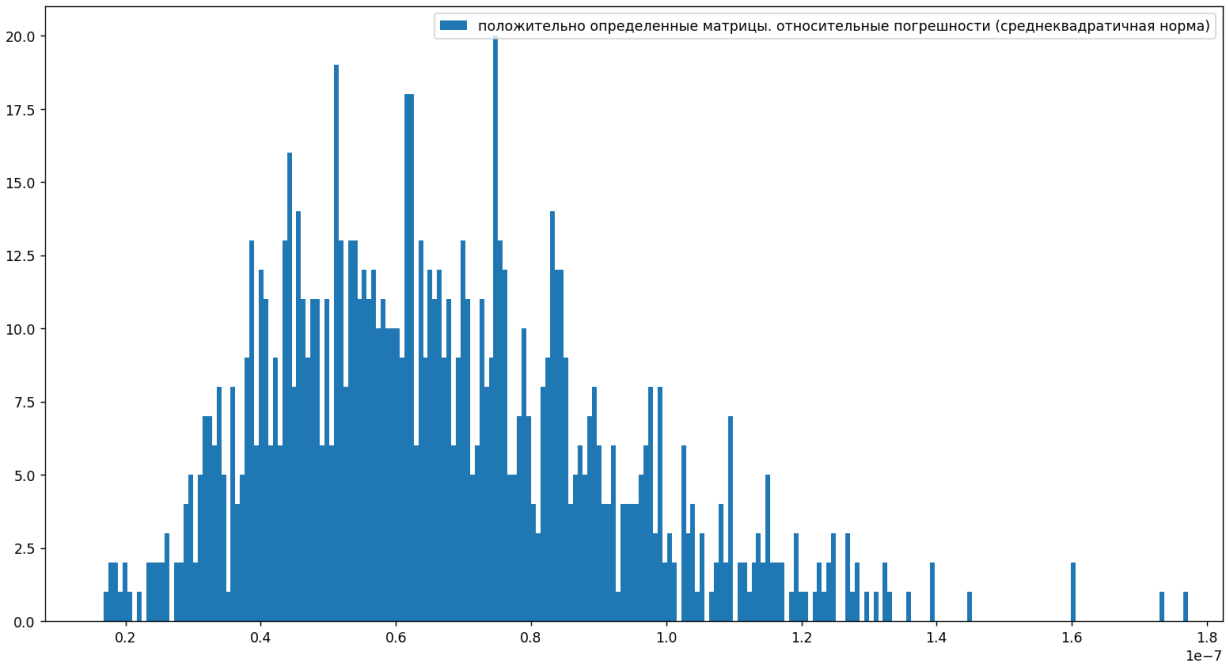

Рис. 5 – распределение относительных погрешностей для класса положительно-определенных матриц (среднеквадратичная норма)

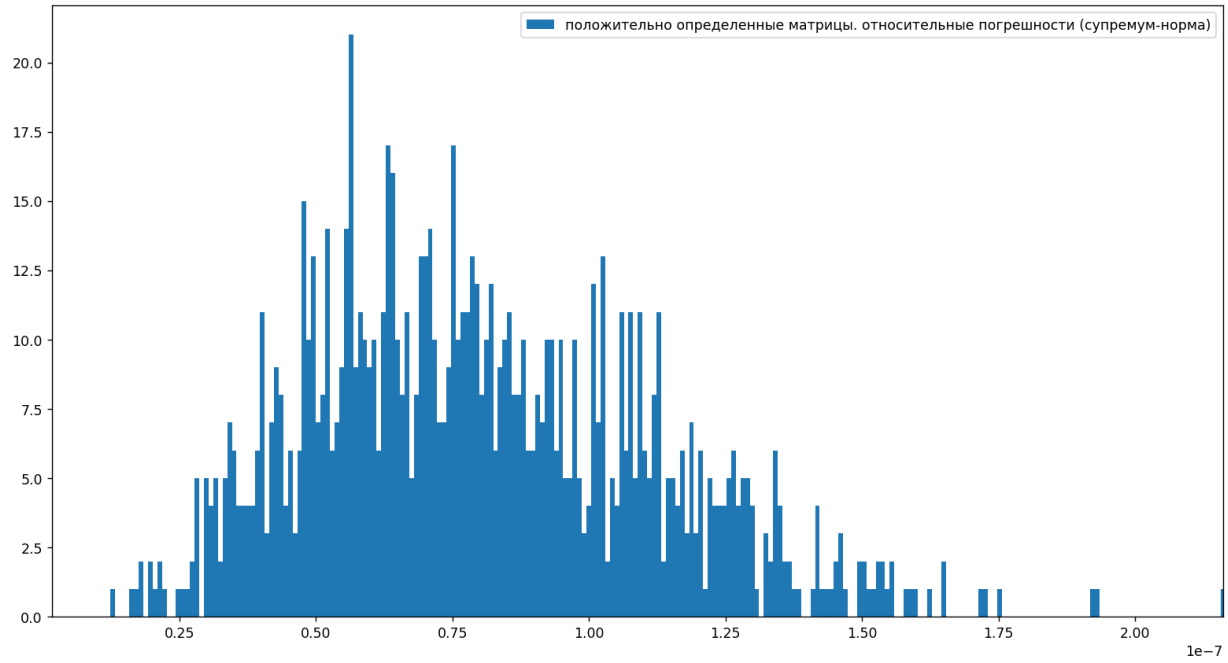

Рис. 6 – распределение относительных погрешностей для класса положительно-определенных матриц (супремум-норма)

4. Распределение спектральных радиусов и чисел обусловленности

В функции из листинга 5, рассмотренной ранее, также присутствует вычисление спектральных радиусов и чисел обусловленности матриц. Так как матрицы генерируются в этой функции, рациональным будет после каждой генерации рассчитывать сразу все требуемые значения, чтобы не хранить листы, состоящие из большого количества матриц.

Спектральный радиус матрицы — это наибольшее по модулю число из ее собственных чисел. Вычислить собственные числа можно решив характеристическое уравнение:

$$|\boldsymbol{A} - \lambda \boldsymbol{E}| = \boldsymbol{0},$$

где $\boldsymbol{A}$ – исходная матрица, для которой ищутся собственные числа, λ искомые собственные числа, $\boldsymbol{E}$ – единичная матрица, $|\boldsymbol{A} - \lambda \boldsymbol{E}|$ – определитель результирующей матрицы.

Число обусловленности матрицы характеризует устойчивость решения СЛАУ к малым возмущениям. Его можно вычислить следующим образом:

$$K(\boldsymbol{A}) = \|\boldsymbol{A}\| \cdot \left\|\boldsymbol{A}^{-1}\right\|,$$

где $\boldsymbol{A}$ – исходная матрица, $K(\boldsymbol{A})$ – ее число обусловленности.

Построенные гистограммы распределения спектральных радиусов и чисел обусловленности матриц, которые были сгенерированы алгоритмом для получения квадратных матриц общего вида, показаны на рис. 7 и 8, алгоритмом для получения трехдиагональных матриц – на рис. 9 и 10, алгоритмом для получения положительно определенных матриц – на рис. 11 и 12.

В функции (листинг 5) спектральных радиусы и числа обусловленности рассчитываются с помощью библиотечных eigvals и cond модуля numpy.linalg.

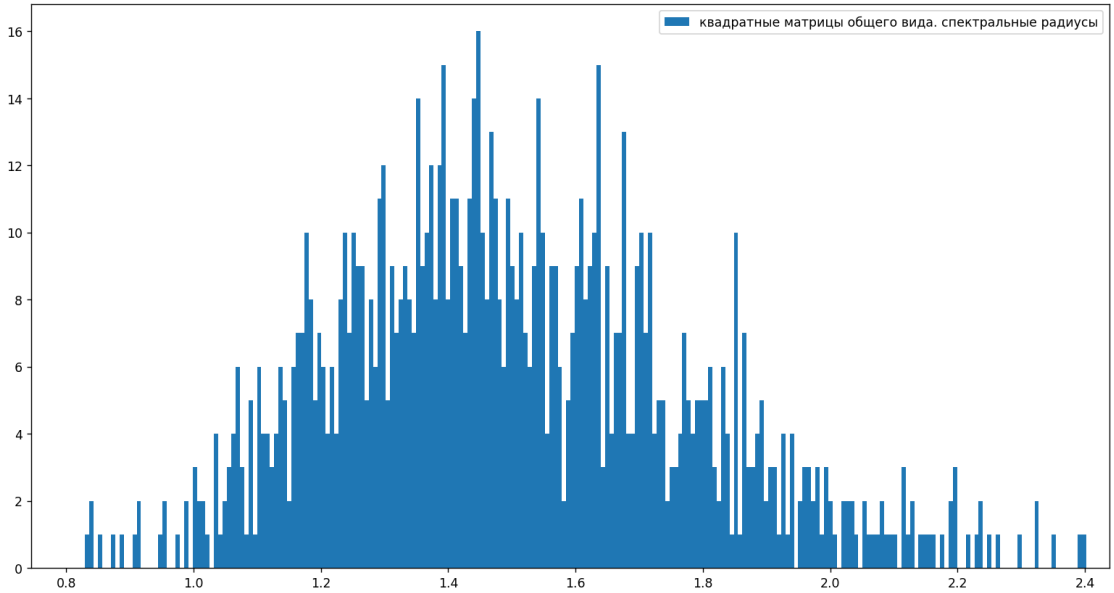

Рис. 7 – распределение спектральных радиусов для сгенерированных матриц общего вида

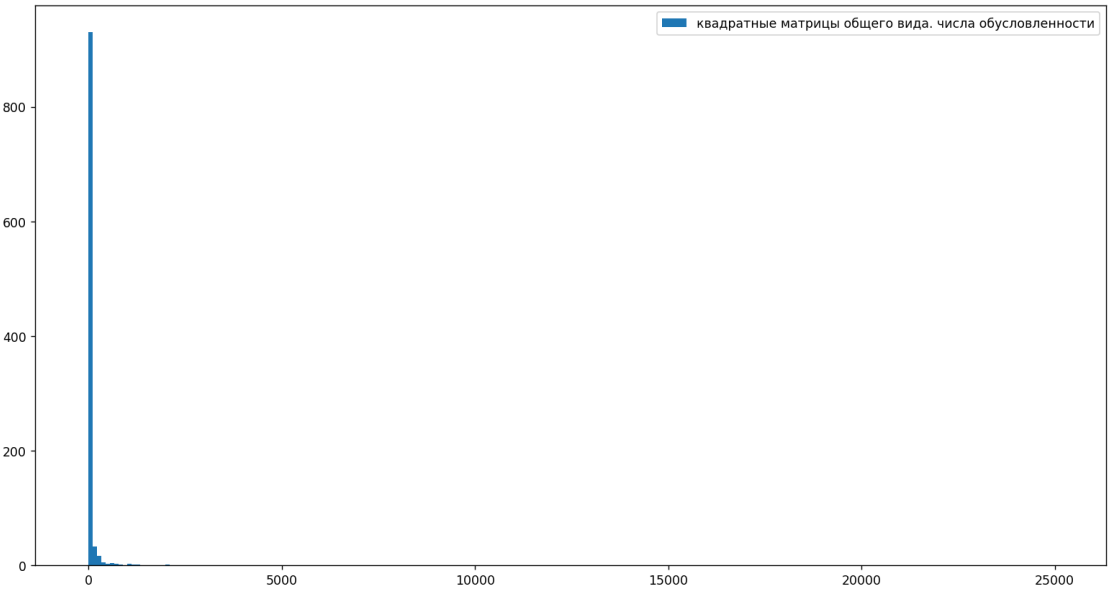

Рис. 8 – распределение чисел обусловленности для сгенерированных матриц общего вида

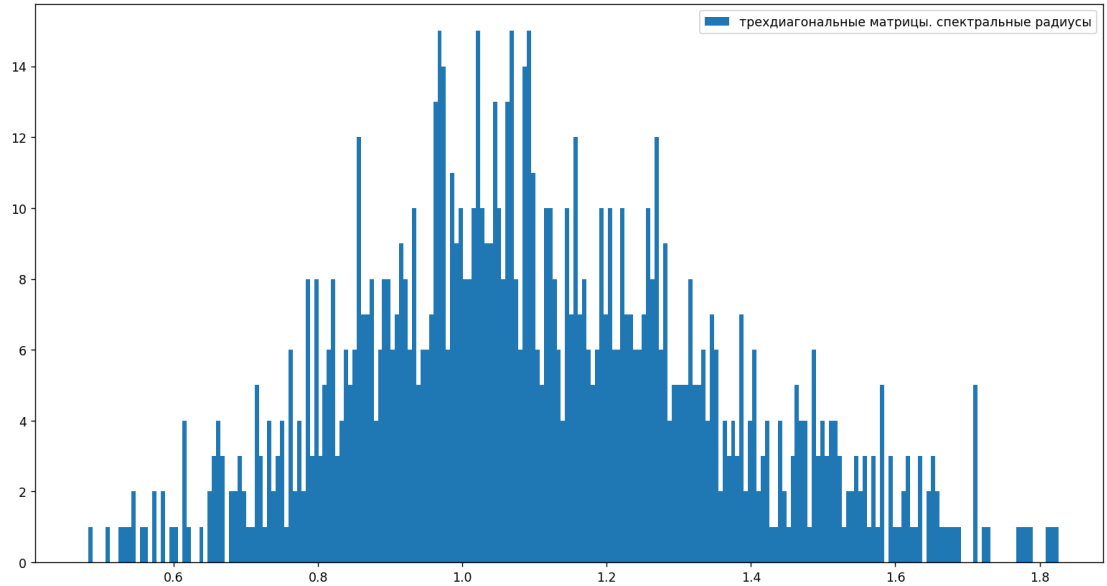

Рис. 9 – распределение спектральных радиусов для сгенерированных трехдиагональных матриц

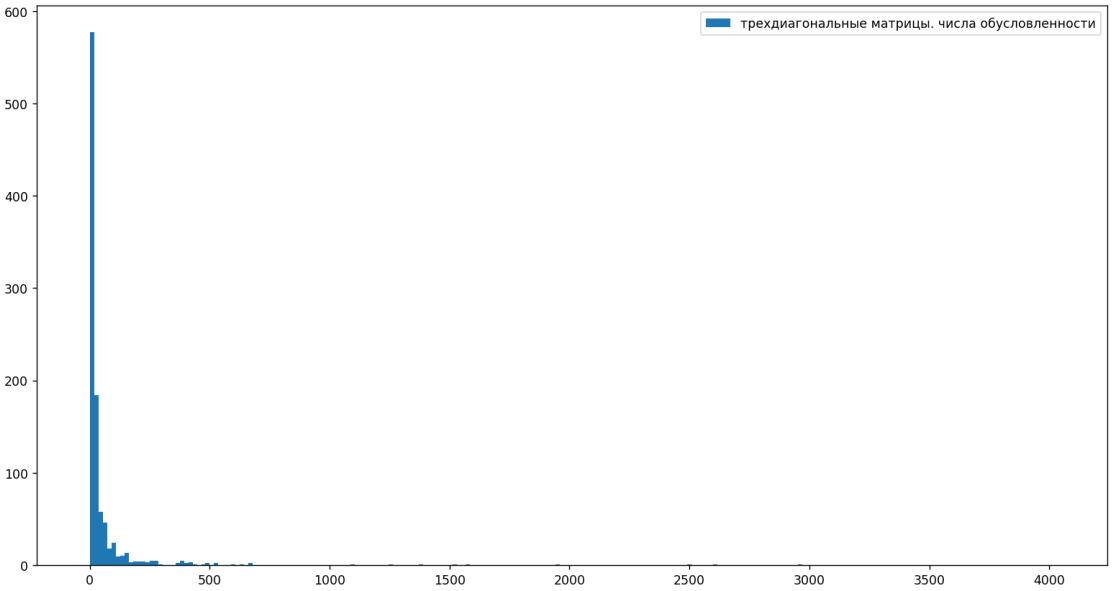

Рис. 10 – распределение чисел обусловленности для сгенерированных трехдиагональных матриц

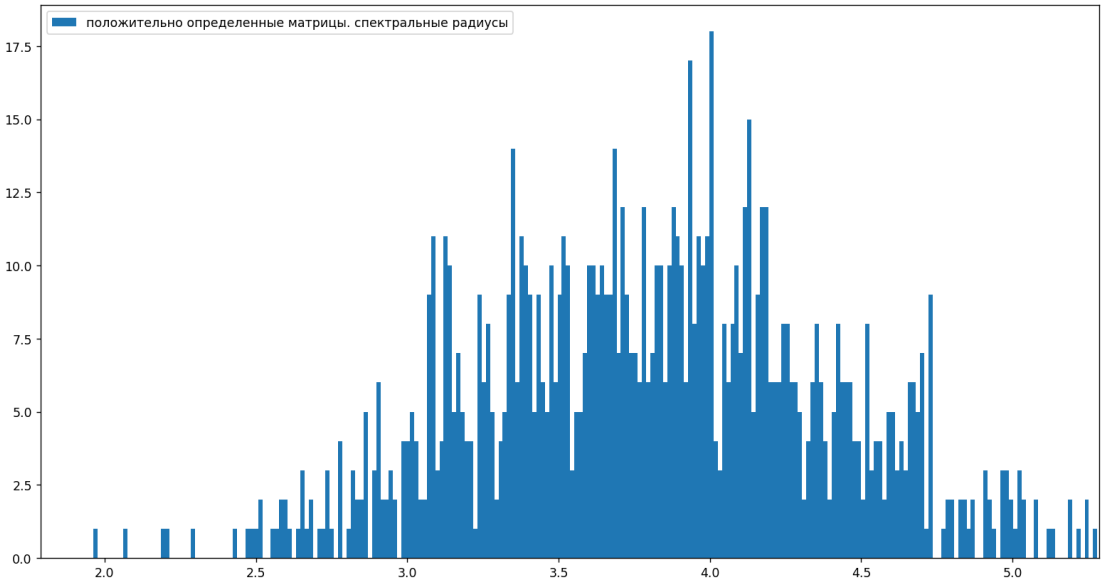

Рис. 11 – распределение спектральных радиусов для сгенерированных положительно определенных матриц

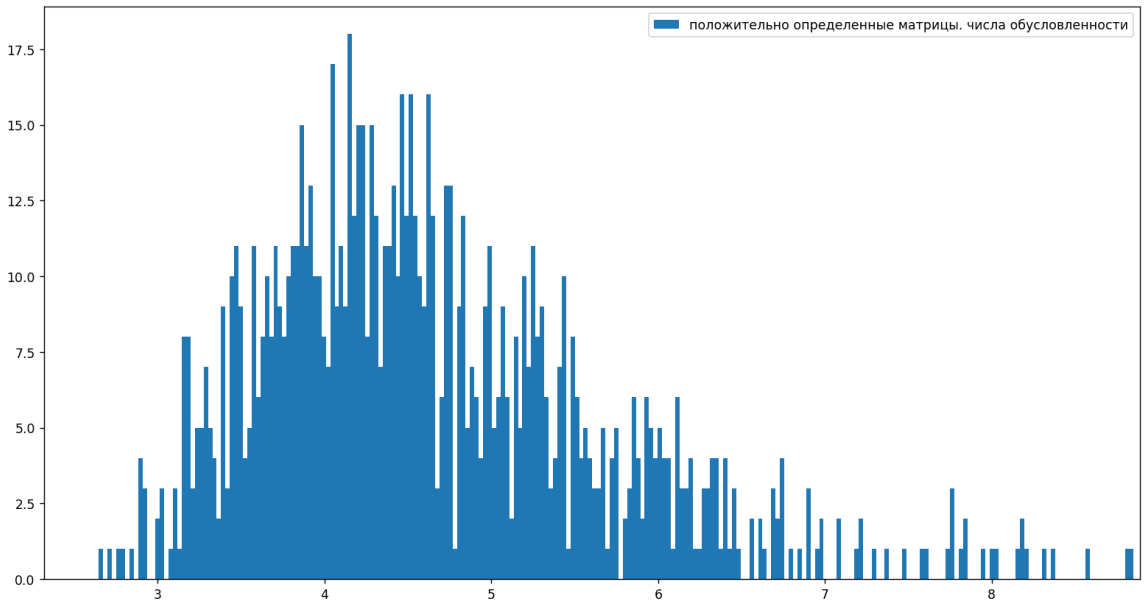

Рис. 12 – распределение чисел обусловленности для сгенерированных положительно определенных матриц

5. Влияние спектрального радиуса на устойчивость алгоритмов

Из полученных гистограмм можно сделать вывод, что нет зависимости между спектральным радиусом и вычислительной устойчивостью алгоритмов решения с помощью прямых методов. Известно, что в случае использования итерационных методов решения СЛАУ, если спектральный радиус матрицы меньше единицы, то итерационный метод устойчивый и сходится к определенному решению. Однако, для прямых методов спектральный радиус матрицы не является столь важным.

6. Влияние отношения максимального к минимального по модулю собственному числу на устойчивость алгоритмов.

Получим гистограммы распределений отношений собственных чисел для ранее сгенерированных матриц. Приведем гистограмму для ранее сгенерированных трехдиагональных матриц (рис. 13). Распределение на данной гистограмме очень похоже на распределение погрешностей на рис. 3 и рис. 4.

Изменим алгоритм генерации трехдиагональных матриц, таким образом, чтобы строились матрицы со строгим диагональным преобладанием. Получим распределение отношений собственных чисел, указанное на рис. 15. Распределение относительных погрешностей для трехдиагональных матриц со строгим диагональным преобладанием показано на рис. 14. Как можно заметить, распределение относительных погрешностей все также схоже с распределением отношений собственных чисел (рис. 14 и рис. 15). При этом, чем более схожи по модулю максимальное и минимальное собственное число, тем ниже получаемая относительная погрешность и выше устойчивость.

На рис. 16 представлена гистограмма распределений чисел обусловленности для трехдиагональных матриц со строгим диагональным преобладанием. Данное распределение очень похоже с распределением на рис. 15 для чисел обусловленности для того же набора матриц. Это свойственно и наборам матриц других классов.

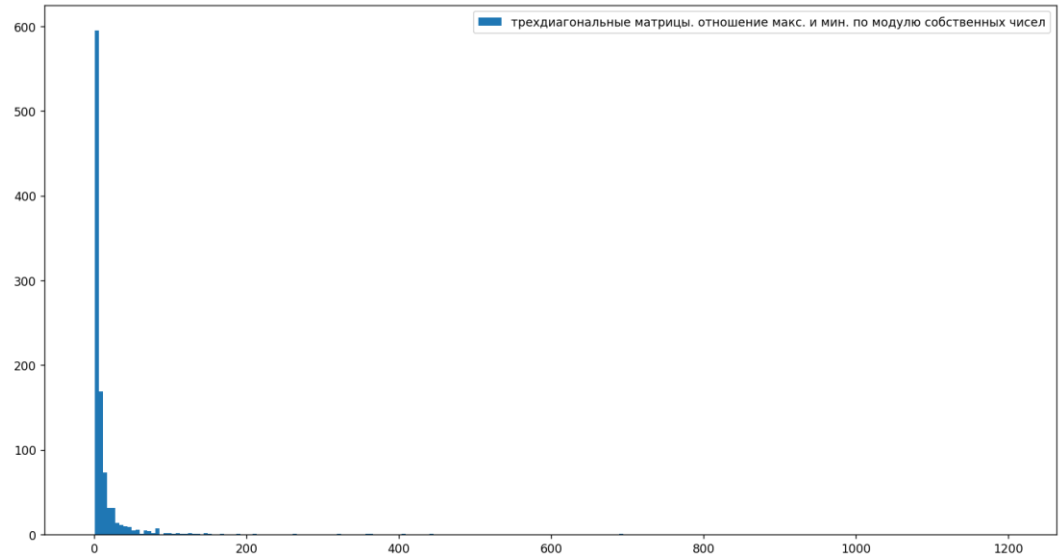

Рис. 13 - распределений отношений собственных чисел для ранее сгенерированных трехдиагональных матриц

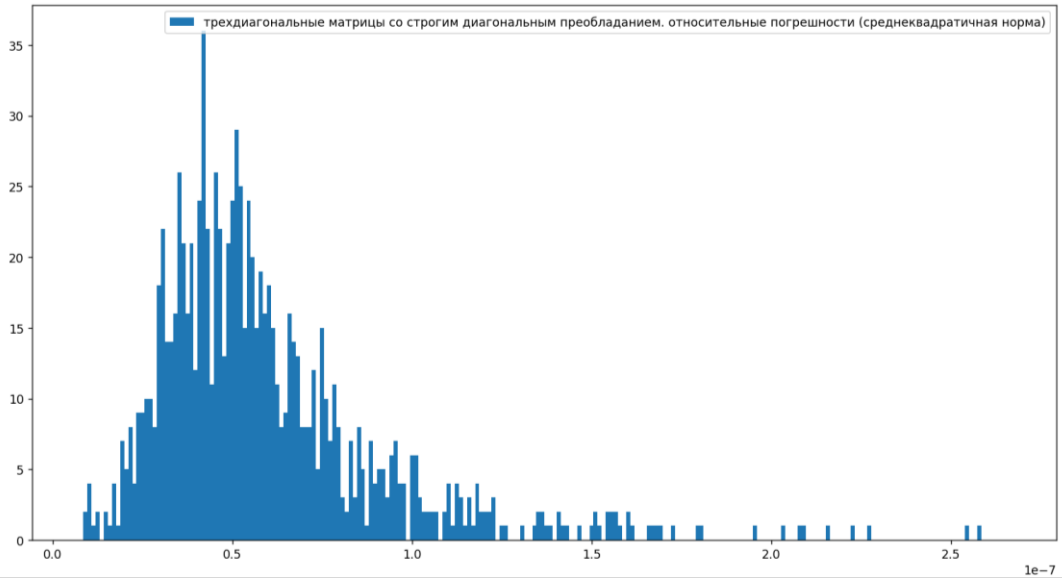

Рис. 14 - распределений относительных погрешностей для трехдиагональных матриц со строгим диагональным преобладанием

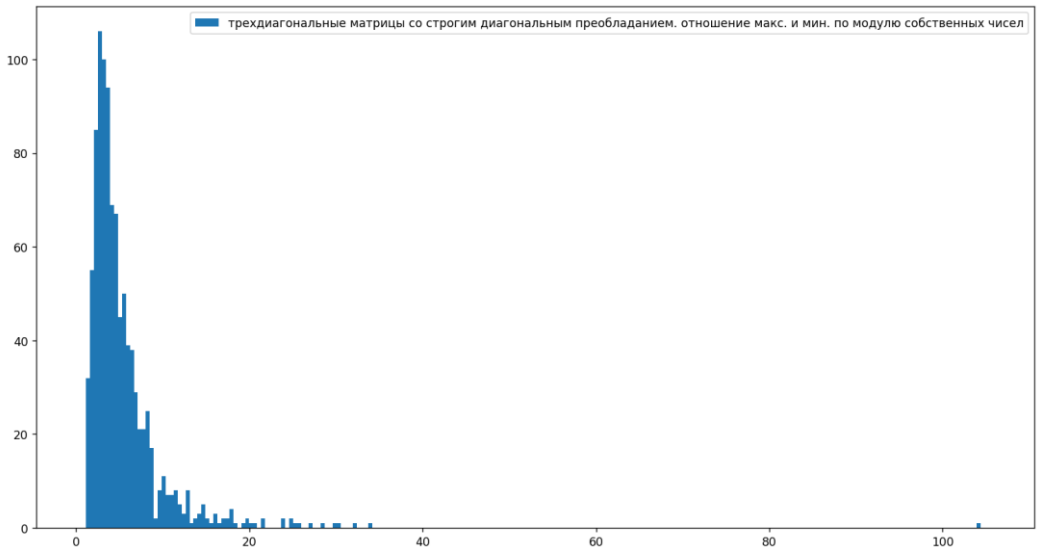

Рис. 15 - распределений отношений собственных чисел для трехдиагональных матриц со строгим диагональным преобладанием

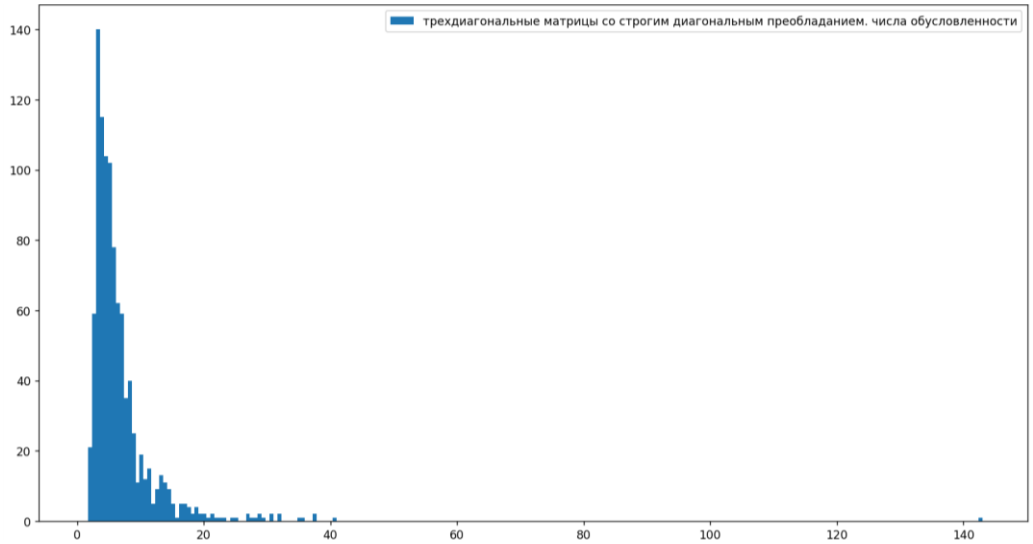

Рис. 16 - распределений чисел обусловленности для трехдиагональных матриц со строгим диагональным преобладанием

7. Влияние числа обусловленности на устойчивость

Число обусловленности матрицы характеризует устойчивость решения СЛАУ к малым возмущениям. Чем меньше число обусловленности, тем меньше будет относительная погрешность ($K(\boldsymbol{A}) \geq 1$). Слишком большое число обусловленности, в свою очередь, приводит к дестабилизации вычислительных погрешностей. По полученным гистограммам видно, что распределение чисел обусловленности имеет сходства с распределением величин относительной погрешности.

Используем изменения, сделанные в алгоритме в предыдущем пункте, чтобы строились матрицы со строгим диагональным преобладанием. В результате получим следующее распределение относительных погрешностей и чисел обусловленности (рис. 14 и рис. 16). Числа обусловленности стали очень близки к 1. Можно заметить, что относительные погрешности

снизились, и их распределение также повторяет распределение чисел обусловленности (рис. 3 и 10 для ранее сгенерированных трехдиагональных матриц, рис. 14 и 16 для позднее сгенерированных трехдиагональных матриц с строгим диагональным преобладанием). Это доказывает влияние чисел обусловленности на устойчивость алгоритмов.

Заключение

Были реализованы некоторые прямые методы решения СЛАУ для различных классов матриц: метод Гаусса, являющийся универсальным, так как может решить СЛАУ, которая задается матрицей общего вида; метод прогонки для трехдиагональных матриц; метод, использующий разложение Холецкого, для положительно определенных матриц. Были построены гистограммы распределений относительных погрешностей реализованных методов, полученные на основе 1000 сгенерированных матриц СЛАУ, а также распределения чисел обусловленности и спектральных радиусов, отношений максимального по модулю к минимальному модулю собственному числу, для этих же матриц.

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)