МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование)

02.03.03. Математическое обеспечение и администрирование информационных систем
(код и наименование направления подготовки, специальности)

Мобильные и сетевые технологии
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему «Сравнительный анализ алгоритмов нахождения собственных значений симметричных матриц большой размерности»

Студент С.А. Скоков
(И.О. Фамилия) (личная подпись)

Руководитель М.А. Тренина
(ученая степень, звание, И.О. Фамилия)

Консультант М.В.Дайнеко
(ученая степень, звание, И.О. Фамилия)

Тольятти 2021

Аннотация

Выпускная квалификационная работа посвящена сравнительному анализу алгоритмов нахождения собственных значений симметричных матриц большой размерности.

Ключевые слова: симметричная, матрица, собственные значения, алгоритмы.

При работе с динамическими системами и соответствующими системами линейных дифференциальных уравнений вычисление собственных значений матриц этих систем позволяет определить особенности их поведения во времени, а также оценить устойчивость такой системы. Поэтому нахождение и оценка спектральных свойств матриц – это задачи, часто возникающие перед специалистами во многих областях, от астрофизики до машиностроения.

Объектом исследования данной бакалаврской работы являются алгоритмы поиска собственных значений симметричных матриц большой размерности.

Предметом исследования является процесс решения задачи собственных значений симметричных матриц большой размерности.

Целью бакалаврской работы является сравнительный анализ методов нахождения собственных значений симметричных матриц большой размерности и их программная реализация.

В ходе выполнения выпускной работы было проведено исследование с применением разработанного на языке C++ приложения, а также сделаны выводы о работе алгоритмов.

Выпускная квалификационная работа представлена на 46 страницах, включает 13 иллюстраций, 7 таблиц, 48 формул, и список используемой литературы, состоящий из 20 источников.

Abstract

The present graduation work is devoted to the comparative analysis of algorithms for finding the eigenvalues of large dimension symmetric matrices.

When working with dynamical systems and the corresponding systems of linear differential equations, the knowledge of the eigenvalues of the obtained matrices allows us to determine the nature and features of the system behavior over a period of time and enables us to evaluate the stability of such a system.

The graduation work consists of 13 figures, 7 tables, 48 formulae and the list of 20 references including 5 foreign sources.

The object of this investigation is the algorithms for finding eigenvalues of the large dimension symmetric matrices.

The subject of this work is the problem solving process of the large dimension symmetric matrices eigenvalues.

The aim of the research is to conduct a comparative analysis of the methods for finding the large dimension symmetric matrices eigenvalues and to provide their software implementation.

The graduation work may be divided into several logically connected parts.

In the first part of the graduation work, we start with the statement of the problem and then logically pass over to its possible solutions.

In the second part of the research, we describe the process of developing a software application that will be used in some future experiments.

In the third part of the investigation, we present the results of the experiments.

Finally, the analysis of the algorithms' operating work is carried out based on the accuracy and the number of iterations with the different ranges of the matrix values.

Содержание

Введение

Собственные значения и соответствующие собственные вектора матрицы – это одни из важных понятий в линейной алгебре. Множество собственных значений, или так называемый спектр матрицы, характеризует некоторые свойства систем, полезные для проведения анализов.

При работе с динамическими системами и соответствующими системами линейных дифференциальных уравнений вычисление собственных значений матриц этих систем позволяет определить особенности их поведения во времени, а также оценить устойчивость такой системы. Поэтому нахождение и оценка спектральных свойств матриц – это задачи, часто возникающие перед специалистами во многих областях, от астрофизики до машиностроения.

Соответственно, актуальность проблемы, рассматриваемой в данной выпускной квалификационной работе тесно связана с возникновением подобных проблем в физики и механике. Для матриц больших размерностей эта задача связана с огромным количеством алгебраический действий, что говорит о большой вычислительной сложности, и приводит к большой трудоёмкости решения даже для современных компьютеров.

Целью бакалаврской работы является сравнительный анализ методов нахождения собственных значений симметричных матриц большой размерности и их программная реализация.

Объектом исследования данной бакалаврской работы являются алгоритмы поиска собственных значений симметричных матриц большой размерности.

Предметом исследования является процесс решения задачи собственных значений симметричных матриц большой размерности.

Задачи, которые необходимо решить для достижения указанной цели, это:

1) выбрать методы для исследования, рассмотрев математическое описание задачи и существующие методы решения;
2) реализовать вычисление собственных значений симметричных матриц большой размерности несколькими различными методами на языке C++;
3) провести вычислительные эксперименты и собрать данные для анализа;
4) провести сравнительных анализ результатов и сделать выводы.

Данная выпускная квалификационная работа состоит из введения, трёх глав и заключения:

В первой главе представлено математическое описание решения задачи нахождения собственных значений матрицы, рассмотрены точные (Метод Данилевского и метод Леверрье-Фадеева) и итерационные алгоритмы (Степенной метод, метод скалярный произведений, метод вращений Якоби, метод QR) решения этой задачи, выбраны методы для дальнейшей реализации.

Во второй главе описана программная реализация алгоритмов на языке C++.

В третьей главе представлены результаты вычислительных экспериментов и их сравнительных анализ.

1 Обзор методов решения задачи нахождения собственных значений симметричных матриц большой размерности

1.1 Постановка задачи нахождения собственных значений

Следует начать математическое описание задачи с определения собственных значений. Данное определение проще сформулировать через собственные вектора. Вектор, умножение матрицы на который в результате даст тот же вектор, но умноженный на определенную скалярную величину называется собственным вектором матрицы, при этом данная скалярная величина и будет искомым собственным значением. То есть, если имеется некая квадратная матрица A с размерностью $n \times n$, то её собственное значение будет удовлетворять равенству (1)

$$A\vec{x} = \lambda\vec{x} \quad (1)$$

где $\vec{x}$ – собственные вектор матрицы A;

λ – собственное значение.

Также следует отметить, что существует условие существования собственных значений, которое выражается в требовании (2)

$$|A - \lambda E| = 0, \quad (2)$$

где E – единичная матрица.

Говоря о собственных значениях, следует помнить, что данные значения могут быть и комплексными, и возникают в случаях, когда рассматриваемая матрица не является симметричной. В такой ситуации собственные значения матрицы являются комплексно-сопряженными числа вида (3)

$$\lambda = \alpha - i\beta \qquad (3)$$

Если же матрица симметрична, всё её значения гарантированно являются вещественными и при этом все собственные вектора будут ортогональны между собой.

Классические методы нахождения спектральный свойств матрицы сводятся к решению её характеристического уравнения вида (4)

$$\Delta(\lambda) = det(A - \lambda E) = a_n (\lambda - \lambda_1)^{n_1} (\lambda - \lambda_2)^{n_2} \dots (\lambda - \lambda_k)^{n_k}, \qquad (4)$$

где $\lambda_1, \lambda_2, \dots, \lambda_k$ – корни многочлена кратности $n_1, n_2, \dots, n_k$.

Такие методы получили название точных (или прямых) и имеют определенные недостатки. Дело в том, что при увеличении размерности матрицы растёт и сложность вычисления корней многочлена (4), и при больших размерностях становится слишком трудно решаемой задачей.

Поэтому чаще используются итерационные методы, которые находят приближение к собственным значениям, не используя характеристическое уравнение. Итерационные методы обладают достаточно высокой точностью и упрощают вычислительный процесс.

Таким образом глобально все методы можно разделить на два типа:

1) точные, основанные на нахождении и решении характеристического многочлена (4) (методы Данилевского, Леверрье-Фадеева и др.);
2) итерационные, реализующий поиск значений путем множества итераций, приближающий значения к истинным с некоторой точностью (степенной метод, метод QR, метод вращений Якоби и т.д.).

Помимо этого, саму задачу поиска собственных значений можно разделить на два типа, в зависимости от цели решения задачи:

1) полная проблема собственных значений, то есть отыскание всех имеющихся собственных чисел;

2) частичная проблема собственных значений, то есть нахождение только некоторых собственных чисел (зачастую максимальное либо минимальное).

Далее подробно рассмотрим некоторые алгоритмы решения данной задачи.

1.2 Точные алгоритмы нахождения собственных значений

1.2.1 Метод А.М. Данилевского

Метод А. М. Данилевского – точный метод решения полной задачи нахождения собственных значений матриц, основанный на поиске и решении характеристического уравнения. Суть метода А.М. Данилевского заключается в преобразовании исходной квадратной матрицы A (5)

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix} \quad (5)$$

в подобную ей матрицу Фробениуса P (6)

$$P = \begin{pmatrix} p_1 & p_2 & \dots & p_{n-1} & p_n \\ 1 & 0 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1 & 0 \end{pmatrix} \quad (6)$$

с помощью матрицы подобия B по формуле (7)

$$P = B^{-1}AB \quad (7)$$

$$b_{n-1,j} = -\frac{a_{n,j}}{a_{n,n-1}} \quad (8)$$

$$b_{n-1,n-1} = \frac{1}{a_{n,n-1}} \quad (9)$$

где $b_{n-1,j}$ – элементы матрицы подобия B, стоящие на n-1 строке,

$a_{n,j}$ – элементы матрицы смежности графа, стоящие на n строке.

Для матрицы P характеристический многочлен может быть найден, если последовательно раскладываться определитель $det|P - \lambda E|$ по элементам первого столбца. В результате получается:

$$det|P - \lambda E| = (-1)^n[\lambda^n - P_1\lambda^{n-1}, \dots, -P_n\lambda - P_n] = (-1)^n P_n(\lambda) \quad (10)$$

Из последнего соотношения видно, что элементы 1-й строки матрицы в форме Фробениуса P являются коэффициентами её собственного многочлена и, следовательно, собственного многочлена исходной матрицы A.

Решив полученное уравнение (10), $P_n(\lambda) = 0$ найдём собственные значения матрицы A. Далее, матрица B, полученная в ходе решения методом Данилевского, может использоваться при нахождении собственных векторов матрицы A.

1.2.2 Метод Леверрье-Фаддеева

Метод Леверрье основан на применении формул Ньютона для сумм степеней корней алгебраического уравнения.

Пусть

$$D(\lambda) = \lambda^n - q_1\lambda^{n-1} - q_2\lambda^{n-2} - \dots - q_n \quad (11)$$

характеристический многочлен матрицы A и $\lambda_1, \lambda_2, \dots, \lambda_n$ – это полная совокупность его корней, то есть каждый корень повторяется столько раз, какова его кратность.

Если обозначим $S_k = \lambda_1^k + \lambda_2^k + \cdots \lambda_n^k$, где $k = 1,2, \dots, n$. Тогда получаем, что при $k \leq n$, справедливы формулы Ньютона (12):

$$kq_k = S_k - q_1 S_{k-1} - \cdots - q_{k-1} S_1, \text{где } k = 1,2,\dots,n \qquad (12)$$

Если все числа S_k известны, то, решив полученную реккурентную систему (13) можно найти необходимые для решения уравнения (12) коэффициенты p_k:

$$\begin{cases} q_1 = S_1 \\ q_2 = -\frac{1}{2}(S_2 - q_1 S_1) \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \\ q_n = -\frac{1}{n}(S_n - q_1 S_{n-1} - \cdots - q_{n-1} S_1 \end{cases} \qquad (13)$$

Также при этом числа $\lambda_i^k, i = 1,2,\dots,n$ и будут являться собственными значения матрицы A^k, а $S_k = SpA^k$, где $k = 1,2,\dots,n$ и SpA^k – это будет сумма элементов главной диагонали матрицы A^k, степени $A^k = A^{k-1}A$ находятся непосредственным перемножением.

Таким образом, общая схема раскладывания определителя по методу Леверрье состоит в следующем:

1) вычисляются A^k, где $k = 1,2,\dots,n$ – степени данной матрицы A;
2) находятся соответствующие S_k – суммы элементов главных диагоналей матриц A^k;
3) по формулам (13) определяются искомые коэффициенты q_k, где $k = 1,2,\dots,n$.

В дальнейшем Д.К. Фадеев предложил видоизменить представленный метод Леверрье. В результате метод Леверрье-Фадеева по сравнению с методом Леверрье более прост в процессе вычисления коэффициентов характеристического уравнения, а также позволяет определить ещё и обратную матрицы и собственные вектора.

Основная суть всех изменений заключается в том, что вместо использования последовательности A^k, где $k = 1,2,\dots,n$, в данной

модификации метода используется последовательность $A_1, A_2, \dots, A_n$, построенной по схеме, приведенной на рисунке 1.

$$
\begin{array}{lll}
A_1 = A, & SpA_1 = q_1, & B_1 = A_1 - q_1 \cdot E \\
A_2 = AB_1, & \dfrac{SpA_2}{2} = q_2, & B_2 = A_2 - q_2 \cdot E \\
\dots\dots\dots & \dots\dots\dots & \dots\dots\dots \\
A_{n-1} = AB_{n-2} & \dfrac{SpA_{n-1}}{n-1} = q_{n-1} & B_{n-1} = A_{n-1} - q_{n-1} \cdot E \\
A_n = AB_{n-1} & \dfrac{SpA_n}{n} = q_n & B_n = A_n - q_n \cdot E,
\end{array}
$$

Рисунок 1 – схема вычисления по методу Леверрье-Фадеева

На рисунке 1 E – это единичная матрица того же порядка, что и матрица A. Таким образом, коэффициенты характеристического уравнения можно определить, используя данную схему.

Описанные выше методы дают возможность точно рассчитать собственные значения матриц, но при исследовании матриц большой размерности становится ясно, что данные методы неприменимы, так как характеристическое уравнение данных матриц будет обладать слишком большой вычислительной сложностью, поэтому в данном случае прибегают к итерационным методам. Эти методы находят приближение к собственным значениям, не используя характеристическое уравнение. Они обладают достаточно высокой точностью и упрощают вычислительный процесс, поэтому рассмотрим их подробнее.

1.3 Итерационные методы нахождения собственных значений

1.3.1 Степенной метод

Степенной метод обычно используется для приближенного вычисления крайних собственных значений, в данном случае рассмотрим нахождение максимального по модулю числа. Рассмотрим данный метод подробнее.

Если матрица A имеет размеры $n \times n$, то у неё обязательно имеются n собственных чисел, при этом данные собственные числа не обязательно должны быть различны, и, предположим, что матрица A имеет n собственных векторов и полную систему из собственных векторов. Тогда это даёт возможность разложить любой вектор в линейную комбинацию собственных векторов (14). Далее, умножая уравнение (1) на исходную матрицу, получим выражение (15), проведя данное умножение k раз получаем равенство (16) и выполнив его преобразования получим (17):

$$x = C_1 a_1 + C_2 a_2 + \cdots + C_n a_n = \sum_{i=0}^{n} C_i a_i \,, \tag{14}$$

где $Aa_i = \lambda_i a_i \,, i = 1..n$ – собственные вектора.

$$Ax = \sum_{i=0}^{n} C_i A a_i = \sum_{i=0}^{n} C_i \lambda_i a_i \,; \tag{15}$$

$$A^k x = \sum_{i=0}^{n} C_i A^k a_i = \sum_{i=0}^{n} C_i \lambda_i^k a_i \,; \tag{16}$$

$$\begin{aligned} A^k x = \lambda_1^k \left(C_1 a_1 + C_1 \left(\frac{\lambda_2}{\lambda_1}\right)^k a_1 + \cdots + C_n \left(\frac{\lambda_n}{\lambda_1}\right)^k a_n \right) = \\ = \lambda_i^k \sum_{i=0}^{n} C_i \left(\frac{\lambda_i}{\lambda_1}\right)^k a_i \,; \end{aligned} \tag{17}$$

Таким образом, воспользовавшись всеми введёнными утверждениями и формулами можно сделать вывод, что данным метод решения частичной задачи нахождения собственного значения реализуется с помощью алгоритма:

1) выбирается или случайно генерируется вектор начального приближения;
2) по формулам (18) и (19) строится приближение вектора:

$$y_{n+1} = A\vec{x}_n\ ; \tag{18}$$

$$\vec{x}_{n+1} = a_{n+1}y_{n+1},; \tag{19}$$

где a_{n+1} не противоречит условию $|\vec{x}_{n+1}| = 1$;

3) Итерационно строим приближения, пока не будет достигнуто условие сходимости (20):

$$\|\vec{x}_{n+1} - \vec{x}_n\ \| \leq \varepsilon, \tag{20}$$

где ε – заданная точность;

4) при достижении условия сходимости $\vec{x}_{n+1}$ полученное приближение будет являться собственным вектором, используя который, можно найти собственное значение по формуле (21):

$$x^k Ax \approx \lambda_1 x^k x = \lambda_{1,} \tag{21}$$

поскольку $\|x\| = 1$.

1.3.2 Метод скалярный произведений

Для нахождения первого собственного значения λ_1 действительной матрицы A можно указать несколько иной итерационный процесс, являющийся иногда более выгодным. Метод основан на образовании скалярных произведений.

$$\left(A^k\vec{y}_0, A'^k\vec{y}_0\right) \text{ и } \left(A^{k-1}\vec{y}_0, A'^k\vec{y}_0\right), k = 1,2,\dots,n \quad (22)$$

где A' - матрица, транспонированная с матрицей A ,

$\vec{y}_0$- выбранный каким-либо образом начальный вектор.

Пусть A - действительная матрица и $\lambda_1, \lambda_2, \dots, \lambda_n$- ее собственные значения, которые предполагаются различными, причем

$$|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \cdots \geq |\lambda_n|. \quad (23)$$

Возьмем некоторый ненулевой вектор $\vec{y}_0$ и с помощью матрицы A построим последовательность итерации

$$\vec{y}_k = A\vec{y}_{k-1}, \qquad k = 1,2,\dots.n, \quad (24)$$

Для вектора $\vec{y}_0$ образуем также с помощью транспонированной матрицы A' вторую последовательность итерации

$$\vec{y}'_k = A'\vec{y}'_{k-1}, k = 1,2,\dots.n, \quad (25)$$

где $\vec{y}'_0 = \vec{y}_0$.

В пространстве E_n выберем два собственных базиса $\{x_j\}$ и $\{x'_j\}$ соответственно для матриц A и A', удовлетворяющих условиям биортонормировки: $\left(x_i, x_j\right) = \delta_{ij}$.в базисе $\{x'_j\}$ – через $b_1, \dots, b_n$, т. е.

$$\vec{y}_0 = a_1x_1 + \cdots + a_nx_n \text{ и } \vec{y}_0 = b_1x'_1 + \cdots + b_nx'_n ; \quad (26)$$

отсюда

$$\vec{y}_k = A^k y_0 = \sum_{i=1}^{n} a_j \lambda_l^k x_j \tag{27}$$

и

$$\vec{y'}_k = A'^k y_0 = \sum_{i=1}^{n} a_j \lambda'^k_l x'_j \tag{28}$$

Составим скалярное произведение

$$(\vec{y}_k, \vec{y'}_k) = \left(A^k \vec{y}_0, A'^k \vec{y}_0\right) = \left(\vec{y}_0, A'^k \vec{y}_0\right) = \left(\sum_{i=1}^{n} a_i x_i \, , \sum_{j=1}^{n} b_j \lambda_j x'_j \right), \tag{29}$$

таким образом,

$$\lambda_1 \approx \frac{(\vec{y}_k, \vec{y'}_k)}{(\vec{y}_{k-1}, \vec{y'}_k)} = \frac{\left(A^k \vec{y}_0, A'^k \vec{y}_0\right)}{\left(A_{k-1} \vec{y}_0, A'^k \vec{y}_0\right)}. \tag{30}$$

1.3.3 Метод вращений Якоби

Данный метод является одним из наиболее часто используемых при решения полной задачи нахождения собственного значения при условии, что используемая матрица симметрична.

Дадим определение матрице вращения. Ортогональная матрица

$$\begin{pmatrix} cos(\varphi) & sin(\varphi) \\ -sin(\varphi) & cos(\varphi) \end{pmatrix} \tag{31}$$

порождает преобразование поворота на угол φ в двумерной плоскости. Матрица $Q = (q_{i,j})^n{}_{i,j=1}$, отличающаяся от единичной только лишь четырьмя элементами (32) называется матрицей вращения:

$$q_{k,k} = cos(\varphi);\ q_{l,l} = cos(\varphi);\ q_{k,l} = sin(\varphi);\ q_{l,k} = -sin(\varphi), \quad (32)$$

где $1 < k < l < n$ – заданные целые числа.

В данном случае Q – ортогональная матрица. Она порождает преобразование поворота на угол φ в двумерной плоскости, натянутой на векторы канонического базиса с номерами k, l.

Далее опишем основную идею метода вращений Якоби. Как уже говорилось матрица A – симметричная матрица. Образуем матрицу T, столбцами которой будут ортонормированные собственные векторы $e_1, e_2, \dots, e_n$. для данной матрицы T справедлива формула (33)

$$T'AT = \Lambda, \quad (33)$$

где Λ — диагональная матрица с элементами $\lambda_1, \lambda_2, \dots, \lambda_n$ на диагонали.

В методе вращения Якоби матрица T строится как предел последовательности ортогональных матриц T_s так, что:

$$\lim_{s\to\infty} \bar{T}_s' \cdot A \cdot T_s = \Lambda, \quad (34)$$

причем при каждом s матрица T_s конструируется как произведение матриц вращения.

Образуем по исходной матрице A матрицу $\hat{A} = \mathrm{Q}'\mathrm{AQ}$, и подбираем параметры матрицы вращения, то есть значения k, l, φ, так, чтобы матрица была максимально близка к диагональной. Опуская простые вычисления, представим формулу для вычисления суммы квадратов внедиагональных элементов матрицы $\hat{A}$:

$$\sum_{i\neq j}\hat{a}_{i,j}^2=\sum_{i\neq j}a_{i,j}^2-2a_{k,l}^2+2\left[a_{k,l}\cos(2\varphi)+\frac{1}{2}\left(a_{l,l}-a_{k,k}\right)\sin(2\varphi)\right]^2. \quad (35)$$

Далее, определим числа k, l из условия (36):

$$\left|a_{k,l}\right|=\max_{i\neq j}\left|a_{i,k}\right| \quad (36)$$

и затем угол φ из условия (37)

$$a_{k,l}\cos(2\varphi)+\frac{1}{2}\left(a_{l,l}-a_{k,k}\right)\sin(2\varphi)=0 \quad (37)$$

или (38)

$$tg(2\varphi)=\frac{2a_{k,l}}{a_{l,l}-a_{l,l}}. \quad (38)$$

Сумма квадратов внедиагональных элементов матрицы $\hat{A}$ принимает наименьшее значение, если использовать данное условие при выборе параметров для матрицы вращения.

Опишем и сам алгоритм метода вращений Якоби. Пусть $A_0 = A$. Образуем последовательность матриц $A_0, A_1, \dots, A_n$, при помощи рекуррентной формулы (39):

$$tA_{p+1}=Q_p^{'}\mathrm{A}_p\mathrm{Q}_p\,;p=0,1,2,\dots,n, \quad (39)$$

где параметры матрицы вращения Q_p задаются так, что сумма квадратов внедиагональных элементов матрицы A_{p+1} уменьшается, то есть используя формулы вида (37-39). Продолжаем итерации до тех пор, пока все внедиагональные элементы матрицы A_p не станут достаточно малыми.

При достижении достаточно малых значений в качестве приближений к собственным числам матрицы A принимаются диагональные элементы матрицы A_p, а столбцы матрицы $\mathrm{Q}_0, \mathrm{Q}_1, \ldots, \mathrm{Q}_p$, считаются приближениями к собственным векторам матрицы A.

1.3.4 Метод QR

Далее приведем описание метода QR. Данный алгоритм использует плоские преобразования вращений Гивенса. Матрица, определяющая эти преобразования имеет вид $\mathrm{G}_{ij} = (g_{ml})_{m,l=1}^{n}$. Данная матрица имеет структуру, похожею на матрицу плоских вращений Якоби T_{ij}, только здесь двумерная подматрица из элементов, стоящих на пересечении i и j строк и столбцов, используется в виде (40):

$$\mathrm{G}_{ij} = \begin{pmatrix} s & c \\ -c & s \end{pmatrix}. \tag{40}$$

Данные числа c и s связываются соотношением $s^2 + c^2 = 1$. Первый полный шаг преобразования Гивенса, применяемого к матрице Хессенберга B n-го порядка в рамках QR метода, состоит из $n-1$ элементарных подшагов, основной целью которых является последовательное обнуление всех поддиагональных элементов в столбцах от первого до $(n-1)$-го [7]. В результате получим разложение матрицы Хессенберга B в виде произведения ортогональной и треугольной матриц.

Для определения c и s на первом промежуточном шаге, используется произведение матриц (41):

$$G_1B = \begin{pmatrix} s & c & 0 & \dots & 0 \\ -c & s & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 1 \end{pmatrix} \begin{pmatrix} b_{11} & b_{12} & b_{13} & \dots & b_{1n} \\ b_{21} & b_{22} & b_{23} & \dots & b_{2n} \\ 0 & b_{32} & b_{33} & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & b_{nn} \end{pmatrix}$$
$$= \begin{pmatrix} sb_{11} + cb_{21} & sb_{12} + cb_{22} & sb_{13} + cb_{23} & \dots & sb_{1n} + cb_{2n} \\ -cb_{11} + sb_{21} & -cb_{12} + sb_{22} & -cb_{13} + sb_{23} & \dots & -cb_{1n} + sb_{2n} \\ 0 & b_{32} & b_{33} & \dots & b_{3n} \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & b_{nn} \end{pmatrix}. \quad (41)$$

При этом $c = cos\theta$ и $s = sin\theta$ берутся такими, что $-cb_{11} + sb_{21} = 0$. Это означает, что в результате первого промежуточного шага матрица $B_1 = G_1B$, полученная при использовании таких c и s, не будет содержать ненулевых элементов под диагональю в первом столбце.

Аналогично совершается второй промежуточный шаг: матрица $B_2 = G_1B_1$ получается из предыдущей B_1 с помощью матрицы Гивенса G_2, отличающейся от G_1 тем, что подматрица $(sc - cs)$ смещается на одну позицию вдоль диагонали и угол поворота подбирается так, чтобы в матрице B_2обнулить элементы $b_{32}^{(2)}$.

Продолжив далее преобразования Гивенса, в итоге получим правую треугольную матрицу:

$$B_{n-1} = G_{n-1}G_{n-2}, \dots, G_2G_1B. \quad (42)$$

Последнее равенство можно переписать в виде:

$$(G_{n-1}G_{n-2}, \dots, G_2G_1)^{-1}B_{n-1} = B, \quad (43)$$

который позволяет считать выполненным требуемое при $k = 1$ разложение

$$B = Q_1R_1 , \quad (44)$$

где $Q_1 = (G_{n-1}G_{n-2}, \dots, G_2G_1)^{-1} = G_1^T, \dots, G_{n-1}^T$ – ортогональная, а $R_1 = B_{n-1}$ – правая треугольная матрицы. При этом матрица

$$A_2 = R_1Q_1 = (G_{n-1}\, G_1)B(G_{n-1}\, G_1)_1^{-1}, \tag{45}$$

являющаяся результатом первого полного шага QR метода, сохраняет не только спектр данной матрицы, но и форму Хессенберга, благодаря чему приведение исходной матрицы A к почти треугольному виду B достаточно сделать только один раз [7].

Очевидно, скалярные параметры $c_j = cos\theta_j$ и $s_j = \sin\theta_j$ матриц Гивенса G_j, благодаря которым происходит переход от матрицы Хессенберга $B = (B_{ij})_{i,j=1}^n$ через матрицы Хессенберга $B = (B_{im}^{(j)})_{i,m=1}^n$ к матрице Хессенберга A_2, можно вычислять на j-ом промежуточном шаге ($j = 1,2, \dots, n-1$) по формулам:

$$c_j = \frac{1}{\sqrt{1+t_j^2}};\ s_j = t_jc_j, \tag{46}$$

где:

$$t_j = \frac{b_{jj}^{(j-1)}}{b_{j+1,j}^{(j-1)}} = tg\theta_j; b_{ij}^{(0)} = b_{ij}. \tag{47}$$

Также можно модифицировать данный метод, уменьшив количество итераций, получая метод называемый QR со сдвигом.

QR алгоритм со сдвигами состоит в следующем. Положить $k = 0$ и выполнить:

1. выбрать сдвиг s вблизи некоторого собственного значения A;

2. вычислить по схеме QR разложение для матрицы $A^{(k)} - s_kE = Q^{(k)}R^{(k)}$;

3. определить матрицу $A^{(k+1)} = R^{(k)}Q^{(k)} + s_k E$;

4. если сходимость к собственным значениям не достигнута, положить $k = k + 1$ и перейти к пункту 1.

Матрицы $A^{(k)}$ и $A^{(k+1)}$ в QR алгоритме со сдвигами будут ортогонально подобными:

$$\begin{aligned} \mathrm{A}^{(\mathrm{k}+1)} = \mathrm{R}^{(\mathrm{k})}\mathrm{Q}^{(\mathrm{k})} + \mathrm{s_k}\mathrm{E} = \left(\mathrm{Q}^{(\mathrm{k})}\right)^{\mathrm{T}} \cdot \mathrm{Q}^{(\mathrm{k})}\mathrm{R}^{(\mathrm{k})}\mathrm{Q}^{(\mathrm{k})} + \mathrm{s_k}\left(\mathrm{Q}^{(\mathrm{k})}\right)^{\mathrm{T}}\mathrm{Q}^{(\mathrm{k})} \\ = \left(\mathrm{Q}^{(\mathrm{k})}\right)^{\mathrm{T}}\left(\mathrm{Q}^{(\mathrm{k})}\mathrm{R}^{(\mathrm{k})} + \mathrm{s_k}\mathrm{E}\right)\mathrm{Q}^{(\mathrm{k})} = \left(\mathrm{Q}^{(\mathrm{k})}\right)^{\mathrm{T}}\mathrm{A}^{(\mathrm{k})}\mathrm{Q}^{(\mathrm{k})}; \end{aligned} \quad (48)$$

Если $\mathrm{s_k}$ – точное собственное значение матрицы $A^{(k)}$, то QR-итерация сойдется за один шаг.

Если $\mathrm{s_k}$ не является точным собственным значением, то элемент $a_{nn}^{(k+1)}$ считается сошедшим, если левый нижний блок $A^{(k+1))}$ достаточно мал. Поскольку матрица $A^{(k+1)}$получается из матрицы A ортогональным подобием, то $A^{(k+1)}$включает в себя ошибки округлений порядка $O(\varepsilon\|A\|)$. Таким образом, любой поддиагональный элемент в $A^{(k+1)}$, по абсолютной величине меньший, чем $O(\varepsilon\|A\|)$, мог быть нулем, поэтому его можно заменить на ноль. То есть при условии $\left|a_{n.j}^{(k+1)}\right| < \varepsilon|A|, j = 1,..n-1,$ можно положить $a_{n.j}^{(k+1)} = 0$.

Также следует упомянуть что не обязательно брать верхнюю треугольною матрицу R, точно также можно выбрать нижнюю треугольную. Это приведёт к QL алгоритму, поскольку данное разложение будет представлено в виде A=QL

Если сначала привести диагональную матрицу к трехдиагональной форме и использовать QL-алгоритм, это даст меньшую ошибку округления, поэтому именно он и применяется на практике в большинстве случаев и выбран для дальнейшей реализации.

1.4 Выводы по главе 1

Таким образом, в первой главе данной работы была описана поставленная задача для исследования, приведены основные термины, используемые в ходе рассмотрена задачи.

Для рассматриваемой проблемы представлены основные методы её решения, а именно два точных метода – метод Данилевского и Леверрье-Фадеева, а также четыре итерационных – степенной метод, метод скалярных произведений, метод вращений Якоби и метод QR . Помимо этого для метода QR рассмотрена его модификация – метод QL со сдвигом.

Учитывая специфику исследования, можно утверждать, что точные методы для решения данной задачи не подходят, так как решения характеристического уравнения для матриц большой размерности имеют очень большую вычислительную сложность. Также Метод скалярных произведения не выбирается по причине его схожести со степенным методом, в которых отличается только итерационный процесс, а сравнивать два практически идентичных алгоритма не имеет смысла.

Поэтому для программной разработки в ходе данной работы выбраны степенной метод, метод вращений Якоби и QL метод со сдвигом.

Для проведения анализа необходимо выбрать характеристики методов, по которым будет проводиться сравнение. В данной работе в качестве данных характеристики будут рассматриваться время выполнения алгоритма, количество итераций до достижения сходимости и точность приближения к собственному значению

В дальнейшем разработанные методы будут использованы для проведения экспериментов и сбора данных, необходимых для сравнительного анализа методов нахождения собственных значений симметричных матриц большой размерности.

2 Программная реализация методов нахождения собственных значений

2.1 Структура программы

Как уже было сказано, для сравнения и реализации были выбраны три основных метода нахождения собственных значений: степенной метод, метод вращений Якоби, и метод QL со сдвигом. Данные методы реализованы на языке C++ в среде Qt Creator.

Qt Creator – это кроссплатформенная свободная среда программирования для разработки на языках C и C++. Разработана для работы с фреймворком Qt и включает в себя графический интерфейс отладчика и визуальные средства разработки интерфейса.

Основа структуры программы включает в себя:

1) подключенные библиотеки (iostream, math, ctime);
2) функция генерации матрицы заданной размерности, использующая генератор псевдослучайный чисел rand;
3) функция вывода матрицы;
4) функция, реализующая степенной метод;
5) функция, реализующая метод вращения Якоби;
6) функция, реализующая преобразование Хаусхолдера, используемая для реализации метода QL;
7) вспомогательная и основная функции, реализующие метод QL со сдвигом;
8) основная часть;
9) интерфейс программы;

Также в коде предусмотрена возможность замера времени выполнения алгоритма и количества итераций, необходимых для дальнейшего сравнительного анализа методов.

Работа программы начинается с ввода пользователем размерности матрицы. Учитывая, что рассматриваемые матрица должна быть квадратной и симметричной, вызывается функция Gen_matrix, которая, используя генератор псевдослучайных чисел создает симметричную квадратную матрицу заданной размерности заполняя её числами в диапазоне от 0 до 1. Пример сгенерированной матрицы приведен на рисунке 2.

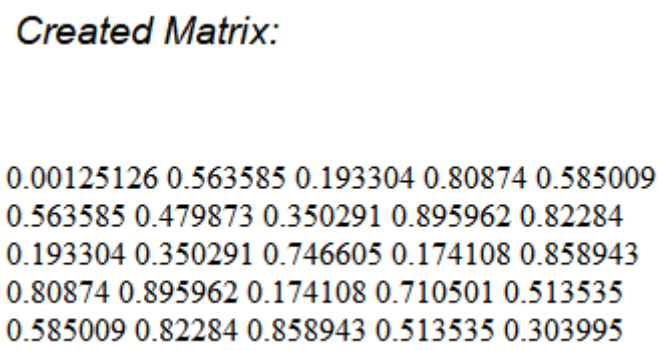

Рисунок 2 – Пример сгенерированной матрицы

После этого пользователь выбирает один из трех методов нахождения поиска собственных значений представленной в программе функция Pow_Meth, Jacobi_Meth, QL_Meth, каждая из который принимает на вхож матрицу и её размерность, после чего выводится матрица с помощью функции Print_Matrix, выводятся собственные значения матрицы и данные для анализа, а именно время выполнения алгоритма, количество итераций и максимальное собственное значение матрицы. Блок-схема главной функции программы приведена на рисунке 3.

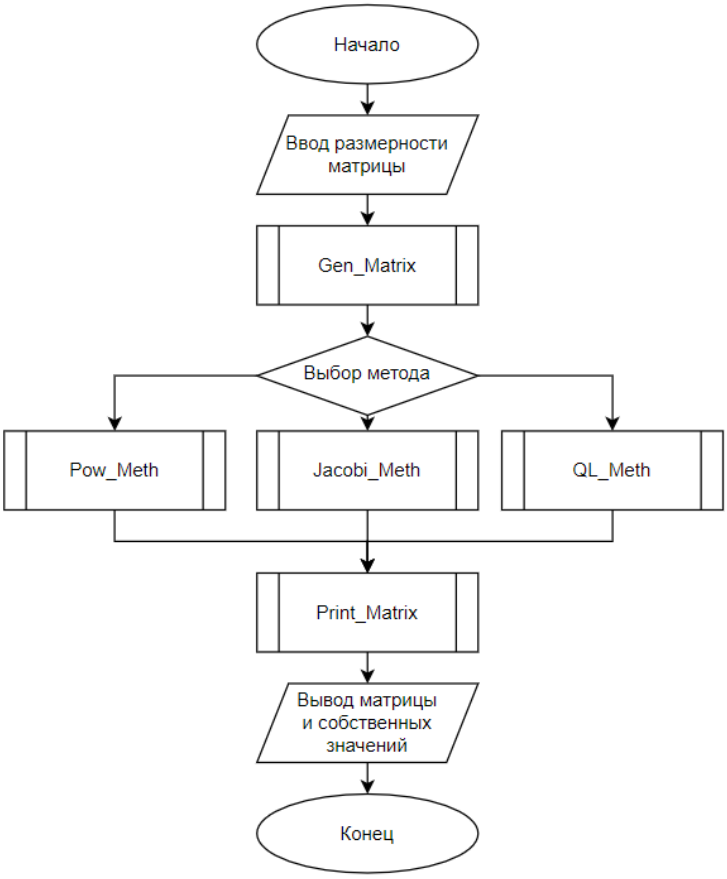

Рисунок 3- Блок-схема главной функции программы

Таким образом можно сказать, что программа принимает на вход только размерность матрицы и выбранный метод нахождения собственных значений, а на выходе выдает собственные значения и данные для анализа. Рассмотрим реализация каждого из методов подробнее

2.2 Реализация методов нахождения собственных значений

2.2.1 Степенной метод

Степенной метод реализован в функции Pow_Meth и принимает на вход матрицу и её размерность. Далее генерируется начальное приближение с помощью генератора псевдослучайный числе rand(), проводится основной блок вычислений и выводятся собственное значение и данные для дальнейшего анализа. В связи с простотой данного алгоритма его проще

представить в виде блок-схемы. Блок-схема данного метода приведена на рисунке 4.

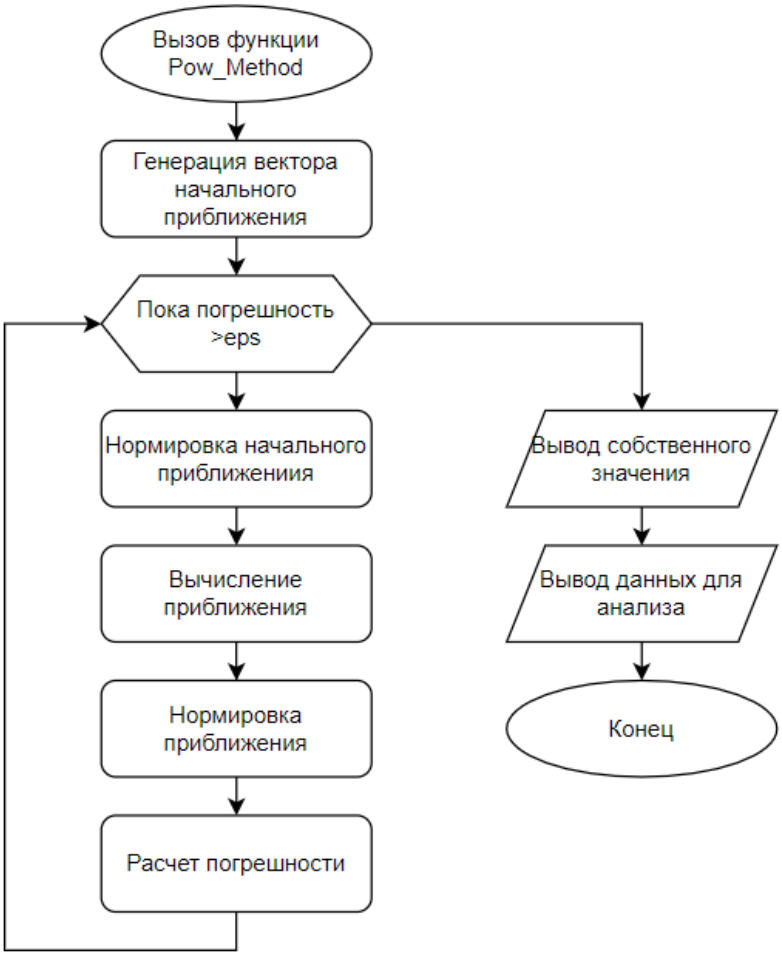

Рисунок 4 – Блок-схема степенного метода.

В коде функции важные переменные это:

1) Matr – исходная матрица;
2) n – размерность матрицы;
3) w0 – вектор начального приближения;
4) eps – точность сходимости;
5) eigenvalue – максимальное собственное значение.

К особенностям реализации степенного метода следует отнести несколько важных пунктов. Во-первых, этот метод решает неполную задачу

поиска собственных значений, в данном случае он находит только одно максимальное собственное значение. Во-вторых, скорость сходимости и количество итераций метода зависит от того, насколько близко будет сгенерировано начальное приближение к истинному значению. И в-третьих, данный метод использует точность eps для определения погрешности, которая является константой и задаётся вручную.

Всё это повлияет на полученные экспериментальные данные и должно быть учтено при проведении сравнительного анализа.

2.2.2 Реализация метода вращения Якоби

В программе реализация метода Якоби представлена в функции Jacobi_Meth. На входе эта функция получает симметричную матрицу и размерность матрицы. На выходе решается полная задача нахождения собственных векторов, то есть выводятся все собственные значении полученной на вход матрицы, а также данные, необходимые для дальнейшего анализа.

Метод Якоби заключается в последовательном обнулении недиагональных элементов итерациями вращения до тех пор, пока не получится диагональная матрица. Процесс сходимости заключается в уменьшении суммы квадратов внедиагональных элементов.

В коде функции важные переменные это:

1) Matr – исходная матрица;
2) n – размерность матрицы;
3) d – массив собственных значений матрицы;
4) переменные c и s – это cos и sin угла поворота;
5) tresh – порог, используемый для обнуление элементов по модулю больших, чем данное значение.

2.2.3 Реализация метода QL со сдвигом

QL метод со сдвигом является одним из самых сложный в реализации методов. В его реализацию включены три функции:

1) Householder_transform.

Эта функция реализует редукцию Хаусхолдера для приведения симметричной матрицы в трехдиагональную форму, необходимую для эффективного использования алгоритма QL со сдвигом. На вход данная функция принимает симметричную матрицу в обычной форме, её размерность и два дополнительных массива d и e. Массив d возвращает диагональ трехдиагональной матрицы, а массив e возвращает внедиагональные элементы.

2) QL_Alg.

Эта функция является вычислительной основой метода QL со сдвигом. На вход принимает трехдиагональную матрицу и возвращает собственные значения матрицы.

3) QL_Meth.

Функция, включающая в себя вызов предыдущих двух функций. Эта функция создаёт все необходимые промежуточные переменные, использует алгоритм трансформации Хаусхолдера, полученную трехдиагональную матрицу применяет в функции QL_Alg, и выводит массив полученных собственных значений и данные о работе алгоритма, для анализа.

Также стоит упомянуть, что при проведении экспериментов в количество итераций и время работы данного алгоритма не входит процесс трансформации Хаусхолдера, так как этот процесс считается подготовительным и не участвует в решении поставленной задачи на нахождение собственных значений.

2.3 Интерфейс разработанного приложения

Для того, чтобы работа с приложением была удобна был разработан пользовательский интерфейс с помощью редактора, интегрированного в QT-creator.

Пример реализованного интерфейса приведен на рисунке 5.

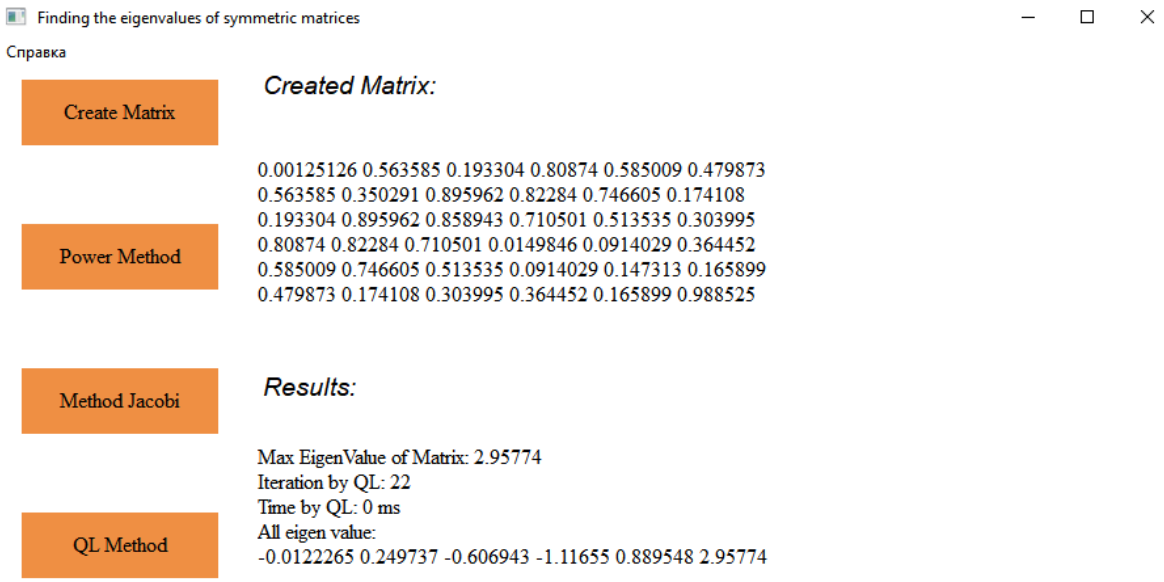

Рисунок 5 – Пример реализованного интерфейса

В данном интерфейсе реализованы 4 кнопки и 2 текстовых окна. Кнопка «Create Matrix» создает симметричную матрицу указанной размерности и выводит её в первое текстовое окно «Created Matrix». Ввод размерности матрицы происходит после нажатия на кнопку в диалоговом окне, представленном на рисунке 6

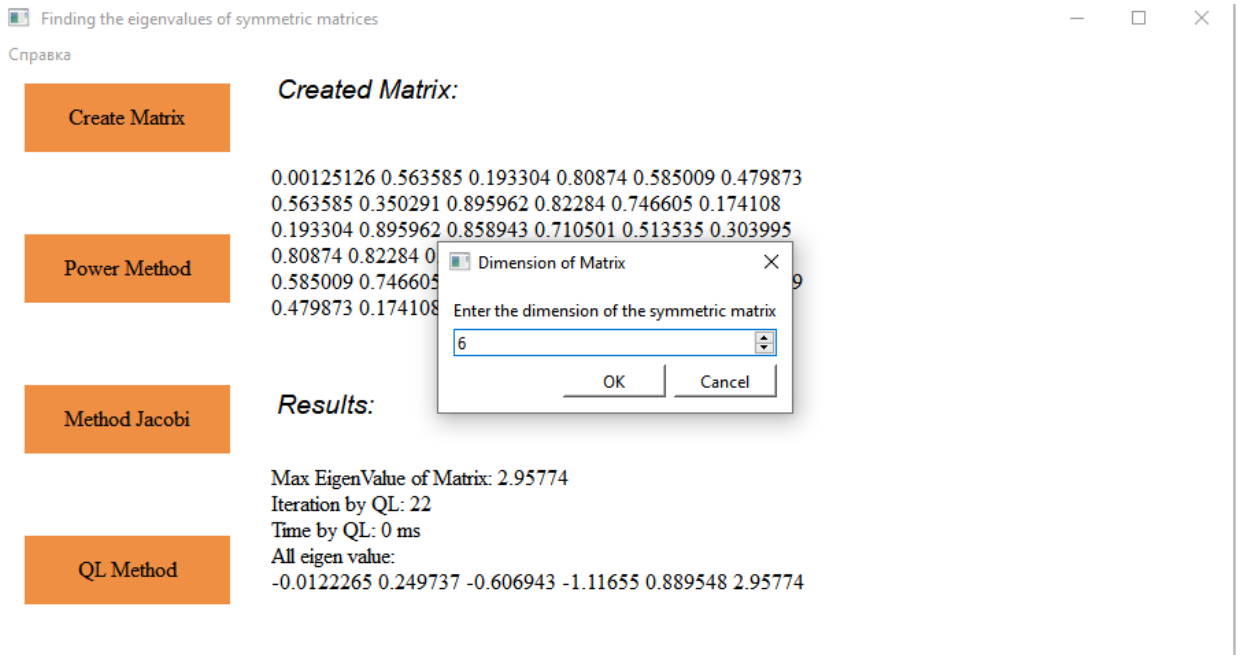

Рисунок 6 – Ввод размерности матрицы

Ввод размерности матрицы предусмотрен при нажатии на любую из кнопок. Далее, нажимая на кнопки, подписанные как методы, будет происходить вывод и отработка соответствующего метода, а результаты помещены во второе текстовое окно «Results».

Помимо этого, учитывая, что программа рассчитана на работу с матрицами большой размерности, и, исходя из предположения, что для анализа полученных результатов нет необходимости в отслеживании абсолютно всех собственных значений и печати матрицы целиком было реализовано дополнительное ограничение. Данное ограничение заключается в том, что в текстовые поля могут быть выведены только первые 10×10 значений строк и столбцов матрицы, а также только 10 собственных значений. Пример такого вывода приведен на рисунках 7 и 8 соответственно. Для примера была подана размерность матрицы в 100 на 100 элементов.

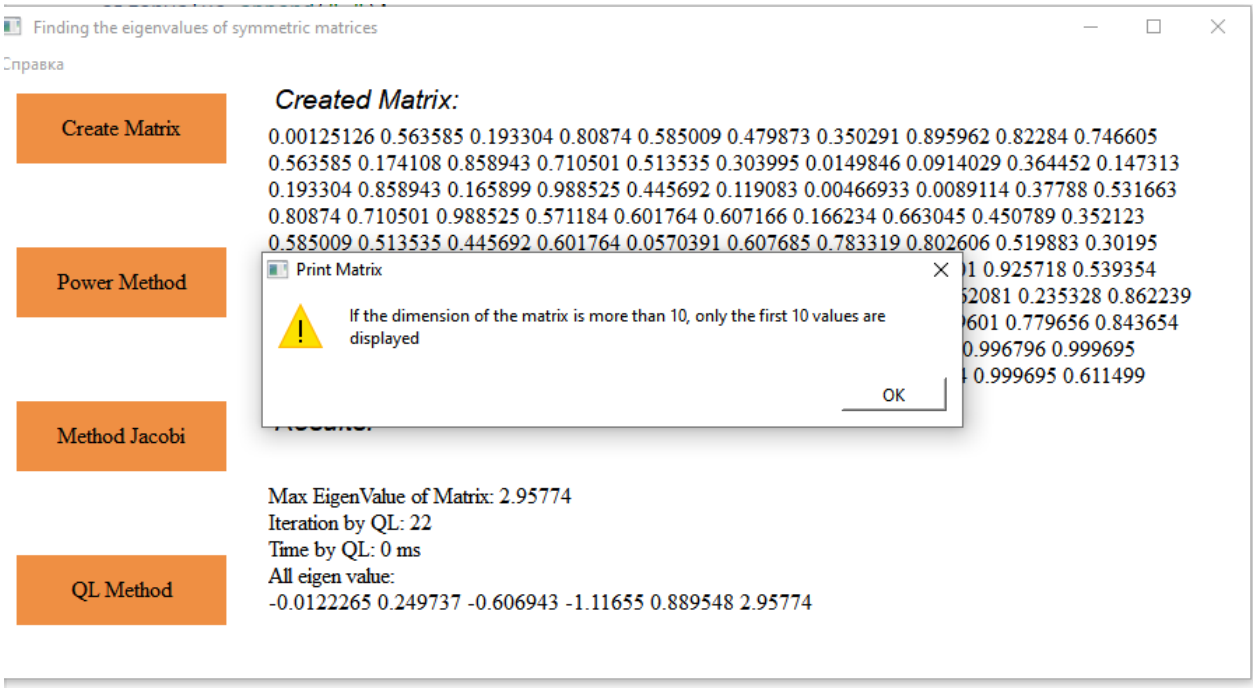

Рисунок 7 – Предупреждение о неполном выводе результатов

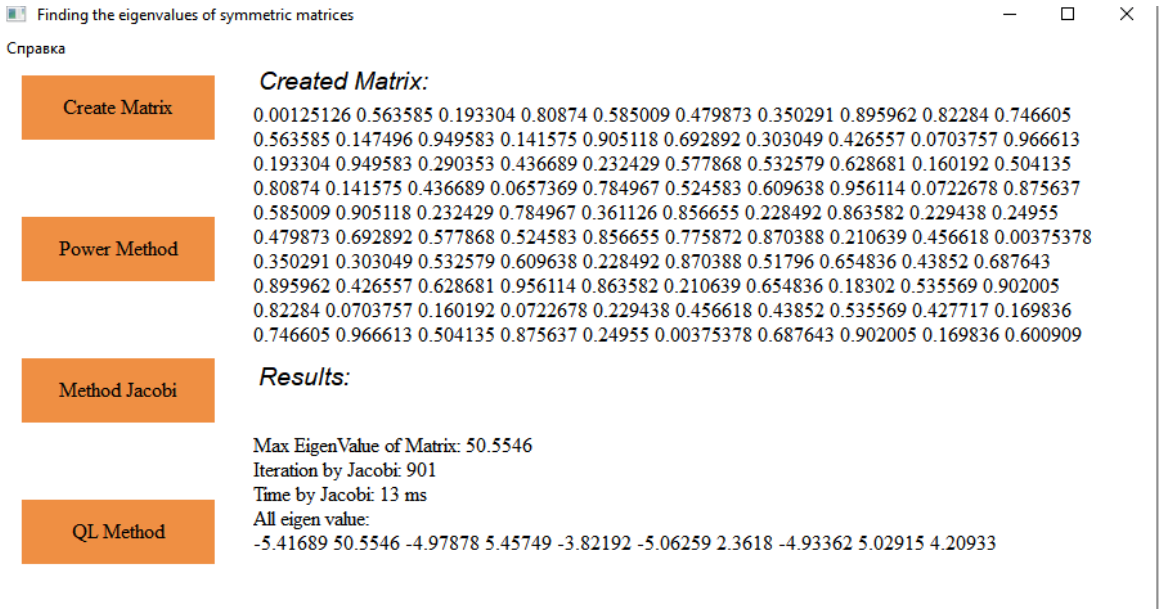

Рисунок 8 – Результат работы метода Якоби при неполном выводе

2.4 Тестирование реализации алгоритмов

Для проверки работоспособности реализованных алгоритмов проведем контрольное тестирование, подав для всех алгоритмов одинаковые входные данные в виде матрицы размерности 5×5 и сверив результаты выполнения функцией, встроенной в пакет Matlab Результаты тестирования приведены на рисунках 9-10. Для этого сгенерируем случайную матрицу и подадим одинаковые входные данные для всех функций.

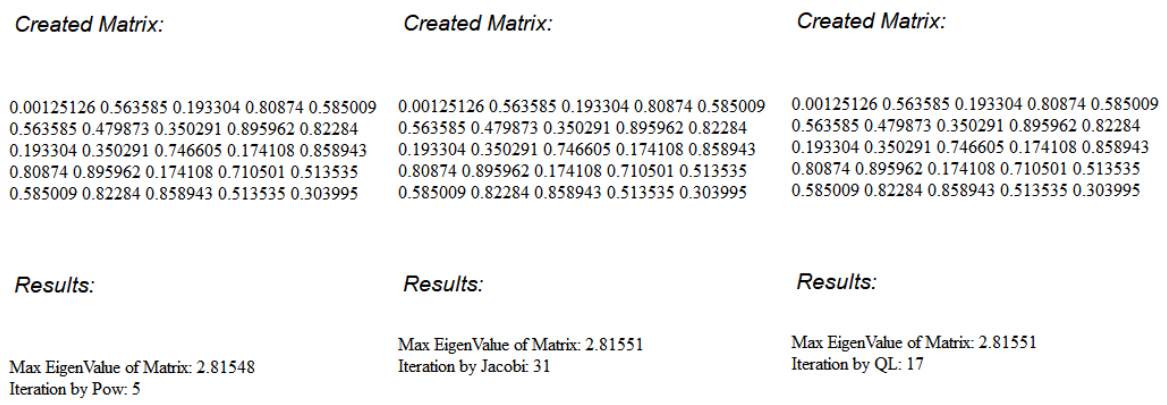

Рисунок 9 – Результаты тестирования алгоритмов

```
A = [0.00125126 0.563585 0.193304 0.80874 0.585009
     0.563585 0.479873 0.350291 0.895962 0.82284
     0.193304 0.350291 0.746605 0.1741108 0.858943
     0.80874 0.895962 0.174108 0.710501 0.513535
     0.585009 0.82284 0.858943 0.513535 0.303995]

Ans = max(eig(A))

Ans =

   2.8155
```

Рисунок 10 – Контрольные данные из пакета Matlab

Таким образом мы наблюдаем, что при одинаковых входных данных все алгоритмы выводят одинаковые результаты собственного значения, совпадающее с контрольными данными, из чего можно сделать вывод, что алгоритмы реализованы без ошибок.

2.5 Выводы по главе 2

Таким образом, были описаны программные реализации описанных методов на языке C++ с использованием QT-creator. В результате разработано приложение, которое будет использоваться для проведения вычислительных экспериментов.

В данной главе были описаны все основные функции приложения, приведены примеры разработанного интерфейса, а также проведено тестирование результатов работы программы при помощи пакета Matlab. По итогу было установлено, приложение работает верно, выводя результаты, схожие с контрольными данными.

Далее необходимо провести вычислительные эксперименты и сравнительный анализ полученных результатов.

3 Сравнительный анализ реализаций

3.1 Формат проводимых экспериментов

Основными характеристиками для приведенных алгоритмов были выбраны время выполнения, замеренное в миллисекундах (ms), количество итераций для достижения сходимости алгоритма и, учитывая, что рассмотренные методы являются итерационными, а не точными, то есть находящие приближенное решение, также рассматривается точность работы алгоритма. Для определения истинных собственных значений матрицы использовалась встроенная в пакет Matlab функций eig(), на одной и той же матрице малой размерности.

Вычислительные эксперименты проводились на персональном компьютере со следующими характеристиками, приведенными в таблице 1.

Таблица 1 – Технические данные ПК

Процессор	AMD Ryzen 5 2600 Six-Core Processor 3.40 GHz
Оперативная память	16 GB
Операционная система	Microsoft Windows 10

Эксперименты проводились на случайно сгенерированных квадратных симметричных плотных матриц с разным диапазоном значений с плавающей точкой. Принятые диапазоны: от 0 до 1, и от 0 до 100.

Для получения экспериментальных данных использовалась разработанная и описанная выше программа, результаты сведены в таблицы для наглядности.

3.2 Результаты экспериментов

Эксперименты проводились в порядке рассмотренных методов. Рассмотрим данные каждого метода в отдельности, собирая данные в таблицу и построив графики зависимости времени работы от размерности матрицы.

Степенной метод отличается от остальных двух методов по многим признакам. Во-первых, данный метод используется для решения неполной задачи нахождения собственных векторов, то есть находит только одно, максимальное собственное значение. Также данный метод является самым настраиваемым, то есть его показатели скорости схождения во многом зависят от начального приближения и заданной точности (eps). В ходе проводимых экспериментов начальное приближение генерировалось случайным образом, а значение *eps* было $1 \cdot 10^{-4}$. Для наглядности результаты сведены в таблицу 2 для диапазона от 0 до 1 и в таблицу 3 для диапазона от 0 до 100.

График зависимости размерности от времени для степенного метода приведен на рисунке 11.

Таблица 2 – Результаты работы степенного метода с диапазоном от 0 до 1

Размер матрицы	Точность	Количество итераций	Время работы
500×500	$3.1720 \cdot 10^{-5}$	12	12 мс
600×600		6	9 мс
700×700		6	11 мс
800×800		8	20 мс
900×900		9	29 мс
1000×1000		8	31 мс

Таблица 3 – Результаты работы степенного метода с диапазоном от 0 до 100

Размер матрицы	Точность	Количество итераций	Время работы
500×500	$6.9708 \cdot 10^{-5}$	8	8 мс
600×600		8	11 мс
700×700		7	14 мс
800×800		7	18 мс
900×900		9	23 мс
1000×1000		12	33 мс

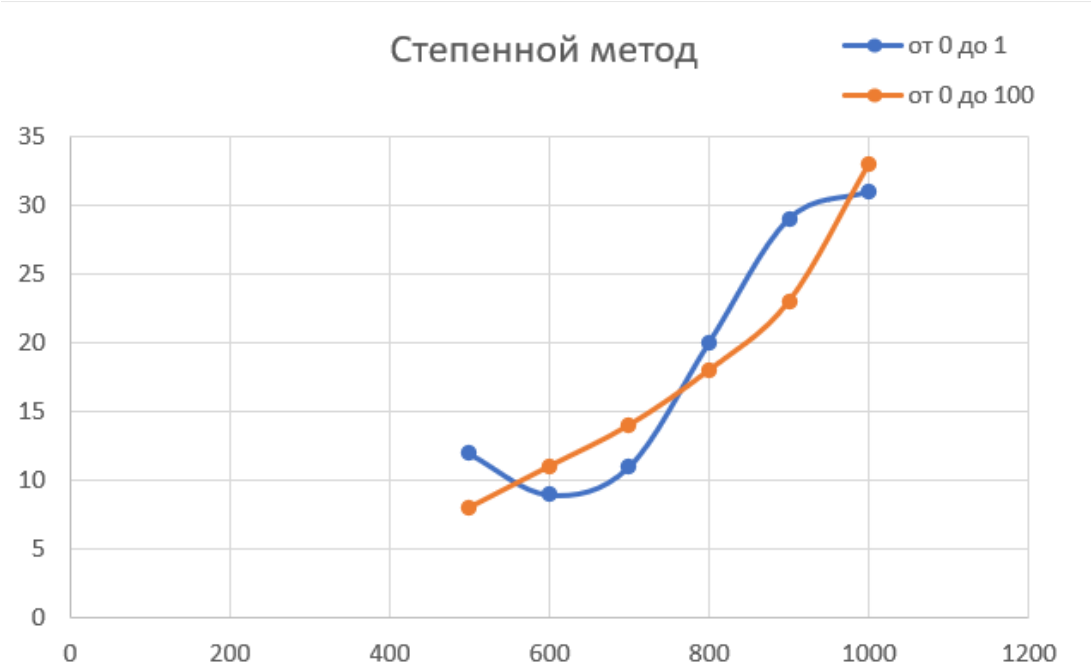

Рисунок 11 – График зависимости степенного метода

По приведенным результатам сразу можно наблюдать особенности степенного метода. Для матрицы с диапазоном значений от 0 до 1. Рост времени и количество итераций неравномерный. Это можно связать с тем, что для матрицы размером 500 было сгенерировано самое неудачное

начальное приближение, в связи с чем на лицо замедление времени работы и увеличение количества итераций.

Далее приведены результаты для метода Якоби. Следует напомнить, что метод Якоби решает полную задачу нахождения собственных значений, а значит время выполнения и количество итераций ожидается значительно больше, чем для степенного метода. Результаты сведены в таблицы 4 и 5, график представлен на рисунке 12.

Таблица 4 – Результаты работы метода Якоби с диапазоном от 0 до 1

Размер матрицы	Точность	Количество итераций	Время работы
500×500	$7.0410 \cdot 10^{-7}$	5501	2524
600×600		6601	3906
700×700		7701	7017
800×800		8801	10159
900×900		10801	15838
1000×1000		12001	26110

Таблица 5 – Результаты работы метода Якоби с диапазоном от 0 до 100

Размер матрицы	Точность	Количество итераций	Время работы
500×500	$8.6730 \cdot 10^{-6}$	5501	2549
600×600		6601	4164
700×700		7701	7020
800×800		8801	10173
900×900		10801	15684
1000×1000		12001	25979

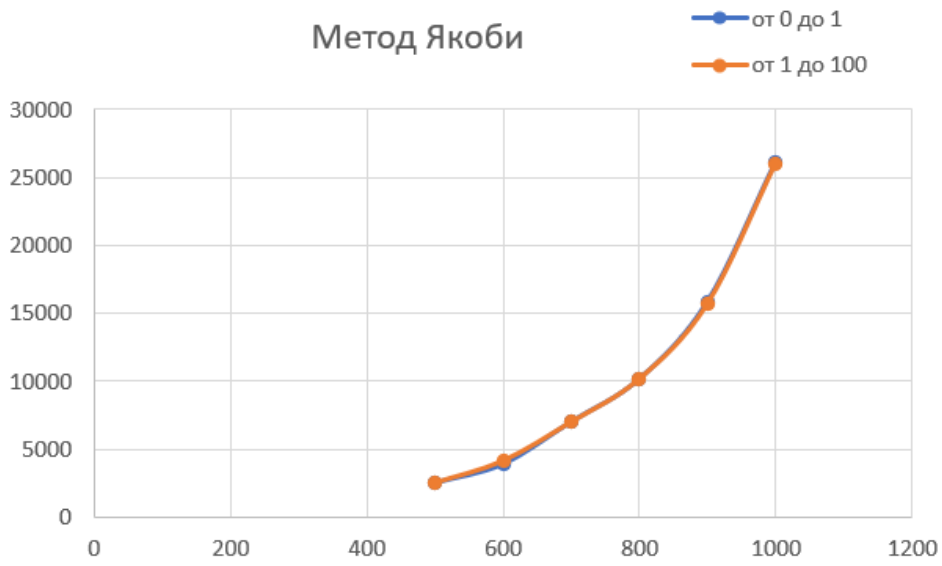

Рисунок 12 – График зависимостей метода Якоби

По полученным результатам можно сказать о том, что метод Якоби не зависит от диапазона значений матрицы, так как линии графиков практически слились. При этом рост графиков равномерный, а значения количества итерация и времени весьма велики.

Далее рассмотри метод QL со сдвигом, учитывая, что для данного метода проводилась дополнительные преобразования матрицы, ожидается улучшение показателей по сравнению с методом Якоби. Результаты работы метода QL сведены в таблицы 6 и 7, график представлен на рисунке 13.

Таблица 6 – Результаты работы метода QL с диапазоном от 0 до 1

Размер матрицы	Точность	Количество итераций	Время работы
500×500	$2.2730 \cdot 10^{-7}$	1803	257
600×600		2177	460
700×700		2529	729

Продолжение таблицы 6

800×800		2833	1081
900×900		3257	1559
1000×1000		3601	2115

Таблица 7 – Результаты работы метода QL со сдвигом с диапазоном от 0 до 100

Размер матрицы	Точность	Количество итераций	Время работы
500×500	$2.1844 \cdot 10^{-5}$	1816	262
600×600		2165	366
700×700		2537	730
800×800		2882	1082
900×900		3242	1541
1000×1000		3585	2191

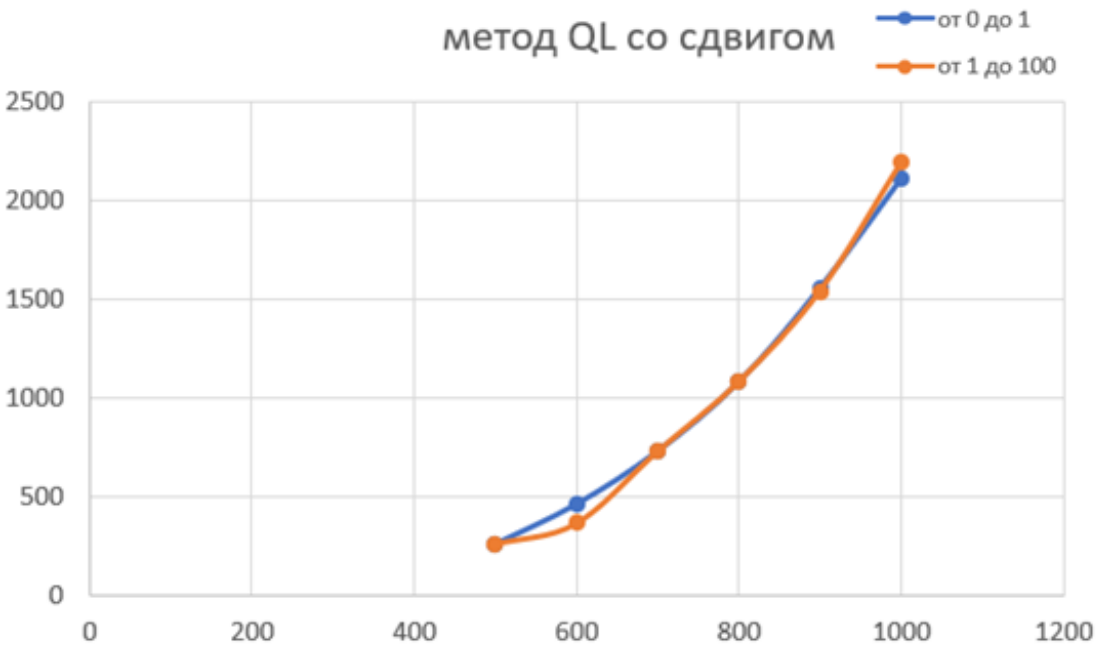

Рисунок 13 – График зависимости метода QL со сдвигом.

Собрав все экспериментальные данные, можно приступать к сравнительному анализу алгоритмов, с учетами всех особенностей данных методов и полученных во время эксперимента данных.

3.3 Сравнение результатов работы методов

Сравним все методы по каждому из полученных показателей. Начнем с одного из ключевых показателей – точности работы. В ходе экспериментов лучший результат по точности показал метод QL со сдвигом ($2.2730 \cdot 10^{-7}$) на матрице с диапазоном значений от 0 до 1.

При этом, данный метод показывает более плохой результат ($2.1844 \cdot 10^{-5}$) на матрице с диапазоном от 0 до 100, что говорит о чувствительности данного метода к разбросу значений. При этом на данном диапазоне лучший результат показывает метод Якоби ($8.6730 \cdot 10^{-6}$), хоть и тоже теряет в точности при увеличении разброса.

Степенной же демонстрирует высокую устойчивость к увеличению разницы значений, содержащийся в матрице, показывая практически идентичные результаты ($3.1720 \cdot 10^{-5}$и $6.9708 \cdot 10^{-5}$ соответственно).

Далее рассмотрим методы по количеству итераций для достижения сходимости алгоритма. Ожидаемо, лучшие результаты здесь показывает степенной метод, так как он использует начальное приближение и определенную точность для сходимости, в результате чего обрабатывает матрицу с любым диапазоном значений за несколько итераций (от 5 до 15, в зависимости от того, насколько удачно было сгенерировано начальное приближение).

Совершенно обратную ситуацию демонстрирует метод Якоби, проводя большое количество итераций и вращений матрицы, необходимых для достижения условий сходимости. Здесь демонстрируется равномерный рост количества итераций в зависимости от размерности матрицы и доходящий до 12 тысяч для матрицы размером 1000×1000, что в тысячу раз больше

значения для степенного метода и в 4 раза больше для метода QL со сдвигом, что говорит о высокой вычислительной сложности метода, вне зависимости от диапазона разброса значений матрицы.

Метод QL со сдвигом демонстрирует умеренные результаты по количеству итераций. Тот же равномерный рост в зависимости от размерности матрицы как у метода Якоби, но значения в 4 раза меньше, до 3500 итераций для матрицы размером 1000×1000. Это говорит о небольшой сложности вычислительной сложности алгоритма, несмотря на то что данный алгоритм самый сложный в реализации и требует подготовки матрицы и её специальной трансформации в трехдиагональную форму.

Рассматривая же время выполнения алгоритмов, ситуация здесь аналогична ситуации с количеством итераций. Степенной метод показывает лучшие результаты, отрабатывая матрицы любой размерности и с любый диапазоном за несколько миллисекунд (до 12 мс в рассмотренных примерах), но имея неравномерный рост, который увеличивает время работы в случае неудачно выбранного начального приближения.

Метод Якоби и QL со сдвигом, наоборот, демонстрируют равномерный рост времени выполнения в зависимости от размерности матрицы, при этом не имея заметной зависимости от диапазона рассматриваемых значений матрицы. Но нельзя не заметить, что скорость работы метода Якоби хуже времени выполнения метода QL со сдвигом, проигрывая по скорости до 13 раз (26110 мс и 2115 мс соответственно для матриц размерностей 1000×1000 и в диапазоне от 0 до 100), и только увеличивает разрыв в зависимости от размерности матрицы.

3.4 Выводы по главе 3

Подводя итоги сравнительного анализа, можно сделать следующие выводы:

1) Самым простым и быстрым методом является степенной метод, во много превосходя остальные по количеству итерации и скорости работы. Но при этом он имеет значительные минусы. Во-первых, он решает неполную задачу нахождения собственных значений, находя только максимальное значение, что сильно сужает вариативность его применения. Во-вторых, эффективность данного метода во многом зависит от того, насколько точно подобрано начальное приближение, несмотря на то что у данного метода хорошая устойчивость к разбросу значений матрицы.
2) Метод Якоби показывает самую большую вычислительную сложность, количество итераций и скорость работы, но при этом у него лучшая точность для матриц с большим диапазоном значений.
3) Метод QL со сдвигом показывает лучшую эффективности среди рассмотренных методов. Данный метод демонстрирует лучшую точность работы алгоритма для матриц с небольшим разбросом диапазона значений, показывает небольшую скорость работы и затраченное количество итераций. При этом данный метод, как и метод Якоби, решает полную задачу нахождения собственных значений, то есть находит все собственные значения матрицы, что расширяет вариативность его применения.

Заключение

В ходе выполнения выпускной квалификационной работы было проведено исследование, объектом которого являлась задача нахождения собственных значений для симметричных матриц большой размерности.

Целью данной выпускной квалификационной работы был сравнительный анализ алгоритмов нахождения собственных значений матриц большой размерности и их программная реализация. В ходе данной работы были поставлены и выполнены следующих задачи.

1) Рассмотрены основные существующие алгоритмы решения задачи поиска собственных значений, а именно метод Данилевского, метод Леверрье-Фадеева, степенной метод, метод вращений Якоби, а также же QR метод и его модификация метод QL со сдвигом.
2) В ходе рассмотрения методов для решения поставленной задачи для реализации были выбраны вращения Якоби, QL со сдвигом, а также степенной метод.
3) Разработано приложение на языке программирования C++ в среде Qt-creator, реализующий перечисленные выше алгоритмы, а также проводящая замер данных, необходимых для сравнительного анализа алгоритмов.
4) Проведены эксперименты с использованием разработанной программы и получены результаты для анализа
5) Проведен сравнительный анализ полученных результатов и сделаны соответствующие выводы.

В результате проведенного сравнительного анализа, были сделаны следующие выводы из проделанной работы:

1) Самым простым и быстрым методом является степенной метод, во много превосходя остальные по количеству итерации и скорости работы. Но при этом он имеет значительные минусы. Во-первых, он решает неполную задачу нахождения собственных значений, находя

только максимальное значение, что сильно сужает вариативность его применения. Во-вторых, эффективность данного метода во многом зависит от того, насколько точно подобрано начальное приближение, несмотря на то что у данного метода хорошая устойчивость к разбросу значений матрицы.

2) Метод Якоби показывает самую большую вычислительную сложность, количество итераций и скорость работы, но при этом у него лучшая точность для матриц с большим диапазоном значений.
3) Метод QL со сдвигом показывает лучшую эффективности среди рассмотренных методов. Данный метод демонстрирует лучшую точность работы алгоритма для матриц с небольшим разбросом диапазона значений, показывает небольшую скорость работы и затраченное количество итераций. При этом данный метод, как и метод Якоби, решает полную задачу нахождения собственных значений, то есть находит все собственные значения матрицы, что расширяет вариативность его применения.

Список используемой литературы

1. Агарагимов М.Р., Паштаев Б.Д., Мазанов Р.Р. Решение проблемы собственных значений матрицы / Инновационные технологии в АПК. Сборник научных трудов Всероссийской научно-практической конференции с международным участием., 2017. С. 213-218
2. Ануфриев И.Е. MATLAB 7 / И.Е. Ануфриев, А.Б. Смирнов, Е.Н. Смирнова. – СПб.: БХВ-Петербург, 2005. – 1104 с.
3. Атискова О.П., Нурдавлетова Л. А. Сравнительных анализ методов нахождения собственных значений матрицы. Достижения и приложения современной информатики, математики и физики. Материалы III Всероссийской научно-практической заочной конференции. А.Н. Вильданов. 2014. С 9-12.
4. Банников А.С. Численные методы. – Ученое пособие / А.С. Банников, И. Н Ким, Латыпова Н.В. – М.: Удмуртский государственный университет (Ижевск), 2018 – 78 с.
5. Бабенко К.И. Основы численного анализа / К.И. Бабенко. – М., Ижевск: НИЦ «Регулярная и хаоптическая динамика», 2002. – 848 с.
6. Бахвалов Н.С. Численные методы в задачах и упражнениях: учебное пособие / Н.С. Бахвалов, А.В. Лапин, Е. В. Чижонков; ред. В.А. Садовничий. М.: Изд-во "Лаборатория знаний" (ранее "БИНОМ. Лаборатория знаний") 3-е издание, 2013. – 240 с.
7. Вержбицкий В.М. Численные методы. Линейная алгебра и нелинейные уравнения / В.М. Вержбицкий. – М.: Оникс 21 век, 2-е издание, 2005. – 430 с.
8. Деммель Дж. Вычислительная линейная алгебра. Теория и приложения / Дж. Деммель. – М.: Мир, 2001. – 430 с.
9. Долгой В.Е. Построение эффективных итерационных алгоритмов для решения систем линейных алгебраических уравнений с плохо

обусловленными матрицами // Известия Южного федерального университета. Технические науки. – 2010. – N 6. – С. 39–42.

10. Долгополов Д.В. Методы нахождения собственных значений и собственных векторов матриц / Д.В. Долгополов. -Санкт-Петербург: СПбГТИ(ТУ), 2005. – 39с.

11. Калиткин Н.Н. Численные методы/ Н.Н Калиткин – М., БХВ-Петербург Учебная литература для вузов, 2011, 592 стр.

12. Козин Р.Г. Алгоритмы численных методов линейной алгебры и их программная реализация: Учебно-методическое пособие / Р.Г. Козин. – М.: НИЯУ МИФИ, 2012. – 192 с.

13. Саад, Ю. Итерационные методы для разреженных линейных систем: в 2 т. / Ю. Саад; пер. с англ. Х.Д. Икрамова. – М.: Изд-во Московского университета, 2013. – Т. 1. – 344 с.

14. Уилкинсон Дж.Х. Справочник алгоритмов на языке АЛГОЛ. Линейная алгебра. Перевод с английского под ред. д-ра техн. наук проф. Ю. И. Топчеева. М., ≪Машиностроение≫, 1976 г. С. 216–221.

15. Фаддеев Д.К., Вычислительные методы линейной алгебры / Д.К. Фаддеев, В.Н. Фаддеева. – Учебники для вузов. Специальная литература М.: Изд-во «Лань», 4-е издание 2009. – 736 с.

16. Stroustrup B. The C++ Programming Language. Fourth Edition. / B. Stroustrup – Addison-Wesley Professional, 4th edition, 2013 – 1376 p.

17. Gregoire M. Professional C++. Third Edition. / M. Gregoire – Wrox, 3rd edition, 2014 – 984 pp.

18. Nocedal J., Numerical Methods for Solving Inverse Eigenvalue Problems. / J. Nocedal – Forgotten Books, 2017 – 24 p.

19. Wilkinson J. H., The Algebraic Eigenvalue Problem. // J. H. Wilkinson – Clarendon Press, Oxford, 1965 – 662 p.

20. Z. Li, C. Bu and H. Wang, "Inverse Eigenvalue Problem for Generalized Arrow-Like Matrices," Applied Mathematics, Vol. 2 No. 12, 2011, p. 1443-1445