МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(национальный исследовательский университет)»

Институт (Филиал) № 8 «Компьютерные науки и прикладная математика» **Кафедра** 806

Группа М8О-406Б-19 **Направление подготовки** 01.03.02 «Прикладная математика и информатика»

Профиль Информатика

Квалификация: бакалавр

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

на тему: Моделирование систем уравнений химической кинетики на базе схем Рунге-Кутты

Автор ВКРБ:	Садаков Александр Александрович	(_____)
Руководитель:	Демидова Ольга Львовна	(_____)
Консультант:	—	(_____)
Консультант:	—	(_____)
Рецензент:	—	(_____)

К защите допустить

Заведующий кафедрой № 806 Крылов Сергей Сергеевич (_____)

____ мая 2023 года

Москва 2023

РЕФЕРАТ

Выпускная квалификационная работа бакалавра состоит из 53 страниц, 25 рисунков, 6 таблиц, 31 использованного источника, 1 приложения.

ЧИСЛЕННЫЕ МЕТОДЫ, МЕТОД РУНГЕ-КУТТУ, ЖЁСТКИЕ СИСТЕМЫ, СДУ, ОДУ, ХИМИЧЕСКАЯ КИНЕТИКА

Объектом разработки является программа, позволяющая решать системы дифференциальных уравнений.

Цель работы — разработка и отладка программы, выбор оптимальных методов решения обыкновенных дифференциальных уравнений.

В процессе работы были использованы явные и неявные методы Рунге-Кутты, метод QR и LU разложения матрицы, методы итераций, Зейделя и Ньютона для решения систем нелинейных алгебраических уравнений.

В результате работы была создана программа, позволяющая моделировать уравнения химической кинетики и решать СДУ при помощи большой коллекции методов.

Обыкновенные дифференциальные уравнения и системы дифференциальных уравнений широко используются для математического моделирования процессов и явлений в различных областях науки и техники. Переходные процессы в радиотехнике, динамика биологических популяций, модели экономического развития, движение космических объектов и так далее исследуются с помощью ОДУ и СДУ.

Данное ПО можно использовать для исследований химической кинетики. Помимо этого её можно использовать в учебных целях для решения простых ОДУ или систем, а также в инженерном моделировании процессов газовой динамики.

Преимуществами моей работы по сравнению с аналогами являются большое число явных, неявных или вложенных методов на выбор, возможность добавления своего метода решения, быстрота вычисления результата, возможность построения правых частей ОДУ при помощи разработанного интерфейса, вывод результатов расчётов в текстовом и графическом виде.

В дальнейшем программу можно улучшить путём разработки модулей для работы с динамикой биологических популяций, моделями экономического развития, движением космических объектов и так далее.

СОДЕРЖАНИЕ

ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

В настоящей выпускной квалификационной работе бакалавра применяют следующие термины с соответствующими определениями:

Жёсткая система — ОДУ, численное решение которого явными методами является неудовлетворительным из-за резкого увеличения числа вычислений или из-за резкого возрастания погрешности при недостаточно малом шаге

Методы Рунге-Кутты — большой класс численных методов решения задачи Коши для обыкновенных дифференциальных уравнений и их систем

Условие химического равновесия — равенство полных химических потенциалов исходных веществ и продуктов

Химическая кинетика — раздел физической химии, изучающий закономерности протекания химических реакций во времени, зависимости этих закономерностей от внешних условий, а также механизмы химических превращений

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

В настоящей выпускной квалификационной работе бакалавра применяют следующие сокращения и обозначения:

ДУ — дифференциальное уравнение

ОДУ — обыкновенное дифференциальное уравнение

СДУ — система дифференциальных уравнений

СНУ — система нелинейных уравнений

ρ — плотность, $\frac{\text{кг}}{\text{м}^3}$

t — время, с

P — давление, Па

T — температура, К

V — объём, м^3

R — универсальная газовая постоянная, $\frac{\text{Дж}}{\text{К} \cdot \text{моль}}$

U — внутренняя энергия, $\frac{\text{Дж}}{\text{кг}}$

N — число компонентов в смеси

G — потенциал Гиббса, $\frac{\text{Дж}}{\text{кг}}$

γ_i — мольно-массовая концентрация i-го компонента, $\frac{\text{моль}}{\text{кг}}$

μ_i — химический потенциал i-го компонента, $\frac{\text{Дж}}{\text{моль}}$

N_R — число реакций

W_i — скорость образования i-го компонента, $\frac{\text{моль}}{\text{м}^3 \cdot \text{с}}$

$\nu^{(r)}$ — стехиометрические коэффициенты

$W^{(r)}$ — скорости r-ой химической реакции, $\frac{\text{моль}}{\text{м}^3 \cdot \text{с}}$

$q^{(r)}$ — молекулярность r-ой элементарной реакции

ВВЕДЕНИЕ

Для решения систем уравнений, описывающих химические процессы, нужно использовать методы, которые дали бы высокую точность при низких затратах по времени, потому что ЭВМ должна работать в режиме реального времени. В данной работе будет рассмотрено семейство методов Рунге-Кутты.

В это семейство входит огромное число методов, как явных, так и неявных. Преимущество явных методов заключается в производительности, так как им не нужно решать на каждом шаге системы алгебраических уравнений. Отсюда следует, что использование явных методов даёт больший выигрыш по времени, чем использование более производительного оборудования и распараллеливания. Главным преимуществом неявных методов является наличие устойчивости (A-устойчивости, L-устойчивости и так далее), в результате чего полученное ими решение является гарантированно устойчивым, в отличие от явных. Однако некоторые СДУ, описывающие химические процессы можно решить и явными методами с требуемой точностью, потому что свойства A и L-устойчивости являются лишь достаточным, но не необходимым условием эффективности решения, поэтому нельзя использовать только те или иные методы. Выбрать оптимальный метод можно путём целевого тестирования.

Результат работы представляет собой программу для работы с СДУ при помощи множества методов семейства Рунге-Кутты. В работе реализовано 18 явных методов до 6 порядка точности, 9 вложенных, включая схему Дормана-Принса 4-5 порядка, 19 неявных до 6 порядка. Для решения СНУ при итерировании в неявных схемах используются схемы первого порядка (простой итерации, Зейделя) и второго порядка (метод Ньютона), причём для обращения матрицы применялся метод LU-разложения. Для дифференцирования функции при построении матрицы Якоби для метода Ньютоны использовались формулы с 4 порядком точности. При необходимости можно использовать формулы с меньшим порядком.

Для того чтобы определить, какие методы можно использовать, был реализован алгоритм вычисления числа жёсткости СДУ, использующий QR-разложение матрицы системы для поиска всех собственных чисел. Систему можно считать жёсткой, если для неё коэффициент жёсткости намного больше единицы. Чёткой границы между жёсткой и не жёсткой

системой нет, поэтому пользователь может выбрать это значение сам, например 100, или же воспользоваться классификацией [1], которая была использована в работе. Тогда для расчёта применялись только неявные схемы Рунге-Кутты.

Для удобного отображения результатов вычислений был создан генератор pdf-отчётов, в которых представлена информация о решаемой задаче, метод её решения и таблица Бутчера для этого метода, график, отображающий решение численными методами и аналитическое (при наличии) и время, затраченное на работу программы. Если задача решалась при помощи жёстких схем, то дополнительно выводится информация о количестве итераций. Так же, при необходимости, может быть построен приближающий полином. Кроме этого, реализована возможность простого добавления новых методов решения на случай, если пользователю нужно решение каким-либо специфическим методом, которого нет в программе. При помощи пользовательского интерфейса можно использовать разработанную программу как для решения ОДУ, так и в целях обучения.

Программа тестировалась как на задачах химической кинетики, так и на модельных уравнениях и дала положительные результаты.

1 АКТУАЛЬНОСТЬ ТЕМЫ

Говоря о применении ОДУ легче найти области, где они не применяются. Их используют в физике для моделирования процессов, начиная от баллистики и заканчивая перемещением по твёрдой поверхности, биологии для прогнозирования динамики биологических популяций, тонких технологических процессах и так далее. На рисунке 1 показаны области использования ОДУ. Прогнозировать те или иные процессы можно как при помощи натурных экспериментов, так и с помощью численного моделирования.

Рисунок 1 – Области применения ОДУ

Для численного моделирования тех или иных процессов нужно использовать специализированные программы и библиотеки, которых на сегодняшний день предоставлено большое количество, однако все они имеют свои ограничения и недостатки.

2 ЦЕЛЬ И ЗАДАЧИ РАБОТЫ

Целью работы является создание комплекса программ для решения ОДУ и СДУ и её дальнейшая отладка на модельных задачах. Безусловно, в этот программный комплекс должны входить различные явные и неявные методы решения ОДУ, графический пользовательский интерфейс, парсер математических выражений и многое другое.

В связи с этим можно сформулировать следующие задачи, представленные на рисунке 2:

- выполнить обзор существующих программ для решения ОДУ, выявить их основные недостатки;
- реализовать стандартный набор явных, неявных, вложенных, диагональных, а так же многошаговых методов решения ОДУ;
- разработать парсер математических выражений, который можно использовать для ввода задачи Коши. Адаптировать парсер для применения его в задачах химической кинетики, в частности для ввода химических компонент, химических реакций, скоростей химических реакций и другой информации, необходимой для моделирования химической кинетики;
- разработка графического пользовательского интерфейса, дизайна программы и инструментов для генерации pdf-отчётов о решениях задач;
- выбрать оптимальные по времени и точности схемы для моделирования различных химических процессов на основе разработки ПО для решения систем ОДУ явными и неявными методами;
- сравнение с существующими программами для решения ОДУ и СДУ.

После выполнения данных задач, можно приступить к разработке отдельных модулей для работы с задачами физики, биологии и т.д. Так же планируется работа над кроссплатформенностью.

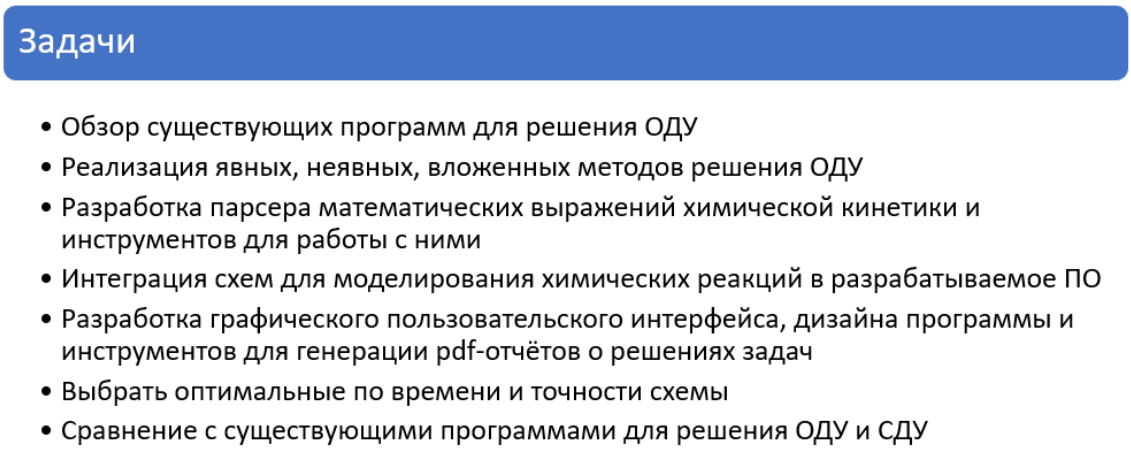


Рисунок 2 – Основные задачи проекта

Идея работы, основные элементы алгоритмов и результаты тестовых испытаний были изложены 11 апреля 2023 г. на 49-й конференции Гагаринских чтений по направлению №7 (”Математические методы в аэрокосмической науке и технике”) секции №3 (”Теоретическая механика и дифференциальные уравнения”). Темой доклада было ”Моделирование и оптимизация химической кинетики на базе схем Рунге-Кутты для решения жёстких систем ОДУ”. Тезисы доклада будут опубликованы в сборнике трудов.

Так же была подана заявка на участие в 23-й международной конференции по вычислительной механике и современным прикладным программным системам (ВМСППС’2023) с темой “Моделирование жёстких систем дифференциальных уравнений на базе схем Рунге-Кутты”, которая будет проводиться с 4 по 13 сентября 2023 г.

3 ПРОГРАММЫ И БИБЛИОТЕКИ ДЛЯ РЕШЕНИЯ ОДУ

Программ для решения ОДУ достаточно много. На рисунке 3 приведены некоторые популярные программы и библиотеки для решения ДУ. Рассмотрим детально достоинства и недостатки каждой из них.

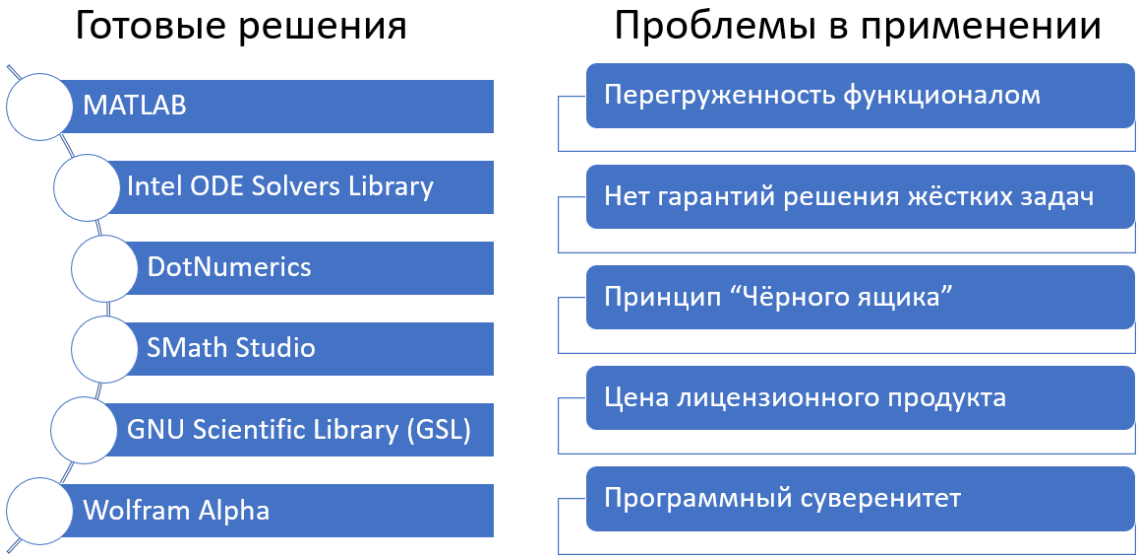


Рисунок 3 – Программы и библиотеки

Наиболее распространённый программный продукт — пакет программ для решения задач технических вычислений MATLAB [2; 3]. Первый выпуск состоялся в 1984 году. Пакет используют более миллиона инженерных и научных работников, он работает на большинстве современных операционных систем, включая Linux, macOS и Windows. Разработчиком является американская компания The MathWorks. Написан с использованием языков программирования C, C++, Fortran, Java.

Для решения ОДУ MATLAB предлагает следующие функции:

– ode23() — метод низкого порядка решения нежёстких ДУ,
– ode45() — метод среднего порядка решения нежёстких ДУ,
– ode113() — метод переменного порядка решения нежёстких ДУ,
– ode15s() — метод переменного порядка решения жёстких ДУ,
– ode23s() — метод низкого порядка решения жёстких ДУ.

MATLAB имеет свой собственный язык программирования для подготовки данных и большую библиотеку для работы с матрицами, построения графиков и численных вычислений. Однако он является довольно тяжёлым, имеет низкую скорость работы и не подходит для решения задач

химической кинетики. Помимо этого, в бесплатной версии недостаточно инструментов, а цена полной версии несколько завышена.

Альтернативу пакету MATLAB может составить отечественная программа SMath Studio [4], разработанная ООО ”ЭсМат”. SMath Studio предназначена для вычисления математических выражений и построения графиков функций. Бета-версия была выпущена в 2018 году и работа над обновлениями продолжается в настоящее время. Работа с интерфейсом программы напоминает работу с обычным листом бумаги, так как все математические выражения в ней записываются не в строчку текстом, а в графическом, удобном для человека, виде (по аналогии с системой Mathcad).

Для решения ОДУ в SMath Studio реализованы следующие методы:

– неявный метод Эйлера (2-го порядка),
– явный метод Рунге-Кутты (классический 4-го порядка).

Ограниченное количество методов для решения ОДУ не позволяет использовать её для решения жёстких задач и задач химичесой кинетики в частности.

В дополнение предыдущей программы можно посмотреть библиотеку Intel ODE Solver Library [4]. Библиотека написана на языке C со всеми вытекающими отсюда зависимостями. Доступны 32- и 64-разрядные версии.

В этой библиотеке следует выделить следующие программы и функции:

– rkm9st() — функция для решения нежёстких и средне-жёстких систем ОДУ с использованием явного метода, который основан на методе Мерсона 4-го порядка и многоступенчатом методе 1-го порядка, включающем до 9 этапов с контролем устойчивости;
– mk52lfn() — специализированная процедура для решения жёстких систем ОДУ с использованием неявного метода, основанного на L-стабильном (5,2)-методе с числовой матрицей Якоби, которая вычисляется с помощью процедуры;
– mk52lfa() — специализированная программа для решения жёстких систем ОДУ с использованием неявного метода, основанного на L-stable (5,2)-методе с численным или аналитическим вычислением матрицы Якоби. Пользователь должен предоставить процедуру для этого вычисления;
– rkm9mkn() — специализированная процедура для решения систем ОДУ с переменной или априори неизвестной жёсткостью,

автоматически выбирает явную или неявную схему на каждом шаге и при необходимости вычисляет числовую матрицу Якоби;

– rkm9mka() — специализированная подпрограмма для решения систем ОДУ с переменной или априори неизвестной жёсткостью, автоматически выбирает явную или неявную схему на каждом шаге. Пользователь должен предоставить процедуру для численного или аналитического вычисления матрицы Якоби.

У данной библиотеки есть хороший набор для решения нежёстких задач, есть автоматический выбор явного или неявного шага на каждой итерации, однако для решения жёстких задач требуется дополнительно вводить Якобиан, что достаточно проблематично.

GNU Scientific Library (или GSL) — это библиотека, написанная на языке программирования C для численных вычислений в прикладной математике и науке. GSL является частью проекта GNU (Массачусетский технологический институт, США) и распространяется на условиях лицензии GPL [5]. Первый релиз состоялся в 1996 году.

GSL используется, в частности, в таком программном обеспечении, как PSPP и Perl Data Language.

Следует отметить следующие достаточно хорошо известные явные методы:

– rk2() — явный метод Рунге-Кутты (2, 3 порядка),
– rk4() — явный 4-й порядок (классический) Рунге-Кутты. Оценка погрешности осуществляется методом удвоения шага,
– rkf45() — явный метод Рунге-Кутты-Фалберга (4, 5 порядка),
– rkck() — явный метод Рунге-Кутты Кэш-Карпа (4, 5 порядка),
– rk8pd() — явный метод Рунге-Кутты Дормана-Принса (4, 5 порядка).

Среди неявных методов можно отметить методы, требующие построение Якобиана вручную и большой набор многошаговых методов типа Адамса:

– rk1imp() — неявный гауссовский метод Рунге-Кутты первого порядка (метод Эйлера). Оценка погрешности осуществляется методом удвоения шага. Для этого алгоритма требуется якобиан;
– rk2imp() — неявный гауссовский метод Рунге-Кутты второго порядка (неявное правило средней точки). Оценка погрешности осуществляется методом удвоения шага. Для этого шагового двигателя требуется якобиан;

– rk4imp() — неявный гауссов Рунге-Кутта 4-го порядка. Оценка погрешности осуществляется методом удвоения шага. Для этого алгоритма требуется якобиан;
– bsimp() — неявный метод Булирша-Стоера (многошаговый). Этот метод, как правило, подходит для несложных задач с небольшой жёсткостью и на финальных этапах решения. Для этого шагового двигателя требуется якобиан;
– msadams() — линейный многоступенчатый метод Адамса с переменным коэффициентом в форме Nordsieck. Этот шаговый процессор использует явные методы Адамса-Башфорта (предсказатель) и неявные методы Адамса-Моултона (корректор) в режиме функциональной итерации $P(EC)^m$. Порядок методов динамически варьируется от 1 до 12;
– msbdf() — метод линейной многоступенчатой формулы обратного дифференцирования с переменным коэффициентом (BDF) в форме Nordsieck. Этот шаговый преобразователь использует явную формулу BDF в качестве предиктора и неявную формулу BDF в качестве корректора. Для решения СНУ используется модифицированный итерационный метод Ньютона. Порядок методов динамически варьируется от 1 до 5. Этот метод, как правило, подходит для сложных задач. Для этого шагового двигателя требуется якобиан.

В данной библиотеке большое количество как явных, так и неявных методов, но для невных методов как и в библиотеке Intel ODE Solver Library требуется самостоятельное вычисление Якобиана. Недостаток именно библиотечного вида заключается в том, что надо писать модули обращения и модули обработки результата.

Следующей библиотекой для рассмотрения является DotNumerics [6]. Она включает в себя числовую библиотеку для .NET. Библиотека написана на чистом C# и содержит более 100 000 строк кода с самыми передовыми алгоритмами для линейной алгебры, ДУ и задач оптимизации. Библиотека включает CSLapack, Cabelas и CSEispack, эти библиотеки являются переводом с Fortran на C# LAPACK, BLAS и EISPACK соответственно.

Основные решатели для нежёстких систем:

– ExplicitRK45() — решает задачу с начальными значениями для

нелинейных обыкновенных ДУ, используя явный метод Рунге-Кутты порядка 4, 5;

– AdamsMoulton() — решает начальную задачу для нелинейных обыкновенных ДУ с использованием метода Адамса-Моултона.

Решатели для жёстких систем:

– ImplicitRK5() — решает начальную задачу для жёстких обыкновенных ДУ с использованием неявного метода Рунге-Кутты порядка 5,
– GearsBDF() — решает начальную задачу для жёстких обыкновенных ДУ с использованием многошагового метода BDF Gear.

В данной библиотеке представлено небольшое число методов решения ОДУ до 5 порядка точности. В основном эта библиотека обслуживает не только задач ДУ, но и задачи линейной алгебры, в том числе задачи оптимизации, поэтому она содержит мало методов для решения поставленной задачи.

Отметим так же систему Wolfram Alpha [7]. Это база знаний и набор вычислительных алгоритмов, вопросно-ответная система.

Wolfram Alpha способен переводить данные между различными единицами измерения, системами счисления, подбирать общую формулу последовательности, находить возможные замкнутые формы для приближенных дробных чисел, вычислять суммы, пределы, интегралы, решать уравнения и системы уравнений, производить операции с матрицами, определять свойства чисел и геометрических фигур. Отдельного внимания стоит парсинг математических выражений.

В частности система Wolfram Alpha способна решать ОДУ стандартными явными методами:

– метод Эйлера (2-го порядка),
– метод средних точек (модифицированный метод Эйлера),
– неявный метод средних точек,
– метод Хойна, вариант модифицированного метода Эйлера,
– метод Рунге-Кутты 3-го порядка,
– метод Рунге-Кутты (классический 4-го порядка),
– метод Рунге-Кутты-Фалберга,
– метод Дормана-Принса.

Система Wolfram Alpha является облачной. Это означает, что для

работы с ней необходимо постоянное подключение к интернету. Ещё одним недостатком является наличие платного функционала и отсутствие готовых модулей для создания и решения СДУ химической кинетики.

Несмотря на большое число программ и библиотек для решение ДУ, появилась потребность в создании программы с большим выбором расчётных схем для решения, в то числе и, жёстких систем ДУ для задач химической кинетики.

Главным недостатком реализации рассмотренных вычислительных методов в известных программных комплексах, является возможная выдача ошибочных результатов для жёстких систем ОДУ без предупреждения пользователей об их недостоверности.

4 КЛАССИЧЕСКИЕ МЕТОДЫ ДЛЯ РЕШЕНИЯ ОДУ

4.1 Постановка задачи

Программа, над которой ведётся работа, предназначена для решения ДУ или СДУ с начальными значениями, то есть задачи Коши — классической математической постановки. Сама по себе задача достаточно сложная и изучалась уже более ста лет.

Конкретная прикладная задача сводится к решению ДУ произвольного порядка. Общий вид такой задачи представлен в примере (1).

$$\begin{cases} y^{(n)} = f(x, y, y', y'', ..., y^{(n-1)}) \\ y(x_0) = y_0 \\ y'(x_0) = y_1 \\ y''(x_0) = y_2 \\ ... \\ y^{(n-1)}(x_0) = y_{n-1} \end{cases} \tag{1}$$

Данное уравнение произвольного порядка n может быть преобразовано в систему из n ДУ первого порядка путём замены переменных. Пример (2) демонстрирует преобразование задачи Коши второго порядка в систему из 2-х уравнений первого порядка, путём замены y' на z:

$$\begin{cases} z' = f(x, y, y', y'') \\ y' = z \\ y(x_0) = y_0 \\ z(x_0) = y_1 \end{cases} \tag{2}$$

Из курса ДУ известно, что задача с начальными условиями при непрерывных правых частях, удовлетворяющих условию Липшица по всем переменным, имеет единственное решение [8; 9].

Методы решения ДУ можно классифицировать на точные, приближенные и численные. Точные методы, которые изучаются в курсе ДУ и могут быть применены к очень ограниченному кругу уравнений, позволяют выразить решение ДУ либо через элементарные функции, либо с помощью

квадратур от элементарных функций. К приближенным методам относятся приемы, в которых решение ДУ получается, как предел некоторой последовательности, элементы которой построены с помощью элементарных функций. Численные методы представляют собой алгоритмы вычисления приближенных значений искомой функции в узлах.

Такие задачи могут отличаться между собой сложностью решения. Так одни задачи можно решать при помощи группы явных методов и получать достаточно точное решение. Другие же задачи, в которых присутствует резкий скачок градиента функции, решать приходится с использованием неявных схем, так как у них повышенные требования к устойчивости схем решения [10]. Такие задачи называются жёсткими.

4.2 Жёсткость

Будем считать линейную систему обыкновенных ДУ жёсткой, если для уравнения вида

$$u' = Au, \tag{3}$$

где A — постоянная матрица $n \times n$, выполняются следующие требования:

а) все собственные числа λ_i матрицы A имеют отрицательную действительную часть, т. е. $Re\lambda_i < 0$, $i = 1, 2, ..., n$;

б) число S, вычисляемое по формуле (4) велико.

$$S = \frac{\max\limits_{1 \le k \le n} |Re\lambda_k|}{\min\limits_{1 \le k \le n} |Re\lambda_k|} \tag{4}$$

Число S называется жёсткостью задачи [11; 12]. Для жёстких задач это число должно быть намного больше единицы, однако чёткой границы между жёсткой и нежёсткой задачей нет. Для нежёстких систем значение шага по времени ошраничивается в первую очередь желаемой точностью решения. Для жёстких же значение шага ограничивается устойчивостью. Так модельные уравнения с числом жёсткости более 100 уже дают небольшие скачки погрешности решения. Рассмотрим таблицу 1.

Таблица 1 – Классификация коэффициентов жёсткости

Классификация	Число жёсткости	Рекомендуемые методы
Умеренно жёсткая	$S = O(10)$	Явные или вложенные 2-4 порядок
Средне жёсткая	$S = O(10^2)$	Явные или вложенные 4-6 порядок
Сильно жёсткая	$O(10^2) \leq S \leq O(10^5)$	Вложенные 6 порядка, неявные 2-4 порядка
Экстремально жёсткая	$O(10^6) \leq S \leq O(10^8)$	Неявные 4-6 порядка
Патологически жёсткая	$S \geq O(10^9)$	Неявные 6 порядка

Разные источники предлагают свою классификацию задач в зависимости от числа жёсткости. Так в [1] в соответствии с величиной S задачу можно классифицировать как умеренно жёсткую, средне жёсткую, сильно жёсткую и так далее, что показано в таблице 1.

В задачах химической кинетики число жёсткости может быть более 10^6.

4.3 Явные схемы для решения ОДУ

Для решения жёстких и нежёстких задач можно использовать различные семейства методов. В данной работе будет рассмотрено семейство одношаговых методов Рунге-Кутты [13].

В семейство методов Рунге-Кутты входит огромное число схем, как явных, так и неявных. Все эти методы представлены в виде таблиц Бутчера [10; 14]. Общий вид явных схем представлен на рисунке 4.

Явные методы обладают нижней диагональной формой таблицы Бутчера и позволяют решать задачи обычными маршевыми методами. В связи с тем, что явные методы являются условно устойчивыми, работа с ними сильно зависит от размера шага итегрирования и для достижения заданной точности требуют достаточно мелкий шаг интегрирования и повторного вычисления для повышения порядка точности процедурой Рунге-Ромберга.

$$\begin{cases} y_{k+1} = y_k + \Delta y_k \\ \Delta y_k = \sum\limits_{i=1}^{s} b_i K_i^k \\ K_i^k = hf(x_k + c_i h, y_k + h \sum\limits_{j=1}^{i-1} a_{ij} K_j^k) \end{cases}$$

c_1	0	0	0		0	0
c_2	a_{21}	0	0		0	0
c_3	a_{31}	a_{32}	0		0	0
c_s	a_{s1}	a_{s2}	a_{s3}		a_{ss-1}	0
	b_1	b_2	b_3		b_{s-1}	b_s

Рисунок 4 – Общий вид явных схем

Все явные методы являются условно устойчивыми. В литературе [10] можно подобрать целую коллекцию одношаговых маршевых многоэтапных методов, которые обеспечивают адекватную точность и подходят для решения нежёстких ОДУ. Среди этих методов можно выделить классический метод Рунге-Кутты 4-го порядка, схема котрого представлена на рисунке 5.

$$\begin{cases} y_{k+1} = y_k + \Delta y_k \\ \Delta y_k = \frac{1}{6}(K_1^k + 2K_2^k + 2K_3^k + K_4^k) \\ K_1^k = hf(x_k, y_k) \\ K_2^k = hf(x_k + \frac{1}{2}h, y_k + \frac{1}{2}K_1^k) \\ K_3^k = hf(x_k + \frac{1}{2}h, y_k + \frac{1}{2}K_2^k) \\ K_4^k = hf(x_k + h, y_k + K_3^k) \end{cases}$$

0	0	0	0	0
$\frac{1}{2}$	$\frac{1}{2}$	0	0	0
$\frac{1}{2}$	0	$\frac{1}{2}$	0	0
1	0	0	1	0
0	$\frac{1}{6}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{6}$

Рисунок 5 – Схема Рунге-Кутты 4-го порядка

Для более жёстких задач метод Рунге-Кутты 4-го порядка может быть недостаточно точным. Иногда приходится сильно уменьшать шаг интегрирования для того, чтобы решение было устойчивым. Другой путь — использование методов более высоких порядков, например метода Рунге-Кутты 6-го порядка. Схема данного метода представлена на рисунке 6.

$$
\begin{cases}
y_{k+1} = y_k + \Delta y_k \\
\Delta y_k = \frac{7}{90}(K_1^k + K_6^k) + \frac{16}{45}(K_2^k + K_5^k) - \\
-\frac{1}{3}K_3^k + \frac{7}{15}K_4^k \\
K_1^k = hf(x_k, y_k) \\
K_2^k = hf(x_k + \frac{1}{4}h, y_k + \frac{1}{4}K_1^k) \\
K_3^k = hf(x_k + \frac{1}{2}h, y_k + \frac{1}{2}K_1^k) \\
K_4^k = hf(x_k + \frac{1}{2}h, y_k + \frac{1}{7}K_1^k + \frac{2}{7}K_2^k + \\
+\frac{1}{14}K_3^k) \\
K_5^k = hf(x_k + \frac{3}{4}h, y_k + \frac{3}{8}K_1^k - \frac{1}{2}K_3^k + \\
+\frac{7}{8}K_4^k) \\
K_6^k = hf(x_k + h, y_k - \frac{4}{7}K_1^k + \frac{12}{7}K_2^k - \\
-\frac{2}{7}K_3^k - K_4^k + \dfrac{8}{7}K_5^k)
\end{cases}
$$

0	0	0	0	0	0	0
$\frac{1}{4}$	$\frac{1}{4}$	0	0	0	0	0
$\frac{1}{2}$	$\frac{1}{2}$	0	0	0	0	0
$\frac{1}{2}$	$\frac{1}{7}$	$\frac{2}{7}$	$\frac{1}{14}$	0	0	0
$\frac{3}{4}$	$\frac{3}{8}$	0	$-\frac{1}{2}$	$\frac{7}{8}$	0	0
1	$-\frac{4}{7}$	$\frac{12}{7}$	$-\frac{2}{7}$	-1	7	0
0	$\frac{7}{90}$	$\frac{16}{45}$	$-\frac{1}{3}$	$\frac{7}{15}$	$\frac{16}{45}$	$\frac{7}{90}$

Рисунок 6 – Схема Рунге-Кутты 6-го порядка

Все явные методы относятся к классу условно устойчивых методов. Точность решения условно устойчивых методов сильно зависит от размера шага, поэтому у каждого из них есть свой критерий устойчивости.

4.4 Явные вложенные схемы для решения ОДУ

В связи с перечисленными недостатками обычных явных схем, есть смысл применить явные вложенные схемы, которые базируются так же на нижней треугольной матрице Бутчера, обладают маршевым методом решения и позволяют на базе одних и тех же поправочных коэффициентов моделировать решение с разным порядком точности и тем самым либо увеличивать шаг интегрирования, либо уменьшать [15; 14]. Общий вид этих схем отличается от явных лишь наличием дополнительной строки в таблице Бутчера, по которая и позволяет моделировать решение с другим порядком точности для сравнения погрешности. На рисунке 7 представлен общий вид явных вложенных схем.

$$\begin{cases} y_{k+1} = y_k + \sum\limits_{i=1}^{s} b_i K_i^k \\ y^*_{k+1} = y_k + \sum\limits_{i=1}^{s} b^*_i K_i^k \\ K_i^k = hf(x_k + c_i h, y_k + h \sum\limits_{j=1}^{i-1} a_{ij} K_j^k) \end{cases}$$

c_1	0	0	0		0	0
c_2	a_{21}	0	0		0	0
c_3	a_{31}	a_{32}	0		0	0
c_s	a_{s1}	a_{s2}	a_{s3}		a_{ss-1}	0
	b_1	b_2	b_3		b_{s-1}	b_s
	b^*_1	b^*_2	b^*_3		b^*_{s-1}	b^*_s

Рисунок 7 – Общий вид явных вложенных схем

Порядок точности таких схем записывется в виде $n(m)$, где n — порядок схемы при использовании верхней строки коэффициентов b, m — нижней. Если разница в решениях при использовании этих порядков высока, то шаг следует уменьшить, если же наоборот — разница очень низкая, то шаг можно увеличить для ускорения работы алгоритма. Пример схемы для метода Фалберга 2(3)-го порядка отображён на рисунке 8.

$$\begin{cases} y_{k+1} = y_k + \frac{1}{2}K_1^k + \frac{1}{2}K_2^k \\ y^*_{k+1} = y_k + \frac{1}{2}K_1^k + \frac{1}{2}K_2^k \\ K_1^k = hf(x_k, y_k) \\ K_2^k = hf(x_k + h, y_k + K_1^k) \\ K_3^k = hf(x_k + \frac{1}{2}h, y_k + \frac{1}{4}K_1^k + \frac{1}{4}K_2^k) \\ R = |y_{k+1} - y^*_{k+1}| \end{cases}$$

0	0	0	0
1	1	0	0
$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{4}$	0
0	$\frac{1}{2}$	$\frac{1}{2}$	0
0	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{4}{6}$

Рисунок 8 – Схема Фалберга 2(3)-го порядка

Среди вложенных методов выделяется Дорман-Принс 4(5)-го порядка, который является наиболее популярным в этой группе методов [16]. Таблица 2 для метода Дормана-Принса обладает большим размером и состоит в основном из громоздких дробей. Стоит отметить, что последний этап этого метода рассчитывается в той же точке, что и первый этап следующего шага, что позволяет сэкономить немного времи на вычислениях.

Таблица 2 – Таблица Бутчера для метода Дормана-Принса 4(5)-го порядка

0	0	0	0	0	0	0	0
$\frac{1}{5}$	$\frac{1}{5}$	0	0	0	0	0	0
$\frac{3}{10}$	$\frac{3}{40}$	$\frac{9}{40}$	0	0	0	0	0
$\frac{4}{5}$	$\frac{44}{45}$	$-\frac{56}{15}$	$\frac{32}{9}$	0	0	0	0
$\frac{8}{9}$	$\frac{19372}{6561}$	$-\frac{25360}{2187}$	$\frac{64448}{6561}$	$-\frac{212}{729}$	0	0	0
1	$\frac{9017}{3168}$	$-\frac{355}{33}$	$\frac{46732}{5247}$	$\frac{49}{176}$	$-\frac{5103}{18656}$	0	0
1	$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{192}$	$-\frac{2187}{6784}$	$\frac{11}{84}$	0
0	$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{192}$	$-\frac{2187}{6784}$	$\frac{11}{84}$	0
0	$\frac{5179}{57600}$	0	$\frac{7571}{16695}$	$\frac{393}{640}$	$-\frac{92097}{339200}$	$\frac{187}{2100}$	$\frac{1}{40}$

Разработано множество вложенных многоэтапных методов, но для решения сложных задач представляют интерес методы, обладающие минимальным числом этапов и максимальным порядком точности.

4.5 Неявные схемы для решения ОДУ

К наиболее сложным методам можно отнести группу неявных схем. Плотно заполненная таблица Бутчера не позволяет использовать маршевые методы и принуждает решать систему алгебраических уравнений на каждом шаге интегрирования, что вызывет определённые сложности у разработчиков [17; 18; 14; 19]. Общий вид неявных схем продемонстрирован на рисунке 9.

$$\begin{cases} y_{k+1} = y_k + \sum\limits_{i=1}^{s} b_i K_i^k \\ K_i^k = hf(x_k + c_i h, y_k + h \sum\limits_{j=1}^{s} a_{ij} K_j^k) \end{cases}$$

c_1	a_{11}	a_{12}	a_{13}		a_{1s-1}	a_{1s}
c_2	a_{21}	a_{22}	a_{23}		a_{2s-1}	a_{2s}
c_3	a_{31}	a_{32}	a_{33}		a_{3s-1}	a_{3s}
c_s	a_{s1}	a_{s2}	a_{s3}		a_{ss-1}	a_{ss}
	b_1	b_2	b_3		b_{s-1}	b_s

Рисунок 9 – Общий вид неявных схем

Особенностью этих методов является то, что в качестве базовых опорных точек являются иррациональные корни многочлена Лежанра. На рисунке 10 показана схема неявного метода Гаусса 4-го порядка.

$$\begin{cases} y_{k+1} = y_k + \frac{1}{2}hK_1 + \frac{1}{2}hK_2 \\ K_1 = f(x_k + h(\frac{1}{2} - \frac{\sqrt{3}}{6}), y_k + \frac{1}{4}hK_1 + \\ +(\frac{1}{4} - \frac{\sqrt{3}}{6})hK_2) \\ K_2 = f(x_k + h(\frac{1}{2} + \frac{\sqrt{3}}{6}), y_k + (\frac{1}{4} + \frac{\sqrt{3}}{6})hK_1 + \\ +\frac{1}{4}hK_2) \end{cases}$$

$\frac{1}{2} - \frac{\sqrt{3}}{6}$	$\frac{1}{4}$	$\frac{1}{4} - \frac{\sqrt{3}}{6}$
$\frac{1}{2} + \frac{\sqrt{3}}{6}$	$\frac{1}{4} + \frac{\sqrt{3}}{6}$	$\frac{1}{4}$
0	$\frac{1}{2}$	$\frac{1}{2}$

Рисунок 10 – Схема неявного Гаусса 4-го порядка

По схеме видно, что для нахождения коэффициентов K_i обычные маршевые методы не подходят. Для их получения нужно использовать итерационные методы решения СНУ, такие как метод простой итерации, метод Зейделя или метод Ньютона. Таблицы Бутчера неявных методов обладают меньшим размером по сравнению с явными, сохраняя тот же порядок точности. Хорошим примером копактной записи послужит таблица 3, неявного метода Гаусса 6-го порядка.

Таблица 3 – Таблица Бутчера для метода неявного метода Гаусса 6-го порядка

$\frac{1}{2} - \frac{\sqrt{15}}{10}$	$\frac{5}{36}$	$\frac{2}{9} - \frac{\sqrt{15}}{15}$	$\frac{5}{36} - \frac{\sqrt{15}}{30}$
$\frac{1}{2}$	$\frac{5}{36} + \frac{\sqrt{15}}{24}$	$\frac{2}{9}$	$\frac{5}{36} - \frac{\sqrt{15}}{24}$
$\frac{1}{2} + \frac{\sqrt{15}}{10}$	$\frac{5}{36} + \frac{\sqrt{15}}{30}$	$\frac{2}{9} + \frac{\sqrt{15}}{15}$	$\frac{5}{36}$
0	$\frac{5}{18}$	$\frac{4}{9}$	$\frac{5}{18}$

Помимо перечисленных групп методов, существуют так же диагональные неявные методы, неявные вложенные, неявные методы без одной строки или столбца, но все они являются подгруппами неявных методов и используют тот же вид общей схемы. Отдельно можно уделить внимание диагональным методам или, как их ещё называют, полунеявным.

4.6 Диагональные схемы для решения ОДУ

Полунеявные или диагональные схемы входят в группу неявных методо, имеют нижнюю треугольную форму таблицы Бутчера и ненулевыми элементами диагонали [20; 21; 14]. Данная особенность позволяет решать системы уравнений для поиска K_i при помощи распараллеливания, что

гораздо быстрее на многоядерных процессорах, а так же с использованием не более одной итерации. К недостаткам таких схем относится плохая устойчивость по сравнению с обычными неявными.

В качестве примера таких схем можно привести Диагональный метод 4-го порядка, представленный на рисунке 11.

$$\begin{cases} y_{k+1} = y_k - hK_1 + \frac{3}{2}hK_2 - hK_3 + \frac{3}{2}hK_4 \\ K_1 = f(x_k + \frac{1}{2}h, y_k + \frac{1}{2}hK_1) \\ K_2 = f(x_k + \frac{2}{3}h, y_k + \frac{2}{3}hK_2) \\ K_3 = f(x_k + \frac{1}{2}h, y_k - \frac{5}{2}hK_1 + \frac{5}{2}hK_2 - \frac{1}{2}hK_3) \\ K_4 = f(x_k + \frac{1}{3}h, y_k - \frac{5}{3}hK_1 + \frac{4}{3}hK_2 - \frac{2}{3}hK_4) \end{cases}$$

$\frac{1}{2}$	$\frac{1}{2}$	0	0	0
$\frac{2}{3}$	0	$\frac{2}{3}$	0	0
$\frac{1}{2}$	$-\frac{5}{2}$	$\frac{5}{2}$	$\frac{1}{2}$	0
$\frac{1}{3}$	$-\frac{5}{3}$	$\frac{4}{3}$	0	$\frac{2}{3}$
0	-1	$\frac{3}{2}$	-1	$\frac{3}{2}$

Рисунок 11 – Схема Диагонального метода 4-го порядка

Подведём итоги. Преимущество явных методов заключается в производительности, так как им не нужно решать на каждом шаге системы алгебраических уравнений. Отсюда следует, что использование явных методов даёт больший выигрыш по времени, чем использование более производительного оборудования и распараллеливания. Главным же преимуществом неявных методов является наличие устойчивости (A-устойчивости, L-устойчивости и так далее), в результате чего полученное ими решение является гарантированно устойчивым, в отличие от явных. Для решения жёстких задач рекомендуется использовать неявные методы как раз потому что они обладают L-устойчивостью.

Однако некоторые СДУ, описывающие химические процессы можно решить и явными методами с требуемой точностью, потому что свойства A и L-устойчивости являются лишь достаточным, но не необходимым условием эффективности решения, поэтому нельзя использовать только те или иные методы. Выбрать оптимальный метод можно путём целевого тестирования.

4.7 Вспомогательные методы для работы неявных схем

Решение систем нелилейных уравнений

Теперь рассмотрим алгоритмы решения СНУ для итерационных процессов в неявных методах. В данной работе реализовано 3 таких

алгоритма:

- метод простой итерации,
- метод Зейделя,
- метод Ньютона.

Метод Зейделя и простой итерации являются достаточно быстрыми алгоритмами, так как им не нужно дифференцировать функции или вычислять Якобиан. Отличие метода Зейделя от метода простой итерации заключается в том, что при вычислении очередного приближения вектора неизвестных используются уже уточненные значения на этом же шаге итерации. Это обеспечивает более быструю сходимость метода Зейделя. Общий вид метода простой итерации и Зейделя для неявных методов представлен в формулах (5) и (6) соответственно.

$$K_i^{j+1} = f(x_k + c_i h, y_k + a_{i1} h K_1^j + a_{i2} h K_2^j + ...a_{ii} h K_i^j + ... + a_{is} h K_s^j) \qquad (5)$$

$$K_i^{j+1} = f(x_k + c_i h, y_k + a_{i1} h K_1^{j+1} + a_{i2} h K_2^{j+1} + ...a_{ii} h K_i^j + ... + a_{is} h K_s^j) \quad (6)$$

В качестве начальных приближений можно взять значения K с предыдущего шага. Для первого же шага можно посчитать значения K явными методами.

Более интересным для рассмотрения является метод Ньютона (7). Медленная скорость работы, обусловленная необходимостью строить матрицу Якоби на каждом шаге итерирования, позволяет получать решение устойчиво независимо от начального приближения K.

$$\begin{pmatrix} K_1^{j+1} \\ K_2^{j+1} \\ ... \\ K_s^{j+1} \end{pmatrix} = \begin{pmatrix} K_1^j \\ K_2^j \\ ... \\ K_s^j \end{pmatrix} - \begin{pmatrix} \frac{\partial F_1}{\partial K_1} & \frac{\partial F_1}{\partial K_2} & ... & \frac{\partial F_1}{\partial K_s} \\ \frac{\partial F_2}{\partial K_1} & \frac{\partial F_2}{\partial K_2} & ... & \frac{\partial F_2}{\partial K_s} \\ ... & & & \\ \frac{\partial F_s}{\partial K_1} & \frac{\partial F_s}{\partial K_2} & ... & \frac{\partial F_s}{\partial K_s} \end{pmatrix} \times \begin{pmatrix} F_1 \\ F_2 \\ ... \\ F_s \end{pmatrix} \qquad (7)$$

$$F_i = K_i^j - f(x_k + c_i h, y_k + a_{i1} h K_1^j + a_{i2} h K_2^j + ...a_{ii} h K_i^j + ... + a_{is} h K_s^j)$$

Несмотря на лучшую сходимость решения, в большинстве случаев для решения уравнений химической кинетики было достаточно использовать методы простой итерации и Зейделя, так как они быстрее. В случаях, когда их

было недостаточно, применялся метод Ньютона. Для построения матрицы Якоби сначала строятся функции F_i, после чего используются алгоритмы численного дифференцирования с различным порядком точности.

Дифференцирование

Дифференцирование — операция взятия полной или частной производной функции [22]. Формулы численного дифференцирования в основном используются при нахождении производных от функции вида $y = f(x)$, заданной таблично. Если функция задана не таблично, то можно просто взять её значения с некоторым, достаточно малым шагом. При решении практических задач, как правило, используются аппроксимации первых и вторых производных. В работе было реализовано 4 метода дифференцирования первого порядка: 2-х точечный, 3-х точечный и два 4-х точечных.

$$
\begin{aligned}
y'_i &= \frac{y_{i+1} - y_{i-1}}{2h} \\
y'_i &= \frac{-3y_i + 4y_{i+1} - y_{i+2}}{2h} \\
y'_i &= \frac{-y_{i+2} + 8y_{i+1} - 8y_{i-1} + y_{i-2}}{12h} \\
y'_i &= \frac{-11y_i + 18y_{i+1} - 9y_{i+2} + 2y_{i+3}}{6h}
\end{aligned}
\tag{8}
$$

Для метода Ньютона использовалась симметричная формула дифференцирования по 4-м точкам.

5 РЕАЛИЗАЦИЯ

5.1 Используемые технологии

Таким образов в программу вошли классические алгоритмы численных методов, такие как методы матричной алгебры, методы Ньютона и так далее. Для её написания были использованы *язык программирования C++*, *фреймворк QT* и автоматизатор сборок проектов *CMake*. Полный список технологий и алгоритмов предоставлен на рисунке 12. Для поиска ошибок и утечек памяти применилась связка из отладчика *GDB* и профилировщика *Valgrind*.

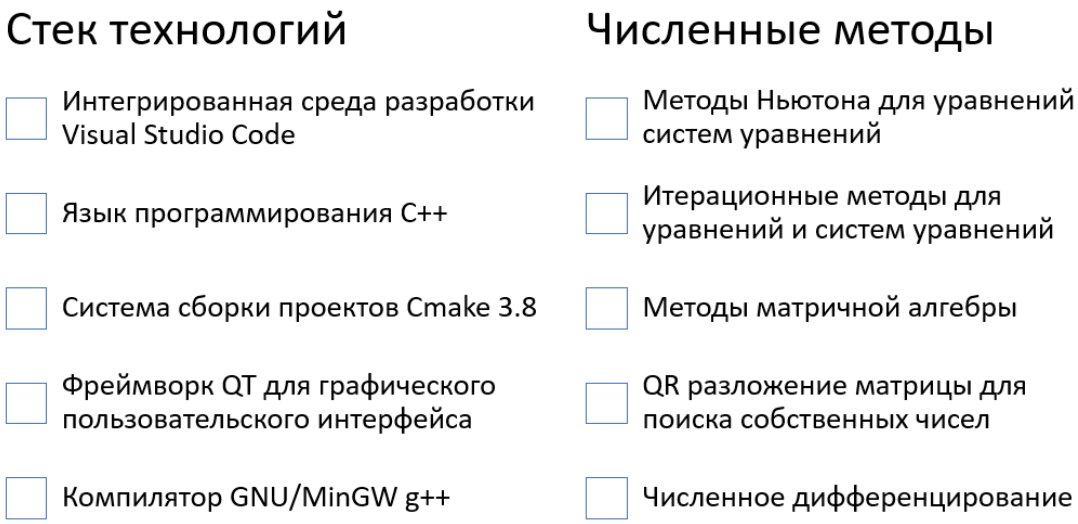


Рисунок 12 – Технологии и алгоритмы

Фреймворк QT нужен для разработки кроссплатформенного программного обеспечения с графическим интерфейсом [23]. Собственная среда *QT Designer* позволяет строить пользовательский графический интерфейс при помощи множества инстументов.

Программа для автоматизации сборок проектов CMake самой сборкой не занимается, а лишь генерирует файлы сборки из предварительно написанного файла сценария *CMakeLists.txt* и предоставляет простой единый интерфейс управления [24]. Возможность нескольких вариаций сборок с разными ключами компиляции, автоматизация процесса установки пакетов и кроссплатформенность делают CMake отличным выбором для разработки.

Отладчик GDB позволяет отслеживать различные ошибки при работе программы. GDB предлагает обширные средства для слежения и контроля за

выполнением исполняемых файлов [25]. При создании больших проектов может возникнуть множество неочевидных ошибок, которые приходится отлавливать благодаря специальным программам, к числу которых и относится GDB.

Если GDB нужен для поиска ошибок, связанных с переменными, то профилировщик Valgrind нужен для поиска ошибок связанных с памятью. К числу таких ошибок относятся: утечки памяти, ошибка сегментации, попытки использования неинициализированной памяти и так далее. Вся эта работа выполняется при помощи инструмента *Memcheck*. Так же Valgrind может предоставить инструменты для профилировки, такие как *Cachegrind*, *Callgrind* и так далее [26].

После рассмотрения используемых технологий можно перейти к алгоритму выполнения программы. Общий алгоритм работы, показан на рисунке 13.

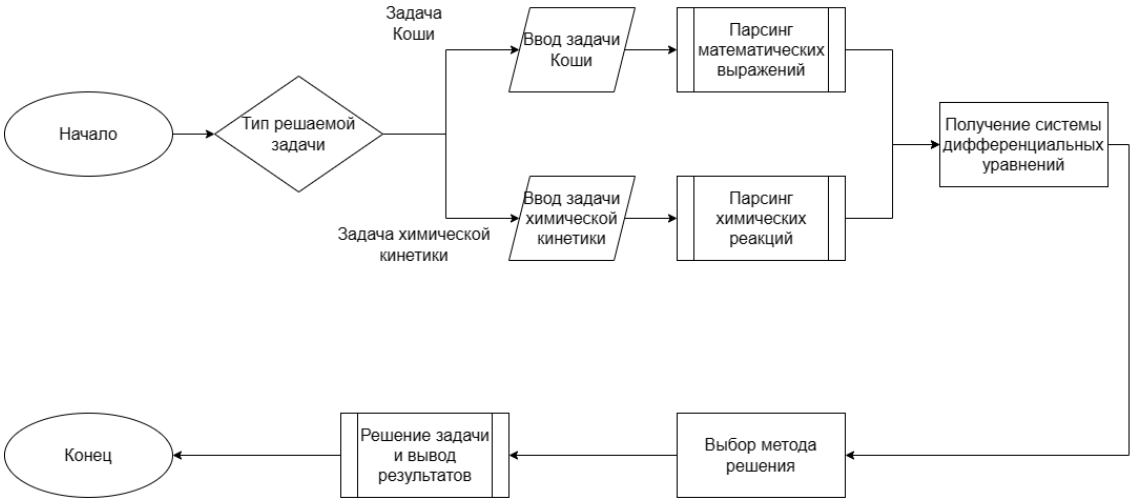


Рисунок 13 – Алгоритм работы программы

Сначала выбирается тип решаемой задача, после чего всплывает форма с соответствующими текстовыми полями для ввода. После этого, в зависимости от типа решаемой задачи, происходит обработка ввода и формирование СДУ, которые решаются выбранным методом. Результаты работы выводятся либо на экран, либо в отдельном файле.

Для решения ОДУ был реализован большой перечень как явных так и неявных методов с различным порядком точности. Для удобства разработки и тестирования, программа была поделена на множество отдельных модулей. Далее приведено детальное рассмотрение каждого из них.

5.2 Общие алгоритмы и структуры

В данном подразделе перечислены основные алгоритмы и структуры, которые использовались в остальных модулях. Сюда вошли алгоритмы LU-разложения матрицы, QR алгоритм нахождения собственных чисел матрицы, алгоритмы дифференцирования с разным порядком точности. Для работы с таблицами Бутчера и алгоритмами решения СНУ был реализован собственный класс матрицы с базовой матричной алгеброй.

Матрица

Реализация матрицы в данной работе представляет из себя класс, содержащий одномерный массив $n \times m$, где n, m — размеры матрицы. В комплект к ней входят простейшие оперции матричной алгебры, такие как умножение, сложение, возведение в степень и так далее. В дополнение к этим операциям были реализованы алгоритм нахождения обратной матрицы и определителя при помощи LU-разложения.

Матрица используется для хранения таблиц Бутчера, вычисления Якобиана, построения матрицы системы уравнений и дальнейшее вычисление числа жёсткости и так далее. Дополнительно были реализованы возможность проверки матрицы на симметричность, квадратность и операции добавления/удаления строк/столбцов.

LU алгоритм

LU-разложение матрицы A представляет собой разложение матрицы A в произведение нижней и верхней треугольных матриц, т.е. $A = LU$, где L — нижняя треугольная матрица на диагонали которой стоят единицы, U — верхняя треугольная матрица. LU–разложение может быть использовано при решении систем линейных алгебраических уравнений вида $Ax = b$. Полученное LU-разложение может быть так же использовано для вычисления определителя матрицы по формуле.

$$det(A) = det(L)det(U) = (\prod_{i=1}^{n} L_{ii})(\prod_{i=1}^{n} U_{ii}) = \prod_{i=1}^{n} U_{ii}, \tag{9}$$

где n — размер квадратной матрицы.

Для нахожения обратной матрицы из задачи $AX = E$, где A — исходная матрица, X — обратная матрица, E — единичная матрица, составляется n

систем уравнений:

$$AX_i = e_i, i = 1, ..., n, \tag{10}$$

где X_i — вектор-столбец обратной матрицы с индексом i, e_i — вектор-столбец единичной матрицы с индексом i.

Нахождение обратной матрицы сводится к решению n уравнений с одной матрицей и разными правыми частями.

QR алгоритм

При решении полной проблемы собственных значений для несимметричных матриц эффективным является подход, основанный на приведении матриц к подобным, имеющим треугольный или квазитреугольный вид. Одним из наиболее распространенных методов этого класса является QR-алгоритм.

В основе QR-алгоритма лежит представление матрицы в виде

$$A = QR, \tag{11}$$

где - Q, ортогональная матрица ($Q^{-1} = Q^T$), а R — верхняя треугольная. Такое разложение существует для любой квадратной матрицы. Одним из возможных подходов к построению QR разложения является использование преобразования Хаусхолдера. Преобразование Хаусхолдера осуществляется с использованием матрицы Хаусхолдера, имеющей следующий вид:

$$H = E - \frac{2}{\nu^T \nu} \nu \nu^T, \tag{12}$$

где E — единичная матрица, ν — произвольный ненулевой вектор-столбец. Само преобрахование имеет вид

$$A_{i+1} = H_i A_i \tag{13}$$

Матрица Q является произведением матриц Хаусхолдера, полученных на каждом шаге. Матрица R, как A_{n-1}.

$$Q = H_1 H_2 ... H_{n-1}, R = A_{n-1} \tag{14}$$

Код реализации алгоритма продемонстрирован на рисунке 14.

```
1  template <class T>
2  T getMainEl (const Matrix<T> &matrix, uint64_t i) {
3      T ans = matrix(i, i), tmp = 0;
4      T sign = ans >= T(0) ? T(1) : T(-1);
5      for (uint64_t j = i; j < matrix.size().n; ++j) {
6          tmp += matrix(j, i) * matrix(j, i);
7      }
8      return ans + sign * std::sqrt(tmp);
9  }
10
11 template <class T>
12 std::tuple<Matrix<T>, Matrix<T>> QR (const Matrix<T> &matrix) {
13     uint64_t n = matrix.size().n;
14     Matrix<T> Q(n), R(matrix), v(n, 1);
15     auto pred = [] (T el) -> bool {
16         return el == T(0);
17     };
18     for (uint64_t i = 0; i < n - 1; ++i) {
19         for (uint64_t j = 0; j < i; ++j)
20             v(j, 0) = 0;
21         v(i, 0) = getMainEl(R, i);
22         for (uint64_t j = i + 1; j < n; ++j) {
23             v(j, 0) = R(j, i);
24         }
25         if (std::count_if(v.toVector().cbegin(), v.toVector().cend
       (), pred) == n) {
26             ++i;
27             continue;
28         }
29         Matrix<T> H = Matrix<T>(n) - 2 * ((v * v.transp()) / (v.
       transp() * v)(0, 0));
30         R = H * R;
31         Q = Q * H;
32     }
33     for (uint64_t i = 0; i < n; ++i)
34         for (uint64_t j = 0; j < i; ++j)
35             R(i, j) = T(0);
36     return std::make_tuple(Q, R);
37 }
```

Рисунок 14 – Код QR алгоритма

Дифференцирование

Для численного дифференцирования был разработана функция, получающая получающая в качестве аргументов функция для дифференцирования, точку, в которой необходимо продифференцировать функцию и схему дифференцирования. Как можно заметить из формул (8), все эти схемы являются достаточно похожими, поэтому появилась идея представления этих схем в следующем виде:

$$y' = \frac{a_1 y_{i-n+1} + a_2 y_{i-n+2} + ... + a_{2n-1} y_{i+n-1}}{ch^p} \tag{15}$$

где n — количество точек аппроксимации.

При таком представлении все схемы можно реализовать при помощи одномерного массива коэффициентов.

$$[a_1, a_2, ..., a_{2n-1}, c, p]$$

где a_i — коэффициенты для точек, c — коэффициент перед шагом в знаменателе, p — степень шага. Такой подход позволяет быстро дополнять текущий набор схем новыми. Помимо этого избегается дублирование одинакого кода. Так, например, вторая 4-х точечная схема

$$y'_i = \frac{-11y_i + 18y_{i+1} - 9y_{i+2} + 2y_{i+3}}{6h} \tag{16}$$

может быть представлена в следующем виде:

$$[0, 0, 0, -11, 18, -9, 2, 6, 1]$$

Вычисление жёсткости

Для вычисления числа жёсткости по формуле (4) нужно построить матрицу Якоби для системы и найти её собственные числа.

Коэффициентом жёсткости задачи называется отношение максимального модуля действительной части собственных чисел матрицы Якоби к минимальной при условии, что все действительные части собственных чисел меньше нуля. Собственные числа данной матрицы находятся при помощи алгоритма QR-разложения. Для построения данной матрицы используется схема дифференцирования по 4-м точкам.

5.3 Графический пользовательский интерфейс

Для ввода задачи была реализована специальная форма с использованием фреймворка QT. Её внешний вид представлен на рисунке 15. Преобразование строк в функции происходит при помощи парсера математических выражений.

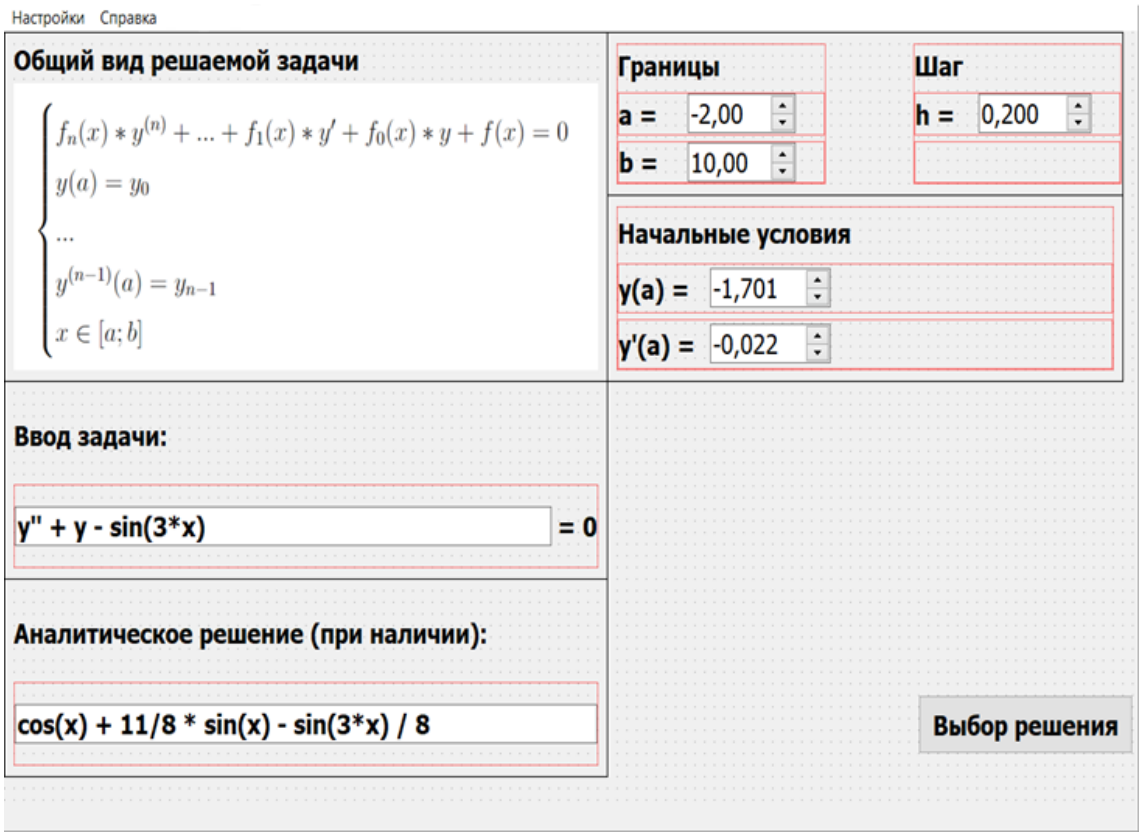


Рисунок 15 – Вид интерфейса для ввода задачи Коши

После ввода задачи, аналитического решения (при наличии), начальных условий и границ интегрирования требуется выбрать метод решения. Перед выбором решения идёт анализ задачи на жёсткость и если число жёсткости задачи велико (> 100), то выбор методов ограничивается явными высоких порядков, неявными и вложенными, иначе выбор методов не ограничивается.

Результаты вичислений либо сохраняются в отдельном файле, либо выводятся на экран.

5.4 Парсер математических выражений

При режиме работы с задачей Коши от пользователя требуется ввести задачу и аналитическое решение (при наличии).

Для обработки строк с задачей был реализован парсер математических

выражений. Принцип его работы следующий: проводится лексический анализ строки, описывающей функцию, и строится дерево математических выражений, содержащее функции и операторы математических выражений в качестве узлов и числовые константы с переменными в качестве листьев.

Реализация в виде дерева была выбрана по нескольким причинам:

– лёгкость в написании,
– простота в расширении функционала,
– некоторые операции (поиск коэффициентов при переменной, выделение поддерева и т.д.) легче выполнять при работе с деревом.

Интерфейс парсера позволяет использовать его как функтор, принимающий либо одну переменную, либо массив переменных. Реализована поддержка базовых функций и числовых констант. Так же есть возможность добавления пользовательских параметров и констант.

Для сравнения эффективности реализации было проведено тестирование на модельных функциях с использованием парсера и обычных статических функций. Сравнения по времени приведены в таблице 4.

Таблица 4 – Таблица сравнения статических функций и парсера

Уравнение	Статическая функция (ms)	Парсер (ms)
$x + 2$	18	35
$\sin(\cos(x)) + \cos(\cos(x))$	253	492
$\sin(-x) + \ln(e^{10}) + \arccos((-1)^x)$	623	1007
$\frac{11 - x^3(3x - 8)}{12(x - 2)^2(x - 3)}$	23	172
$x \exp(\frac{1}{x})$	29	109
$\exp(x \sin(\ln(x)) + x)$	393	385
$\frac{x^3}{4} - \frac{1}{x}$	22	108
$1 + \ln(abs(x))$	40	65
$\cos(\sqrt{4x}) + \sin(\sqrt{4x})$	303	349
$abs(x)^{\frac{3}{2}}$	249	293

На данной таблице видно, что парсер уступает по скорости обычным функциям в 1.5-2 раза. Это связано с тем, что при работе с деревом приходится проходить по указателям к нижним узлам, что сильно замедляет

работу алгоритма. Так же была протестирована реализация парсинга математических выражений при помощи обратной польской записи, однако она давала небольшой выигрыш по времени при урезанном функционале.

Рисунок 16 показывает листинг с примерами использования парсера.

```
1  #include <General/FuncMaker.hpp>
2
3  int main () {
4      //массив аргументов
5      std::vector<std::string> args;
6      FuncMaker func;
7
8      //пример для функции от го1- аргумента
9      args = {"x"};
10     //построение функции при помощи х2- аргументов: строки функции
       и массива аргументов
11     //в вычислениях используется числовая константа pi
12     func.reset("sin(x + pi) * x", args);
13     //вывод значения функции при x = 5
14     std::cout << "Func(5) = " << func(5) << "\n";
15
16     //пример для функции от х2- аргументов
17     args = {"x", "y"};
18     //установка параметра a = 10
19     func.setValue("a", 10.0);
20     //построение функции при помощи х2- аргументов: строки функции
       и массива аргументов
21     func.reset("x^2 + a*x*y + y^2", args);
22     //вывод значения функции в точке (2, 4)
23     std::cout << "Func(2, 4) = " << func({2, 4}) << "\n";
24     return 0;
25 }
```

Рисунок 16 – Пример использования парсера

В дальнейшем планируется добавление возможность аналитического дифференцирования функций, добавление пользовательских функций от 2-х и более перменных и дальнейшая оптимизация скорости работы парсера.

5.5 Парсер химичесских уравнений

Химические уравнения обычно записываются в виде

$$Sub_1 + Sub_2 \Longleftrightarrow Sub_3 + Sub4, \tag{17}$$

где Sub_i — некоторые вещества. Помимо формул, для моделирования реакций нужно знать термодинамические свойства веществ [27] и Аррениусовские константы скоростей для каждой реакции [28].

На основе этих данных формируется СДУ, порядок которой равен числу веществ, участвующих в реакциях. Подробнее про это расписано в следующих разделах.

5.6 Реализация методов решения

Как уже говорилось выше, все методы семейства Рунге-Кутты можно представить в виде таблиц Бутчера. В связи с этим появилась идея реализовать алгоритм, принимающий в качестве аргументов задачу и таблицу Бутчера и возвращающий решение в виде таблицы с координатами. Благодаря этому алгоритму добавлять новые методы не вызывает никаких сложностей. Используемые методы перечислены на рисунке 17. Всего в данной работе используется 62 схемы со 2 по 6 порядок точности.

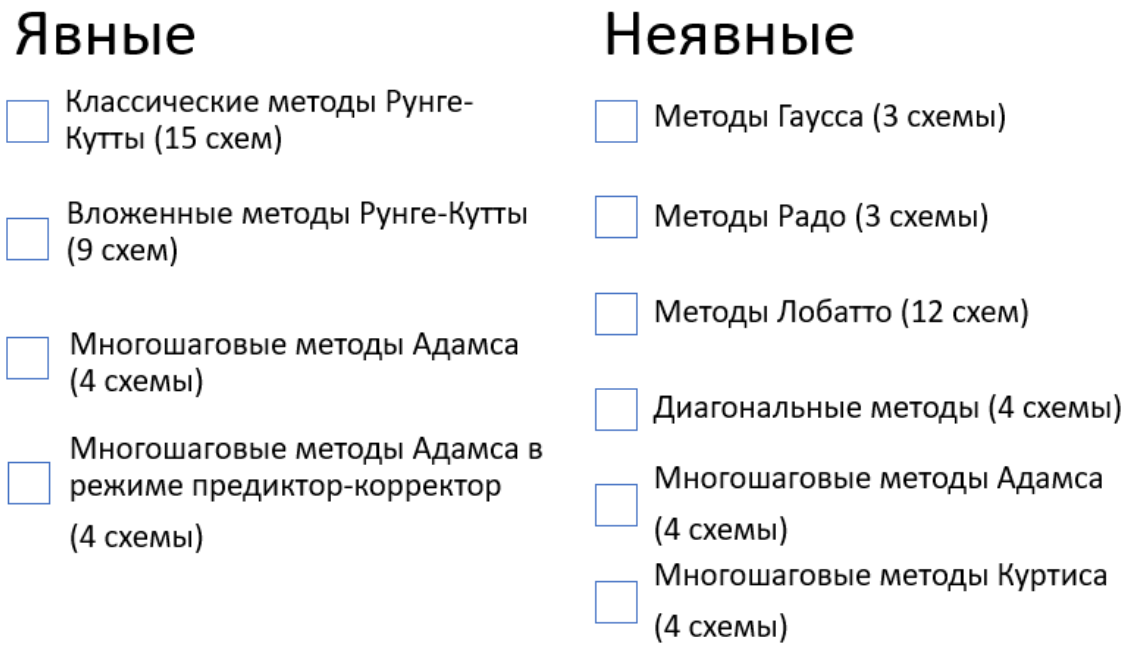


Рисунок 17 – Методы решения

По желанию пользователя можно добавить другой метод при помощи специального конструктора.

5.7 Генерация отчёта

При генерации отчёта учитывается количество уравнений, тип решаемой задачи, количество шагов интегрирования. При решении задач химической кинетики аналитическое решение не вводится. Отдельно выводятся графики изменения плотности и температуры. В обоих типах задач при большом количестве точек их число уменьшается в несколько раз (оставляется каждая вторая или каждая третья). На рисунке 18 показан пример одной страницы отчёта.

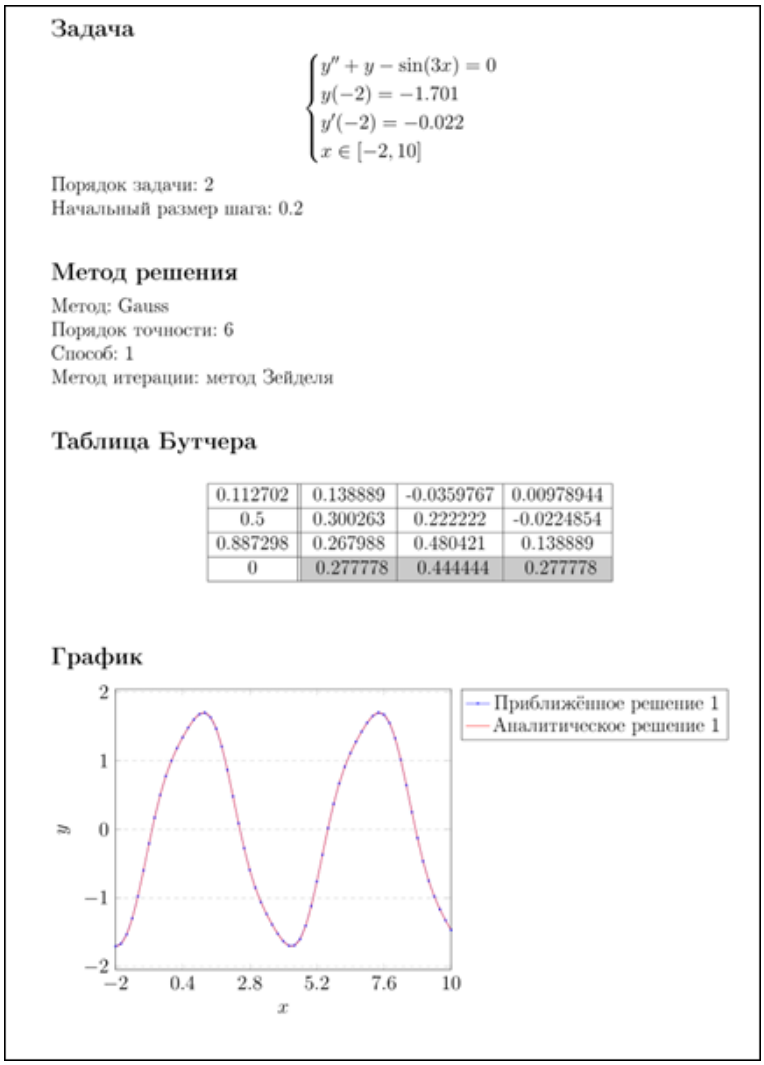


Рисунок 18 – Пример страницы отчёта

Для версии программы без интерфейса отчёт выводится либо в текстовом виде, либо в TeX-файле.

6 РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ

Задача №1

Рассмотрим пример решения нежёсткой системы ДУ 2-го порядка с использованием явного методы Рунге-Кутты 4-го порядка. Решение показано на рисунке 19.

$$\begin{cases} y'' + y - \sin(3x) = 0 \\ y(-2) = -1.701 \\ y'(-2) = -0.022 \\ x \in [-2, 10] \end{cases} \tag{18}$$

Аналитическое решение:

$$y(x) = cos(x) + \frac{11}{8} sin(x) - \frac{sin(3x)}{8} \tag{19}$$

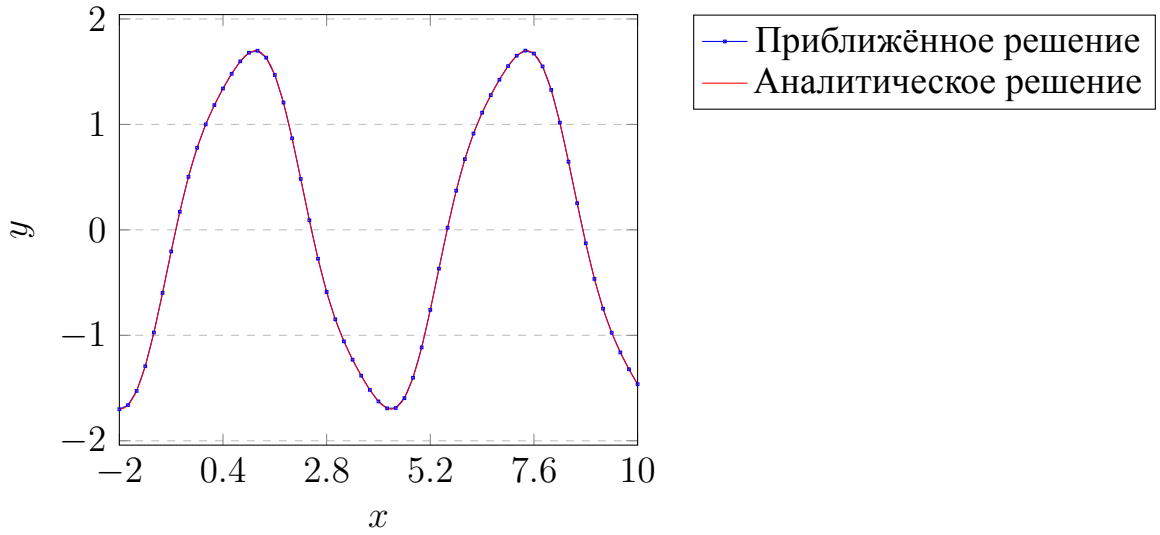


Рисунок 19 – Пример №

Задача №2

Рассмотрим СДУ порядка n, представленную в виде n дифференциальных уравнений 1-го порядка с начальными условиями. На рисунке 20 демонстрируется решение нежёсткой системы из двух уравнений с

использованием вложенного метода Фалберга 2-го порядка.

$$\begin{cases} y' = -12y + 10z^2 \\ z' = y - z - z^2 \\ y(0) = 1 \\ z(0) = 1 \\ x \in [0, 3] \end{cases} \tag{20}$$

Аналитическое решение:

$$\begin{gathered} y(x) = \exp(-2x) \\ z(x) = \exp(-x) \end{gathered} \tag{21}$$

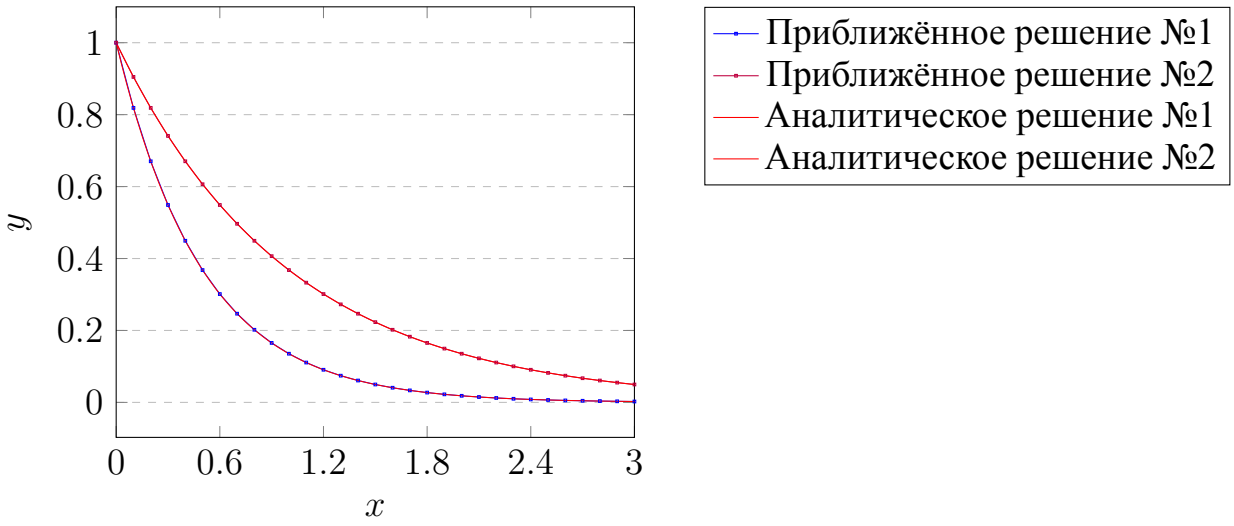


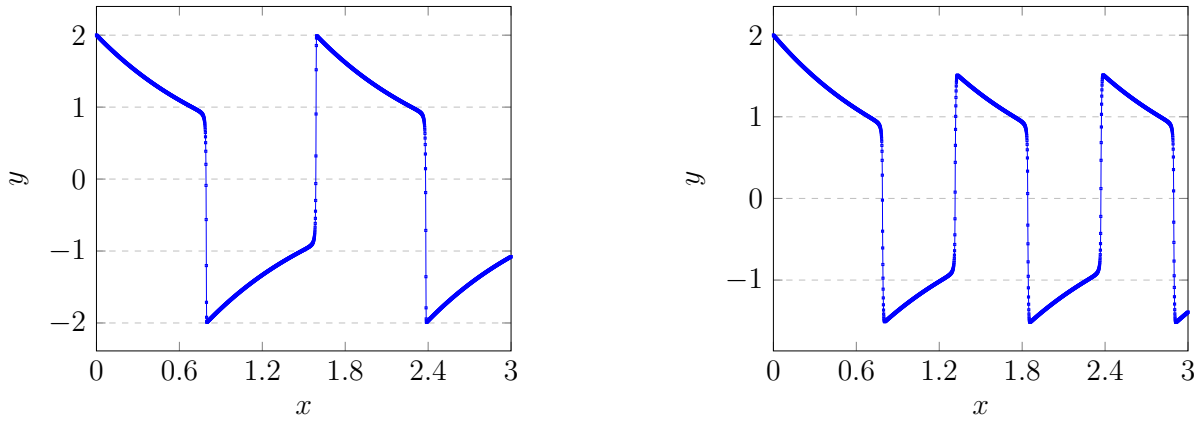
Рисунок 20 – Решение задачи №2

Задача №3

Отдельного внимания стоит пример решения жёсткой задачи уравнений Ван дер Поля, представленное при помощи явного метода Рунге-Кутты 6-го

порядка и неявного метода Гаусса 6-го порядка на рисунке 21.

$$\begin{cases} y'' - 500(1 - y^2)(y + y') = 0 \\ y(0) = 2 \\ y'(0) = 0 \\ x \in [0, 3] \end{cases} \tag{22}$$



а) б)

Рисунок 21 – Решение задачи №3:
а) неявным методом Гаусса 6-го порядка; б) явным методом Рунге-Кутты 6-го порядка

Коэффициент жёсткости данной задачи 500 и по графику видно, что жёсткость таких задач проявляется в резком скачке градиента функции. При применении явного метода расчёт идёт сначала верно, однако после первого скачка сбивается, показывает неправильный период волн и в целом даёт неверный результат. Неявный же метод смог решить задачу в соответствии с расчётами, взятыми из [29; 3]. Данный пример демонстрирует необходимость использования неявных методов, так как решение при помощи явных методов высокого порядка на первый взгляд кажется верным, но таковым может не являться.

7 РЕШЕНИЕ ЗАДАЧ ХИМИЧЕСКОЙ КИНЕТИКИ

7.1 Модели химической кинетики

Для описания химической реакции необходимо знать закономерности её протекания во времени, а именно, скорость и механизм. Скорость и механизм химических превращений изучает раздел химии — химическая кинетика [30; 31; 28].

Будем рассматривать многокомпонентную систему переменного состава из N веществ, в которой протекает N_R реакций вида:

$$\sum_{i=1}^{N} \overrightarrow{\nu_i^{(r)}} M_i \underset{\overrightarrow{W^{(r)}}}{\overleftarrow{W^{(r)}}} \sum_{i=1}^{N} \overleftarrow{\nu_i^{(r)}} M_i,$$

$$\overleftrightarrow{q^{(r)}} = \sum_{i=1}^{N} \overleftrightarrow{\nu_i^{(r)}}, \tag{23}$$

$$r = 1, ..., N_R,$$

$$i = 1, ..., N$$

Здесь r — порядковый номер реакции, i — порядковый номер вещества, $\overleftrightarrow{\nu_i}$ — стехиометрические коэффициенты (коэффициенты, стоящие перед молекулами веществ в химических уравнениях), $\overleftrightarrow{q^{(r)}}$ — молекулярность соответствующих элементарных реакций (число частиц, которые участвуют в элементарном акте химического взаимодействия).

В записи каждой реакции фигурирует $W^{(r)}$ — скорость химической реакции. Она прямо пропорциональна произведению объёмных концентраций участвующих в ней компонентов и так называемой константы скорости реакции $\overleftrightarrow{K^r}(T)$, зависящей от температуры (в общем случае и от давления).

Общий вид формул скоростей химических реакций:

$$\overleftrightarrow{W^r} = \overleftrightarrow{K^r}(T) \prod_i (\rho\gamma_i)^{\overleftrightarrow{\nu^r}} \tag{24}$$

Константы скорости реакции $\overleftrightarrow{K^r}(T)$ расчитываются для прямого и

оборатного хода реакции по следующим формулам:

$$\overrightarrow{K^r}(T) = AT^n \exp\left(-\frac{E}{T}\right) \tag{25}$$

$$\overleftarrow{K^r}(T) = \overrightarrow{K^r}(T) \exp\left(\sum_{i=1}^{N}(\overrightarrow{\nu_i^{(r)}} - \overleftarrow{\nu_i^{(r)}})(\frac{G_i^0(T)}{RT} + \ln\frac{RT}{p_0})\right) \tag{26}$$

Здесь A, n, E — Аррениусовские константы, $r = 1, ..., N_R$ — порядковый номер реакции, $G_i^0(T)$ — стандартный молярный потенциал Гиббса. Для вычисления $G_i^0(T)$ используются полиномиальные аппроксимационные формулы:

$$G_i^0(T) = \Delta_f H^0(T_0) - [H^0(T_0) - H^0(0)] - T\Phi(T_0), \tag{27}$$

где $H^0(0)$ — стандартная энтальпия при абсолютном нуле, $H^0(T_0)$ — стандартная энтальпия при T_0. Для задания $\Phi^0(T)$ применяют полиномы.

$$\Phi^0(T) = \phi_0 + \phi_{ln}\ln x + \phi_{-2}x^{-2} + \phi_{-1}x^{-1} + \phi_1 x + \phi_2 x^2 + \phi_3 x^3 \tag{28}$$

Здесь $\phi_0, \phi_{ln}, \phi_{-2}, ..., \phi_3$ — числовые коэффициенты, индивидуальные для каждого вещества.

Используя скорости $W^{(r)}$, можно составить уравнения изменения мольно-массовых концентраций по времени, имеющих следующий вид:

$$\begin{cases} \rho\dfrac{d\gamma_i}{dt} = W_i(\rho, T, \gamma_1, ..., \gamma_N) \\ \gamma_i(t_0) = \gamma_i^0, \\ i = 1, ..., N \end{cases} \tag{29}$$

где W_i — скорость образования i-го вещества. Вычисляется W_i по формуле (30):

$$W_i = \sum_{r=1}^{N_R}(\overrightarrow{\nu_i^{(r)}} - \overleftarrow{\nu_i^{(r)}})(\overrightarrow{W^{(r)}} - \overleftarrow{W^{(r)}}) \tag{30}$$

В формулах (29) помимо γ_i фигурируют плотность ρ и температура T.

Их значения могут быть как константами по времи вычисления, так и переменными. В работе реализовано моделирование случая, когда плотность меняется по закону (31), а температура константна, и случая, когда температура меняется, а плотность считается константой.

$$\rho = \frac{P}{RT \sum_{i=1}^{N} \gamma_i} \tag{31}$$

Второй случай, связанный с изменением температуры немного сложнее в реализации, так как температуру приходится находить итерационными методами из уравнения (32).

$$U = \sum_{i=1}^{N} (G_i^0(T) - T\frac{\partial G_i^0(T)}{\partial T} - RT)\gamma_i, \tag{32}$$

где U — полная внутренняя энергия системы, которая считается перед решением и сохраняется постоянной до конца расчётов. Если представить данное уравнение в виде (33), то можно использовать, к примеру, метод Ньютона для поиска корня уравнения.

$$F(T) = U - \sum_{i=1}^{N} (G_i^0(T) - T\frac{\partial G_i^0(T)}{\partial T} - RT)\gamma_i \tag{33}$$

Так как U, является константой, то решение $F(T) = 0$ будет соответствовать значению температуры на следующем шаге интегрирования.

7.2 Примеры расчёта химического состава смеси

В задачах химической кинетики на вход программе поступают не дифференциальные уравнения, а список химических уравнений, начальные температура, давление, описание примесей, при их наличии, и начальные концентрации веществ.

Задача №4

В данном примере на рисунке 22 показано моделирование пяти реакций с начальными концентрациями $\mu_{H_2} = 0.5$ и $\mu_{O_2} = 0.5$, нормальным атмосферным

давлением $P = 101325$Па и температурой $T = 2300$К.

$$\begin{cases} H_2 + O_2 \Longleftrightarrow 2OH \\ H + O_2 \Longleftrightarrow OH + O \\ H_2 + OH \Longleftrightarrow H_2O + H \\ H_2 + O \Longleftrightarrow OH + H \\ 2OH \Longleftrightarrow H_2O + O \end{cases} \tag{34}$$

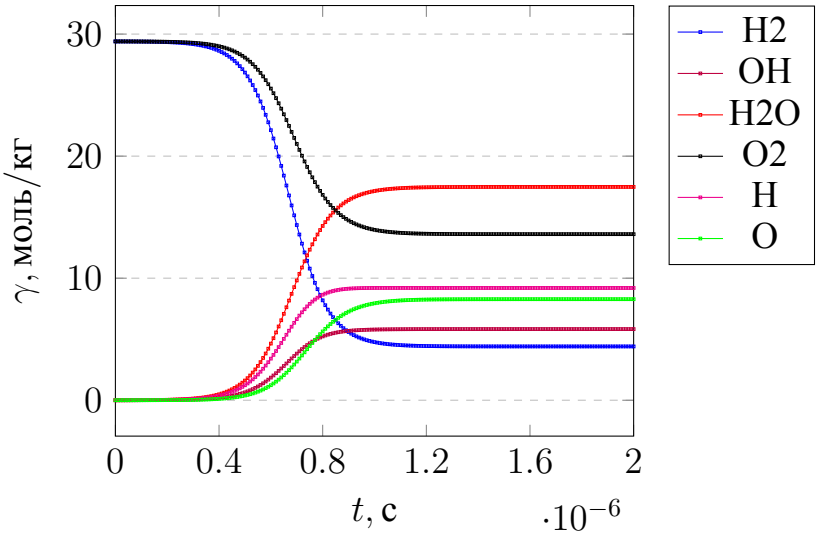


Рисунок 22 – Решение задачи №4

По графику видно, что реакция протекает примерно за 1 микросекунду после чего система приходит в равновесие.

По оси X указано время, по оси Y — мольно-массовые концентрации веществ. Так как концентрации H_2 и O_2 заданы по 0.5, то их мольно-массовые концентрации оказались равны примерно 29.3991 моль/кг.

Для решения данной задачи использовался неявный метод Гаусса 6го порядка с итерациями Зейделя, однако для неё можно использовать и другие неявные методы. В таблице 5 приведены сравнения некоторых неявных и явных методов для решения данной задачи по времени и количеству шагов.

Таблица 5 – Таблица сравнения методов

Метод	Время работы (мс)	Количество шагов	Точность
Неявный Гаусс 6-го порядка	254	35	0.001
Неявный Гаусс 4-го порядка	11	35	0.001
Неявный Гаусс 2-го порядка	235	50	0.001
Явный Рунге-Кутта 6-го порядка	92	100	0.001
Явный Рунге-Кутта 4-го порядка	55	100	0.001
Вложенный Дормана-Принс 4(5)-го порядка	191	200	0.001
Вложенный Фалберг 2(3)-го порядка	97	200	0.001

Задача №5

В следующем примере на рисунке 23 показано моделирование семи реакций с начальными концентрациями $\mu_{H_2} = 0.5$ и $\mu_{O_2} = 0.5$, нормальным атмосферным давлением $P = 101325$Па и температурой $T = 2300K$. Данный пример отличается от предыдущего наличием добавки M в последних двух реакциях.

$$\begin{cases} H_2 + O_2 \Longleftrightarrow 2OH \\ H + O_2 \Longleftrightarrow OH + O \\ H_2 + OH \Longleftrightarrow H_2O + H \\ H_2 + O \Longleftrightarrow OH + H \\ 2OH \Longleftrightarrow H_2O + O \\ H + OH + M \Longleftrightarrow H_2O + M \\ 2H + M \Longleftrightarrow H_2 + M \end{cases} \tag{35}$$

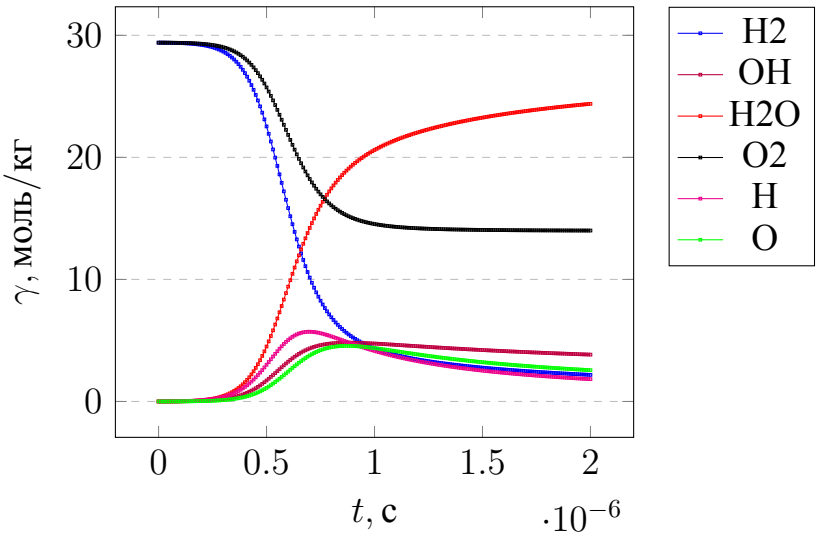


Рисунок 23 – Решение задачи №5

Количество ДУ в данном случае не меняется и зависит только от количества участвующих в реакциях веществ.

Таблица 6 – Таблица сравнения методов

Метод	Время работы (мс)	Количество шагов	Точность
Неявный Гаусс 6-го порядка	331	50	0.001
Неявный Гаусс 4-го порядка	145	35	0.001
Неявный Гаусс 2-го порядка	201	100	0.001
Явный Рунге-Кутта 6-го порядка	64	100	0.001
Явный Рунге-Кутта 4-го порядка	41	100	0.001
Вложенный Дормана-Принс 4(5)-го порядка	245	200	0.001
Вложенный Фалберг 2(3)-го порядка	102	200	0.001

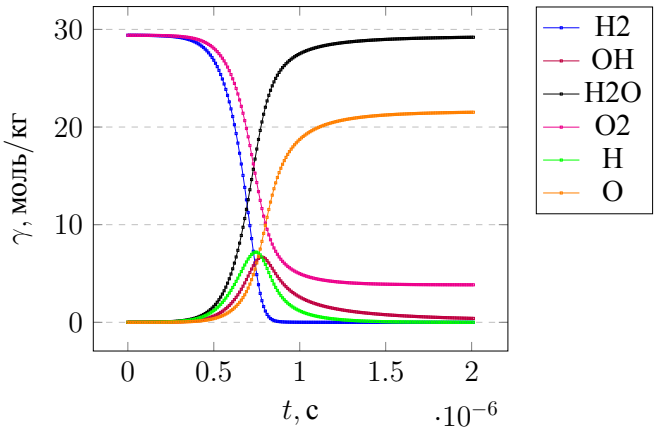
Стоит отметить, что добавка M существенно влияет на химическое взаимодействие и распределение концентраций химических веществ, поэтому приходится отдельно обрабатывать случай наличия добавки в реакции и

время работы алгоритма заметно падает. В таблице 6 приведено сравнение различных методов по времени и количеству шагов.

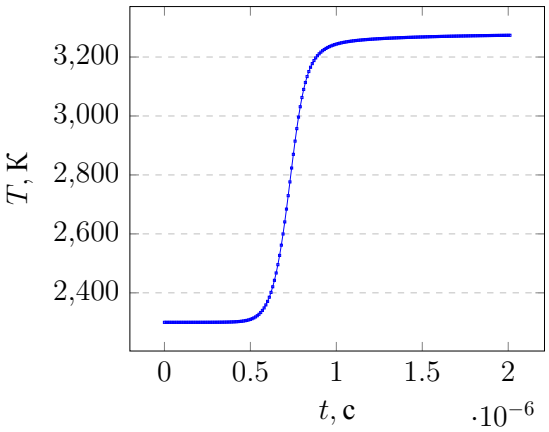
Оба этих примера показывают, что задачи химической кинетики в некоторых случаях можно решать и при помощи использования явных методов, но при этом может кратно увеличиться время работы и количество шагов.

7.3 Моделирование взрыва в камере постоянного объёма

Оба этих примера расчитывались при постоянной температуре и меняющейся плотности. Рассмотрим замкнутую систему неизменного объёма, заполненную смесью газов. Предположим, что в данной системе протекают 7 реакций вида (23). В задачах моделирования взрыва температура меняется вместе с концентрациями по закону (32). На рисунке 24 показано моделирование взрыва "бомбы"с начальными концентрациями $\mu_{H_2} = 0.5$ и $\mu_{O_2} = 0.5$, нормальным атмосферным давлением $P = 101325$Па и температурой $T = 2300$К.



а)



б)

Рисунок 24 – Моделирование взрыва: а) Мольно-массовые концентрации; б) Температура

Конечное значение температуры стало равно примерно $3285K$. Сами концентрации меняются быстрее, чем в предыдущем примере и система приходит в равновесие примерно за 2 микросекунды.

ЗАКЛЮЧЕНИЕ

Выпускная квалификационная работа бакалавра представляет собой программу для работы с СДУ при помощи множества методов семейства Рунге-Кутты. В работе реализовано 62 схемы со 2 по 6 порядок точности. Из них 18 явных со 2 по 6 порядок точности, 9 вложенных, включая схему Дормана-Принса 4(5) порядка, 22 неявных, в том числе схемы Радо, Гаусса и Лобатто для полных и неполных матриц. Помимо этого, протестирован один L-стабильный диагональный метод. Для неявных схем используются схемы решения САУ первого порядка (простой итерации, Зейделя) и второго порядка (метод Ньютона), причём для обращения матрицы применялся метод LU-разложения. Для дифференцирования функции при построении матрицы Якоби для метода Ньютоны использовались формулы с 4 порядком точности. При необходимости можно использовать формулы с меньшим порядком.

Для того чтобы определить, какие методы можно использовать, был реализован алгоритм вычисления числа жёсткости СДУ, использующий QR-разложение матрицы системы для поиска всех собственных чисел. При высокой жёсткости для расчёта применяютя только жёсткие схемы.

Для удобного отображения результатов вычислений был создан генератор pdf-отчётов, в которых представлена информация о решаемой задаче, метод её решения и таблица Бутчера для этого метода, график, отображающий решение численными методами и аналитическое (при наличии) и время, затраченное на работу программы. Если задача решалась при помощи жёстких схем, то дополнительно выводится информация о количестве итераций. Так же, при необходимости, может быть построен приближающий полином. Кроме этого, реализована возможность простого добавления новых методов решения на случай, если пользователю нужно решение каким-либо специфическим методом, которого нет в программе. При помощи пользовательского интерфейса можно использовать разработанную программу как для решения ОДУ, так и в целях обучения.

Программа тестировалась как на задачах химической кинетики, так и на модельных уравнениях и дала удовлетворительные результаты. В дальнейшем на базе данной программы планируется реализация решений краевых задач.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Леонов В. В. Самохина С. И. Исследование на жесткость системы обыкновенных дифференциальных уравнений, соответствующей математической модели деформационного упрочнения сплавов со сверхструктурой L12 // Инноватика - 2020 : сборник материалов XVI Международной школы-конференции студентов, аспирантов и молодых ученых, 23-25 апреля 2020 г., г. Томск, Россия. Томск. — 2020. — С. 384—387.

2. Wikipedia. MATLAB. — 2022. — URL: https://ru.wikipedia.org/wiki/MATLAB (дата обращения 21.05.2023).

3. Плохотников К. Э. Теория и практика в среде MATLAB. — М : Горячая линия, 2013. — С. 496.

4. Wikipedia. SMath Studio. — 2022. — URL: https://ru.wikipedia.org/wiki/SMath_Studio (дата обращения 21.05.2023).

5. Wikipedia. GNU Scientific Library. — 2023. — URL: https://ru.wikipedia.org/wiki/GNU_Scientific_Library (дата обращения 21.05.2023).

6. Castillo J. A. D. S. DotNumerics. — 2016. — URL: http://www.dotnumerics.com/ (дата обращения 21.05.2023).

7. Wikipedia. WolframAlpha. — 2023. — URL: https://ru.wikipedia.org/wiki/WolframAlpha (дата обращения 21.05.2023).

8. Олемской И. В. Метод типа Рунге-Кутты интегрирования систем и дифференциальных уравнений второго порядка специального вида. — Москва, 2004. — URL: https://cyberleninka.ru/article/n/metod-tipa-runge-kutty-integrirovaniya-sistem-i-differentsialnyh-uravneniy-vtorogo-poryadka-spetsialnogo-vida.

9. Пунтус А. А. Дифференциальные уравнения. — 2014. — С. 364. — ISBN 978-5-703-52323-0.

10. Hairer E., Norsett S., Wanner G. Solving Ordinary Differential Equations I: Nonstiff Problems. Т. 8. — 01.1993. — ISBN 978-3-540-56670-0. — DOI: 10.1007/978-3-540-78862-1.

11. Пошивайло И. П. Жёсткие и плохо обусловленные нелинейные модели и методы их расчета // Инноватика - 2020 : сборник материалов XVI Международной школы-конференции студентов, аспирантов и молодых ученых, 23-25 апреля 2020 г., г. Томск, Россия. Томск. — 2014.

12. Галанин М. П. Ходжаева С. Р. Методы решения жестких обыкновенных дифференциальных уравнений. Результаты тестовых расчетов : Препринты ИПМ им. М.В.Келдыша. — Москва, 2013. — eprint: https://keldysh.ru/papers/2013/prep2013_98.pdf. — URL: https://library.keldysh.ru/preprint.asp?id=2013-98.

13. Авдюшев В. А. Численные методы интегрирования обыкновенных дифференциальных уравнений. — Томск : ИДО ТГУ, 2009. — С. 52.

14. John C. Butcher. Numerical methods for ordinary differential equations: early days // The Birth of Numerical Analysis / под ред. A. Bultheel, R. Cools. — World Scientific, 2009. — С. 35—44. — DOI: 10.1142/9789812836267_0003. — URL: https://doi.org/10.1142/9789812836267%5C_0003.

15. Олемской И. В. Вложенный пятиэтапный метод пятого порядка типа Дормана-Принса // Ж. вычисл. матем. и матем. физ. — 2005. — Т. 45, № 7. — С. 1140—1150. — DOI: 10.7868/S0044466915030114.

16. Куликов Г. Ю. Вложенные симметричные неявные гнездовые методы Рунге-Кутты типов Гаусса и Лобатто для решения жестких обыкновенных дифференциальных уравнений и гамильтоновых систем // Ж. вычисл. матем. и матем. физ. — 2015. — Т. 55, № 6. — С. 986—1007. — DOI: 10.7868/S0044466915030114.

17. Hairer E., Wanner G. Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems. Т. 14. — 01.1996. — DOI: 10.1007/978-3-662-09947-6.

18. Фалейчик Б. В. Одношаговые методы численного решения задачи Коши. — 2010. — С. 42.

19. Холодов А. С. Лобанов А. И. Евдокимов А. В. Разностные схемы для решения жестких обыкновенных дифференциальных уравнений в пространстве неопределенных коэффициентов. — 1985. — С. 49.

20. Скворцов Л. М. Диагонально-неявные методы Рунге–Кутты для жестких задач // Ж. вычисл. матем. и матем. физ. — 2006. — Т. 46, № 12. — С. 2209—2222. — DOI: 10.1134/S0965542506120098.

21. Широбоков Н. В. Диагонально-неявные схемы Рунге-Кутты // Ж. вычисл. матем. и матем. физ. — 2002. — Т. 42, № 7. — С. 974—979.

22. Демидова О. Л. Малинина Н. Л. Нелинейные системы. Теория приближения функций в электронных таблицах. Практика вычислений в электронных таблицах. — Московский авиационный институт, 2021. — С. 47. — ISBN 978-5-4316-0774-5.

23. Wikipedia. Qt. — 2023. — URL: https://ru.wikipedia.org/wiki/Qt (дата обращения 21.05.2023).

24. Wikipedia. CMake. — 2023. — URL: https://ru.wikipedia.org/wiki/CMake (дата обращения 21.05.2023).

25. Wikipedia. GNU Debugger. — 2023. — URL: https://ru.wikipedia.org/wiki/GNU_Debugger (дата обращения 21.05.2023).

26. Wikipedia. Valgrind. — 2023. — URL: https://ru.wikipedia.org/wiki/Valgrind (дата обращения 21.05.2023).

27. Гурвич Л. В. Вейц И. В. Медведев В. А. и др. Термодинамические свойства индивидуальных веществ. Т. 1. — 1978. — С. 328.

28. Гидаспов В. Ю. Северина Н. С. Элементарные модели и вычислислительные алгоритмы физической газовой динамики. Термодинамика и химическая кинетика. — Издательство Вузовская книга, 2014. — С. 84.

29. Жук Д. М. Маничев В. Б. Сахаров М. К. Сравнение современных решателей жестких систем обыкновенных дифференциальных уравнений с решателями Си библиотеки sadel // Машиностроение и компьютерные технологии. — 2012. — № 8. — С. 284—300. — DOI: 10.7463/0812.0445558.

30. Булатов П. Е. Белов А. А. Калиткин Н. Н. Расчет химической кинетики явными схемами с геометрически-адаптивным выбором шага // Препринты ИПМ им. М. В. Келдыша. — 2018. — С. 32.

31. Гидаспов В. Ю. Северина Н. С. Некоторые задачи физической газовой динамики. — Издательство МАИ, 2016. — С. 79.

ПРИЛОЖЕНИЕ А

Исходный код

Исходный код программы выложен в публичном репозитории GitHub. QR код на репозиторий представлен на рисунке А.1.

Рисунок А.1 – Рисунок в приложении