

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
вычислительной техники

Кафедра
МОиПЭВМ

Направление подготовки 09.03.04 «Программная инженерия»

Направленность Программная инженерия

БАКАЛАВРСКАЯ РАБОТА

на тему

**Программный комплекс параметрической
идентификации модели типа «черный ящик»
динамической системы**

Студент _____ Тында Евгений Александрович
(подпись, дата) (ФИО полностью)

Руководитель _____ д.т.н., профессор Макарычев П.П.
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр _____ Такташкин Д.В.
(подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от _____ № _____)

Заведующий кафедрой _____ Козлов А.Ю.
(подпись) (фамилия, инициалы)

Работа защищена с отметкой _____ (протокол заседания ГЭК от _____ № _____)

Секретарь ГЭК _____ Попова Н.А.
(подпись) (фамилия, инициалы)

Пенза, 2026

Федеральное государственное бюджетное образовательное учреждение
высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет

Вычислительной техники

Кафедра

МО и ПЭВМ

«Утверждаю»
Заведующий кафедрой
А.Ю. Козлов
«26» декабря 2025 г

**ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА**

1. Студент Тында Евгений Александрович гр. 22ВП1 факультета ФВТ

<Фамилия, имя, отчество полностью>

2. Тема работы Программный комплекс параметрической идентификации модели типа «черный ящик» динамической системы

Тема утверждена приказом ПГУ №1583/о от "30" декабря 2025 г.

3. Руководитель работы д.т.н., профессор кафедры «МО и ПЭВМ» Макарычев П.П.

4. Задание на работу (назначение разработки, исходные данные и т.п.)

1) Назначение программный комплекс параметрической идентификации модели типа «черный ящик» для восстановления параметров динамической системы с одним входом и одним выходом на основе интегрально-степенных рядов Вольтерра.

2) Среда выполнения: операционная система Windows 11

Языковые средства разработки: C#

Инструментальные средства: Visual Studio 2022, WinEdit 8.1, Visual Paradigm

Технологии разработки: .NET 8, Windows forms

3) Основные функции приложения:

а) загрузка входных и выходных сигналов из файла табличного формата;

б) загрузка демонстрационного набора данных;

в) отображение входных данных в табличной форме;

г) построение графиков входного и выходного сигналов;

д) выбор метода идентификации параметров модели;

е) задание количества базисных функций для первого и второго ядер модели;

ж) автоматическое определение количества узлов коллокации;

з) выполнение расчета параметров модели методом коллокации;

- и) выполнение расчета параметров модели методом наименьших квадратов;
л) вычисление невязки, среднеквадратической, относительной и максимальной ошибки;
м) построение графиков;
н) визуализация ядер;
о) выход пользователя из программы

5. Перечень подлежащих разработке вопросов

- 1) Анализ предметной области и требования к системе;
2) Проектирование программных средств;
3) Реализация программных средств;
4) Тестирование программных средств;
5) Оформление пояснительной записки;

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области и требований к системе	10%	07.04.2026 – 13.04.2026	
2	Проектирование программных средств	20%	14.04.2026 – 25.04.2026	
3	Реализация программных средств	30%	27.04.2026 – 11.05.2026	
4	Тестирование программных средств	20%	12.05.2026 – 21.05.2026	
5	Оформление пояснительной записки	20%	22.05.2026 – 02.06.2026	

Дата выдачи задания «26» декабря 2025г.

Руководитель бакалаврской работы _____ 26.12.2025 г.
 (подпись, дата)

Задание к исполнению принял студент _____ 26.12.2025 г.
 (подпись, дата)

Работу к защите допустить
 Декан ФВТ _____ Л.Р. Фионова
 (подпись, дата)

Реферат

Пояснительная записка содержит 106 листов, 15 рисунков, 13 таблиц, 29 использованных источников и 4 приложения.

ДИНАМИЧЕСКАЯ СИСТЕМА, МОДЕЛЬ «ЧЕРНЫЙ ЯЩИК», ИДЕНТИФИКАЦИЯ ПАРАМЕТРОВ, ИНТЕГРАЛЬНО-СТЕПЕННЫЕ РЯДЫ ВОЛЬТЕРРА, ЯДРА ВОЛЬТЕРРА, МЕТОД КОЛЛОКАЦИИ, МЕТОД НАИМЕНЬШИХ КВАДРАТОВ, МНОГОЧЛЕНЫ ЧЕБЫШЕВА, C#, .NET, WINDOWS FORMS.

Объект исследования – динамическая система с одним входом и одним выходом, рассматриваемая в виде модели типа «черный ящик».

Цель выпускной квалификационной работы – разработать программный комплекс параметрической идентификации модели типа «черный ящик» на основе интегрально-степенных рядов Вольтерра.

Разработка программного комплекса проводилась на языке программирования C# с использованием платформы .NET 8 и технологии Windows Forms в среде программирования Visual Studio 2022. Для подготовки пояснительной записки использовались WinEdt 8.1 и MiKTeX.

Среда исполнения – Windows 11.

В ходе выполнения выпускной квалификационной работы был разработан программный комплекс, обеспечивающий загрузку входных данных, выбор метода идентификации, расчет коэффициентов ядер Вольтерра, восстановление выходного сигнала, вычисление показателей ошибки, построение графиков и сохранение результатов. Реализованы метод коллокации и метод наименьших квадратов. Все требования к разработке программного комплекса соблюдены.

					ПГУ 09.03.04 – 08ВР221.21 ПЗ			
Изм.	Лист	№ докум.	Подпись	Дата	Программный комплекс параметрической идентификации модели типа «черный ящик» динамической системы	Лит.	Лист	Листов
Разраб.	Тында Е. А.						4	106
Провер.	Макарычев П. П.					Группа 22ВП1		
Реценз.								
Н. Контр.	Такташкин Д. В.							
Утверд.	Козлов А. Ю.							

Содержание

Введение	8
1 Идентификация динамических систем на основе модели «черного ящика»	12
1.1 Роль идентификации в задачах анализа динамических систем	12
1.2 Динамические системы с одним входом и одним выходом	14
1.3 Модель «черного ящика»	15
1.4 Линейные и нелинейные системы с памятью	16
1.5 Интегрально-степенные ряды Вольтерра	17
1.6 Свойства ядер Вольтерра	19
1.7 Частотное представление рядов Вольтерра	20
1.8 Применение моделей Вольтерра в различных областях	21
1.9 Постановка задачи идентификации для модели второго порядка	22
1.10 Переход от теоретической модели к программному комплексу	23
1.11 Ограничения модели Вольтерра	23
1.12 Выводы по первой главе	24
2 Численные методы идентификации параметров модели Вольтерра	26
2.1 Общая постановка задачи идентификации	26
2.2 Аппроксимация ядер модели	28
2.3 Полиномы Чебышева как базисные функции	29
2.4 Вычисление интегральных коэффициентов модели	31
2.5 Формирование матричной модели	33
2.6 Метод коллокации	34
2.6.1 Идея метода коллокации	34
2.6.2 Получение системы уравнений	35
2.6.3 Выбор узлов коллокации	35
2.6.4 Алгоритм метода коллокации	36
2.6.5 Достоинства и ограничения метода коллокации	37

2.7 Метод наименьших квадратов	38
2.7.1 Идея метода наименьших квадратов	38
2.7.2 Матричная форма метода	39
2.7.3 Применение МНК к модели Вольтерра.....	39
2.7.4 Алгоритм метода наименьших квадратов	40
2.7.5 Преимущества и ограничения МНК	40
2.8 Невязка и критерии качества идентификации	41
2.9 Сравнение метода коллокации и метода наименьших квадратов	42
2.10 Выбор параметров модели	44
2.11 Алгоритмическая схема идентификации	44
2.12 Выводы по второй главе	45
3 Проектирование программного комплекса	47
3.1 Назначение и область применения программного комплекса	47
3.2 Анализ требований к программному комплексу	49
3.3 Функциональные требования	51
3.4 Нефункциональные требования	52
3.5 Формат входных и выходных данных	53
3.6 Выбор средств разработки	55
3.7 Проектирование архитектуры программного комплекса	56
3.8 Проектирование пользовательского интерфейса	58
3.9 Проектирование структуры классов и модулей	59
3.10 Проектирование алгоритма работы программного комплекса	61
3.11 Проектирование обработки ошибок.....	64
3.12 Выводы по третьей главе	64
4 Реализация программного комплекса	66
4.1 Структура программной реализации	66
4.2 Загрузка и предварительная подготовка данных.....	68
4.3 Реализация математического ядра.....	69
4.4 Программная реализация методов идентификации	70

4.4.1 Реализация метода коллокации	70
4.4.2 Реализация метода наименьших квадратов	71
4.5 Реализация пользовательского интерфейса	72
4.6 Порядок работы с программой	72
4.7 Подготовка программы к эксплуатации	73
4.8 Выводы по четвертой главе	74
5 Тестирование и анализ результатов работы программного комплекса.....	75
5.1 Методика и условия тестирования	75
5.2 Функциональное тестирование	76
5.3 Проведение вычислительного эксперимента.....	77
5.3.1 Результаты метода коллокации	77
5.3.2 Результаты метода наименьших квадратов.....	78
5.3.3 Сравнение методов	80
5.4 Анализ достоверности и ограничений результатов	81
5.5 Выводы по пятой главе	82
Заключение	84
Список использованных источников	86
Приложение А (рекомендуемое) Спецификации вариантов использования.	89
Приложение Б (рекомендуемое) Дополнительные результаты вычислительного эксперимента	93
Приложение В (рекомендуемое) Протокол функционального тестирования	97
Приложение Г (обязательное) Ключевые фрагменты программного кода...	99

Введение

Современные технические, информационные и физические объекты во многих случаях представляют собой динамические системы, поведение которых определяется не только текущим входным воздействием, но и предысторией его изменения. При этом внутренняя структура исследуемого объекта может быть известна неполностью, быть слишком сложной для непосредственного математического описания или вообще не использоваться в рамках прикладной задачи. В таких условиях актуальной становится задача построения приближенной модели по наблюдаемым входным и выходным данным, то есть задача идентификации параметров динамической системы. Общие принципы идентификации линейных и нелинейных динамических систем изложены в работах [1, 2].

Одним из распространенных подходов к описанию таких объектов является модель «черного ящика». В данной постановке система рассматривается как оператор, преобразующий входной сигнал $x(t)$ в выходной сигнал $y(t)$, а задача исследования состоит в построении такой математической модели, которая по известному входу воспроизводит выход с приемлемой точностью. Данный подход удобен для программной реализации, поскольку позволяет работать с различными прикладными областями без детального учета физической природы объекта.

Для описания нелинейных динамических систем с памятью в работе используется интегрально-степенное представление на основе рядов Вольтерра. Ряд Вольтерра можно рассматривать как обобщение степенного ряда Тейлора на случай операторов с памятью: если ряд Тейлора описывает статическую нелинейную зависимость между текущими значениями входа и выхода, то ряд Вольтерра учитывает влияние прошлых значений входного сигнала через многомерные интегральные свертки. В работах по моделированию

нелинейных систем отмечается, что ядра Вольтерра являются многомерным обобщением импульсной характеристики линейной системы [3]. В учебных материалах по применению рядов Вольтерра в радиочастотных цепях также подчеркивается, что такой аппарат позволяет учитывать частотные эффекты и память слабонелинейных систем, которые не описываются обычным рядом Тейлора [4].

В рамках выпускной квалификационной работы рассматривается система с одним входом и одним выходом. Основная математическая модель имеет вид усеченного ряда Вольтерра второго порядка:

$$y(t) = \int_0^t K_1(s)x(t-s)ds + \int_0^t \int_0^t K_2(s_1, s_2)x(t-s_1)x(t-s_2)ds_1ds_2, \quad t \in [0, T]. \quad (1)$$

где $x(t)$ – входной сигнал, $y(t)$ – выходной сигнал, $K_1(s)$ – ядро первого порядка, $K_2(s_1, s_2)$ – ядро второго порядка. Ядро K_1 описывает линейную часть реакции системы, а ядро K_2 позволяет учитывать квадратичную нелинейность и взаимодействие значений входного сигнала, взятых с различными задержками.

Актуальность темы определяется необходимостью разработки программного средства, позволяющего выполнять идентификацию параметров такой модели по заданным данным, проводить сравнение численных методов и наглядно представлять результаты вычислений. Практическая значимость работы связана с тем, что программный комплекс может использоваться для анализа входных и выходных сигналов в задачах обработки данных, моделирования динамических объектов, исследования систем связи и других систем, где требуется построение вход-выходной модели без детального описания внутреннего устройства объекта.

Целью выпускной квалификационной работы является разработка программного комплекса идентификации параметров динамической системы

в модели «черного ящика» с одним входом и одним выходом на основе интегрально-степенной модели Вольтерра.

Для достижения поставленной цели необходимо решить следующие задачи:

- рассмотреть основные понятия теории динамических систем и модели «черного ящика»;
- изучить интегрально-степенное представление Вольтерра для нелинейных систем с памятью;
- сформулировать задачу идентификации ядер Вольтерра первого и второго порядка;
- рассмотреть численные методы идентификации параметров модели, включая метод коллокации и метод наименьших квадратов;
- спроектировать структуру программного комплекса и определить основные функциональные требования;
- реализовать загрузку входных данных, расчет параметров модели, построение графиков и вывод численной оценки невязки;
- провести тестирование разработанного программного комплекса и проанализировать полученные результаты.

Объектом исследования является нелинейная динамическая система с одним входом и одним выходом, рассматриваемая в виде модели «черного ящика». Предметом исследования являются методы идентификации параметров интегрально-степенной модели Вольтерра и их программная реализация.

Методологическую основу работы составляют теория интегральных моделей Вольтерра, методы аппроксимации функций, метод коллокации, метод наименьших квадратов, методы численного решения систем алгебраических уравнений, а также методы проектирования и реализации прикладного программного обеспечения.

Выпускная квалификационная работа состоит из введения, пяти разделов, заключения, списка использованных источников и приложений. В первом разделе рассматриваются теоретические основы идентификации динамических

систем и модели Вольтерра. Во втором разделе описываются численные методы идентификации параметров модели. В третьем разделе выполняется проектирование программного комплекса. В четвертом разделе рассматривается программная реализация. В пятом разделе приводятся методика тестирования и результаты вычислительных экспериментов.

1 Идентификация динамических систем на основе модели «черного ящика»

1.1 Роль идентификации в задачах анализа динамических систем

Современные технические, информационные и физические объекты редко могут быть описаны только статической зависимостью между входом и выходом. В большинстве практически значимых случаев объект обладает динамикой: его текущее состояние и текущий выход определяются не только текущим значением входного воздействия, но и предысторией изменения этого воздействия. Поэтому при построении математической модели необходимо учитывать зависимость выходного сигнала от времени, а также влияние предыдущих значений входного сигнала. Именно такие объекты далее будем называть динамическими системами.

В общем виде динамическая система может рассматриваться как оператор, преобразующий входной сигнал $x(t)$ в выходной сигнал $y(t)$:

$$y(t) = F[x](t), \quad t \in [0, T], \quad (2)$$

где F – оператор системы, $x(t)$ – входное воздействие, $y(t)$ – наблюдаемая реакция, $[0, T]$ – интервал наблюдения. Если структура оператора F полностью известна, то задача сводится к прямому моделированию: по заданному входу требуется найти выход. Однако во многих прикладных ситуациях внутренняя структура объекта либо неизвестна, либо известна неполностью, либо слишком сложна для непосредственного использования в расчетах. В этом случае возникает задача идентификации.

Под идентификацией динамической системы понимают построение математической модели системы по наблюдаемым данным [1, 2]. Обычно такими данными являются экспериментально или численно полученные пары сигналов $x(t)$, $y(t)$. Цель идентификации состоит в выборе такой модели и таких параметров модели, при которых рассчитанный выход $\hat{y}(t)$ достаточно близок к известному выходному сигналу $y(t)$. В простейшей постановке это можно

записать как задачу минимизации невязки:

$$\varepsilon = \|y(t) - \hat{y}(t)\| \rightarrow \min, \quad (3)$$

где норма выбирается в зависимости от постановки задачи. В вычислительных приложениях часто используются максимальная норма, среднеквадратичная ошибка, интегральная среднеквадратичная ошибка или дискретная сумма квадратов отклонений.

Актуальность задачи идентификации определяется тем, что во многих областях невозможно или нецелесообразно строить модель объекта только на основании физических законов. Например, для сложной электронной схемы можно записать систему дифференциальных уравнений, однако такая модель может быть слишком громоздкой для практического анализа. Для биомедицинского объекта физическая модель часто содержит множество индивидуальных параметров, которые трудно измерить непосредственно. Для механической конструкции, находящейся под случайным ветровым или волновым воздействием, полная модель может быть вычислительно дорогой. Во всех этих случаях удобным оказывается подход, при котором объект рассматривается как преобразователь входного сигнала в выходной, а его параметры определяются по наблюдениям.

Задачи идентификации тесно связаны с задачами прогнозирования, управления, диагностики и цифровой обработки сигналов. Если модель достаточно точно описывает связь между входом и выходом, то ее можно использовать для анализа поведения системы при новых входных воздействиях, для сравнения нескольких систем, для оценки влияния параметров, для обнаружения аномальных режимов работы и для построения программных средств моделирования.

В рамках данной выпускной квалификационной работы рассматривается идентификация параметров нелинейной динамической системы. Особенность задачи состоит в том, что система рассматривается в виде модели

«черного ящика», а в качестве математической основы используется интегрально-степенное представление на основе рядов Вольтерра. Такой подход позволяет описывать не только линейную динамическую реакцию, но и нелинейные эффекты, связанные с взаимодействием значений входного сигнала, взятых с различными задержками.

1.2 Динамические системы с одним входом и одним выходом

В зависимости от числа входов и выходов динамические системы разделяют на одноканальные и многоканальные. В данной работе рассматривается система с одним входом и одним выходом. В англоязычной литературе для таких систем часто используется термин SISO-система (single input – single output). В такой постановке имеется один входной сигнал $x(t)$ и один выходной сигнал $y(t)$, а задача моделирования состоит в описании зависимости между ними.

Схематически такая система может быть представлена в виде

$$x(t) \longrightarrow F \longrightarrow y(t), \quad (4)$$

где блок F обозначает неизвестное или частично известное преобразование. В отличие от статической зависимости, в которой значение выхода определяется только текущим значением входа, динамическая система может учитывать значения $x(\tau)$ при $\tau < t$. Поэтому выход в момент времени t в общем случае является функционалом от участка входного сигнала на интервале $[0, t]$:

$$y(t) = F[x(\tau), 0 \leq \tau \leq t]. \quad (5)$$

Одноканальная постановка удобна для первичной разработки программного комплекса. Она позволяет сосредоточиться на алгоритмах идентификации, аппроксимации ядер, численном вычислении интегралов и визуализации результатов. При этом сама идея может быть в дальнейшем расширена на многовходовые или многовыходовые системы, если возникнет необходимость учитывать несколько входных воздействий или несколько измеряемых выходных сигналов.

В разрабатываемой программе входной и выходной сигналы предполагается задавать либо в виде таблицы значений, загружаемой из файла, либо в виде заранее подготовленных тестовых функций. Для каждого момента времени t_k известны значения $x(t_k)$ и $y(t_k)$. На основе этих данных требуется определить параметры модели, позволяющей вычислять восстановленный выход $\hat{y}(t_k)$ и сравнивать его с исходным выходом.

1.3 Модель «черного ящика»

Модель «черного ящика» используется в тех случаях, когда исследователя интересует связь между входом и выходом объекта, но внутренняя структура объекта не рассматривается явно. Иными словами, вместо детального описания внутренних физических, химических, биологических или технических процессов строится внешняя модель преобразования входного сигнала в выходной.

Такой подход особенно полезен при работе со сложными системами. Внутренние уравнения объекта могут быть неизвестны, содержать большое число параметров или требовать сложных вычислительных процедур. При этом в распоряжении исследователя могут быть экспериментальные данные: входной сигнал и соответствующий ему выходной сигнал. В такой ситуации задача состоит не в построении полной физической модели, а в нахождении приближенного оператора, который воспроизводит наблюдаемое поведение.

Для модели «черного ящика» важны следующие элементы:

- входной сигнал $x(t)$, подаваемый на систему;
- выходной сигнал $y(t)$, наблюдаемый после прохождения через систему;
- структура математической модели, выбранная для описания преобразования;
- параметры модели, которые требуется определить;
- критерий качества, показывающий, насколько хорошо модель воспроизводит выход.

В данной работе в качестве структуры модели выбран усеченный ряд

Вольтерра второго порядка. Это означает, что система описывается суммой линейной интегральной части и квадратичной интегральной части. Линейная часть характеризует обычную динамическую реакцию системы, а квадратичная часть позволяет учитывать нелинейное взаимодействие различных задержанных значений входного сигнала.

Преимущество модели «черного ящика» состоит в ее универсальности. Одна и та же математическая схема может использоваться для сигналов различной природы: электрических, механических, биомедицинских, экономических и других. Для программного комплекса это особенно важно, поскольку приложение не должно быть жестко привязано к одной физической интерпретации данных. Пользователь предоставляет входной и выходной сигналы, выбирает метод идентификации и получает параметры модели, графики и оценку ошибки.

1.4 Линейные и нелинейные системы с памятью

Одним из основных признаков динамической системы является наличие памяти. Система с памятью учитывает не только текущее значение входа $x(t)$, но и предшествующие значения $x(t - s)$, где $s \geq 0$ – задержка. Для линейной стационарной системы с памятью связь между входом и выходом обычно описывается интегралом свертки:

$$y(t) = \int_0^t h(s)x(t-s)ds, \quad (6)$$

где $h(s)$ – импульсная характеристика системы. Эта функция показывает, как система реагирует на элементарное входное воздействие с учетом задержки. Формула (6) является базовой для анализа линейных динамических систем.

Если же система нелинейна, то простого интеграла свертки недостаточно. В нелинейной системе выход может зависеть не только от отдельных значений входа, но и от их произведений, степеней и взаимных взаимодействий. Например, при квадратичной нелинейности в выходном сигнале могут появляться слагаемые

вида $x(t - s_1)x(t - s_2)$, где s_1 и s_2 – различные задержки. Поэтому для описания нелинейных систем с памятью требуются более общие модели.

Для статической нелинейной системы, не имеющей памяти, можно использовать степенной ряд Тейлора:

$$y(t) = a_1x(t) + a_2x^2(t) + a_3x^3(t) + \dots \quad (7)$$

Однако в этой формуле выход зависит только от текущего значения входа. Если же необходимо учитывать предысторию сигнала, то каждый степенной член должен быть заменен интегральным оператором. Именно такая идея лежит в основе интегрально-степенных рядов Вольтерра.

В источниках по рядам Вольтерра подчеркивается, что ряд Вольтерра является аналогом ряда Тейлора для операторов с памятью: ряд Тейлора описывает статическую нелинейность, а ряд Вольтерра – нелинейное динамическое преобразование входа в выход [3, 4]. Это делает аппарат Вольтерра удобным для описания слабонелинейных систем, в которых нелинейные эффекты существенны, но не приводят к полной потере сходимости модели.

1.5 Интегрально-степенные ряды Вольтерра

Теория функциональных и интегро-дифференциальных уравнений Вольтерры составляет историческую основу используемого представления [5]. Возможность аппроксимации нелинейных операторов с памятью рядами Вольтерра и свойства таких представлений исследованы в работах [6, 7].

Ряд Вольтерра представляет выход нелинейной динамической системы в виде суммы однородных операторов различных порядков. В общем виде можно записать

$$y(t) = H_0 + H_1[x](t) + H_2[x](t) + \dots + H_n[x](t) + \dots, \quad (8)$$

где H_0 – постоянная составляющая, H_1 – оператор первого порядка, H_2 – оператор второго порядка и т.д. Каждый оператор H_j описывается многомерным

интегралом с соответствующим ядром Вольтерра:

$$H_j[x](t) = \int_0^t \dots \int_0^t K_j(s_1, \dots, s_j) \prod_{r=1}^j x(t - s_r) ds_1 \dots ds_j. \quad (9)$$

Функция $K_j(s_1, \dots, s_j)$ называется ядром Вольтерра j -го порядка. Для $j = 1$ ядро зависит от одной переменной и совпадает по смыслу с импульсной характеристикой линейной системы. Для $j = 2$ ядро зависит от двух переменных и описывает квадратичное взаимодействие задержанных значений входного сигнала. Чем выше порядок ядра, тем более сложные нелинейные эффекты может описывать модель, но тем больше становится число неизвестных параметров и вычислительная сложность.

В практических задачах бесконечный ряд Вольтерра обычно усекают. В данной работе используется модель второго порядка:

$$y(t) = \int_0^t K_1(s)x(t-s)ds + \int_0^t \int_0^t K_2(s_1, s_2)x(t-s_1)x(t-s_2)ds_1ds_2, \quad t \in [0, T]. \quad (10)$$

Данная формула соответствует схеме, в которой выход формируется суммой линейной реакции и квадратичной нелинейной реакции.

Выбор второго порядка является компромиссом между выразительностью модели и сложностью ее идентификации. С одной стороны, модель второго порядка уже позволяет учитывать нелинейные эффекты, отсутствующие в линейной свертке. С другой стороны, она остается достаточно простой для программной реализации, поскольку требуется определить только два ядра: K_1 и K_2 . Модели более высокого порядка потребовали бы построения и хранения многомерных ядер, что резко увеличило бы объем вычислений.

1.6 Свойства ядер Вольтерра

Ядра Вольтерра являются основными параметрами интегрально-степенной модели. Именно они определяют, каким образом задержанные значения входного сигнала влияют на выход. Ядро первого порядка $K_1(s)$ характеризует линейную динамическую составляющую. Если в модели оставить только первое слагаемое, то получится обычная линейная система с памятью:

$$y_1(t) = \int_0^t K_1(s)x(t-s)ds. \quad (11)$$

Ядро второго порядка $K_2(s_1, s_2)$ описывает квадратичные взаимодействия. Вклад второго порядка имеет вид

$$y_2(t) = \int_0^t \int_0^t K_2(s_1, s_2)x(t-s_1)x(t-s_2)ds_1ds_2. \quad (12)$$

Здесь переменные s_1 и s_2 задают две задержки, а произведение $x(t-s_1)x(t-s_2)$ показывает взаимодействие двух значений входного сигнала. Если ядро K_2 отлично от нуля, то система является нелинейной.

С математической точки зрения ядро второго порядка можно рассматривать как функцию двух переменных на квадрате $[0, T] \times [0, T]$. В ряде задач его можно считать симметричным:

$$K_2(s_1, s_2) = K_2(s_2, s_1). \quad (13)$$

Симметризация ядер используется в теории Вольтерра, поскольку порядок переменных интегрирования в произведении входных сигналов не всегда имеет самостоятельный физический смысл [8]. В данной работе условие симметрии на искомый двумерный массив коэффициентов не накладывается. Программная реализация определяет полный массив C_{ij} , по которому строится приближенное ядро второго порядка.

Для целей идентификации ядра K_1 и K_2 заранее неизвестны. Они заменяются конечномерными аппроксимациями, например разложениями по

базисным функциям. После этого задача нахождения функций сводится к задаче нахождения конечного набора коэффициентов. Более подробно такая параметризация рассматривается во второй главе.

1.7 Частотное представление рядов Вольтерра

Ряды Вольтерра могут рассматриваться как во временной, так и в частотной области. В частотной области ядрам Вольтерра соответствуют частотные функции Вольтерра. Для ядра j -го порядка такая функция определяется как многомерное преобразование Фурье:

$$H_j(\omega_1, \dots, \omega_j) = \int_0^\infty \dots \int_0^\infty K_j(s_1, \dots, s_j) e^{-i(\omega_1 s_1 + \dots + \omega_j s_j)} ds_1 \dots ds_j. \quad (14)$$

Для линейной системы частотная характеристика показывает, как система изменяет амплитуду и фазу гармонического входного сигнала. Для нелинейной системы частотные функции Вольтерра являются обобщением этой идеи. Они позволяют анализировать появление гармоник, интермодуляционных составляющих и других частотных эффектов, возникающих при нелинейном преобразовании сигнала. В работах по моделированию нелинейных систем отмечается, что VFRF можно рассматривать как частотные аналоги ядер Вольтерра и как обобщение частотной характеристики линейной системы [3].

Для данной выпускной квалификационной работы основным является временное представление, поскольку разрабатываемая программа должна работать с табличными значениями входного и выходного сигналов. Тем не менее частотная интерпретация важна для понимания областей применения рядов Вольтерра, особенно в задачах анализа радиочастотных цепей, усилителей, смесителей и каналов связи. В этих задачах нелинейность приводит к искажениям в спектре, а ряды Вольтерра позволяют связать эти искажения с параметрами модели [4].

1.8 Применение моделей Вольтерра в различных областях

Аппарат рядов Вольтерра используется в различных областях науки и техники. В инженерной механике такие модели применяются для описания нелинейной реакции конструкций на ветровые и волновые воздействия, для построения редуцированных моделей и для анализа случайных процессов. В статье [3] отмечается, что ряды Вольтерра могут использоваться как для построения модели по наблюдаемым данным, так и для анализа систем, уже заданных аналитическими уравнениями.

В радиотехнике и системах связи ряды Вольтерра применяются для анализа слабонелинейных систем с памятью. К таким системам относятся усилители, смесители, зеркала тока и другие электронные узлы, в которых паразитные емкости и индуктивности приводят к частотно-зависимым эффектам. В учебных материалах по применению рядов Вольтерра в радиочастотных цепях подчеркивается, что обычный ряд Тейлора не учитывает память, тогда как ряд Вольтерра позволяет анализировать высокочастотные искажения [4].

В обработке сигналов модель Вольтерра может использоваться для построения нелинейных фильтров. Линейный фильтр учитывает только взвешенную сумму прошлых значений входа, а фильтр Вольтерра дополнительно включает произведения значений сигнала с различными задержками. Это делает его применимым для задач, где линейная модель дает недостаточную точность.

В биомедицинских приложениях модели типа «черного ящика» могут использоваться для описания связи между измеряемыми физиологическими сигналами. При этом физический смысл входного и выходного сигналов может быть различным, но математическая постановка остается той же: по известному входу требуется воспроизвести выход. В данной работе медицинская интерпретация не является основной; упор делается на универсальную математическую и программную постановку задачи.

Таким образом, выбранная тема имеет междисциплинарный характер. С

одной стороны, она связана с теорией интегральных операторов и численными методами. С другой стороны, результат работы представляет собой программный комплекс, который может использоваться для обработки и анализа сигналов различной природы.

1.9 Постановка задачи идентификации для модели второго порядка

Пусть на интервале $[0, T]$ известны входной сигнал $x(t)$ и выходной сигнал $y(t)$. Требуется найти такие ядра K_1 и K_2 , чтобы модель второго порядка (10) воспроизводила выходной сигнал с заданной точностью. Введем восстановленный выход

$$\hat{y}(t) = \int_0^t K_1(s)x(t-s)ds + \int_0^t \int_0^t K_2(s_1, s_2)x(t-s_1)x(t-s_2)ds_1ds_2. \quad (15)$$

Тогда невязка модели определяется выражением

$$r(t) = y(t) - \hat{y}(t). \quad (16)$$

Задача идентификации состоит в выборе функций K_1 и K_2 , при которых невязка мала в выбранном смысле. Однако непосредственное восстановление функций является бесконечномерной задачей. Для перехода к численной реализации ядра представляют через конечное число параметров:

$$K_1(s) \approx \sum_{i=0}^{m_1} A_i \varphi_i(s), \quad K_2(s_1, s_2) \approx \sum_{i=0}^{m_2} \sum_{j=0}^{m_2} C_{ij} \varphi_i(s_1) \varphi_j(s_2), \quad (17)$$

где φ_i – базисные функции, A_i и C_{ij} – неизвестные коэффициенты. После такой замены требуется определить уже не сами функции, а конечный набор числовых параметров.

Таким образом, исходная задача идентификации сводится к построению системы алгебраических уравнений или задачи минимизации. Для этого могут применяться различные численные методы. В данной работе рассматриваются два подхода: метод коллокации и метод наименьших квадратов. Их подробное описание приведено во второй главе.

1.10 Переход от теоретической модели к программному комплексу

Теоретическая модель Вольтерра удобна для математического описания системы, но для практического использования требуется ее программная реализация. Разрабатываемый программный комплекс должен обеспечить полный вычислительный цикл: загрузку входных данных, выбор метода идентификации, задание параметров модели, расчет коэффициентов, построение восстановленного выходного сигнала и оценку ошибки.

На уровне данных программа работает с дискретными значениями сигналов:

$$\{t_k, x_k, y_k\}_{k=1}^N, \quad x_k = x(t_k), \quad y_k = y(t_k). \quad (18)$$

Поэтому непрерывные интегралы модели при реализации заменяются численными квадратурами или дискретными суммами. Выбор конкретной схемы вычислений зависит от используемого метода идентификации и от структуры входных данных.

Для пользователя программный комплекс должен представлять модель в понятном виде. Входными данными являются сигналы, параметрами расчета являются порядок аппроксимации, длина памяти и выбранный метод, а результатом являются коэффициенты модели, график исходного и восстановленного выходного сигнала, график невязки и численное значение ошибки. Такая структура соответствует общей схеме модели «черного ящика»: исследователь не задает внутренние уравнения системы, а получает приближенный оператор по входу и выходу.

1.11 Ограничения модели Вольтерра

Несмотря на универсальность, модель Вольтерра имеет ряд ограничений. Во-первых, она наиболее эффективна для слабонелинейных систем. Если нелинейность слишком сильна, то для адекватного описания могут потребоваться члены высоких порядков, а ряд может плохо сходиться или становиться

практически непригодным. В источниках по применению рядов Вольтерра отмечается, что для сильной нелинейности сумма членов может расходиться, поэтому метод обычно применяют к системам, для которых первые члены ряда дают основной вклад [4].

Во-вторых, при увеличении порядка модели резко возрастает вычислительная сложность. Ядро первого порядка зависит от одной переменной, ядро второго порядка – от двух переменных, ядро третьего порядка – от трех и т.д. Поэтому хранение, аппроксимация и идентификация ядер высоких порядков требуют значительно большего числа параметров.

В-третьих, результат идентификации зависит от качества входных данных. Если данные содержат значительный шум, пропуски или недостаточно возбуждают систему, то найденные коэффициенты могут быть неустойчивыми. Поэтому в практической реализации важны проверка входных данных, выбор адекватного порядка аппроксимации и анализ невязки.

В рамках данной работы эти ограничения учитываются за счет выбора модели второго порядка, использования конечномерной аппроксимации ядер и сравнения двух методов идентификации. Такой подход позволяет получить достаточно содержательную модель и одновременно сохранить реализуемость программного комплекса.

1.12 Выводы по первой главе

В первой главе рассмотрены основные понятия, необходимые для постановки задачи выпускной квалификационной работы. Показано, что динамическая система может рассматриваться как оператор, преобразующий входной сигнал в выходной, а при неизвестной внутренней структуре объекта удобно использовать модель «черного ящика». Для системы с одним входом и одним выходом такая модель позволяет сосредоточиться на связи между наблюдаемыми сигналами без детального описания внутренних физических процессов.

Рассмотрено отличие линейных и нелинейных систем с памятью. Показано, что линейная система описывается интегралом свертки с импульсной характеристикой, а для нелинейной системы требуется более общий аппарат. В качестве такого аппарата выбран интегрально-степенной ряд Вольтерра, являющийся обобщением ряда Тейлора на случай операторов с памятью.

В качестве основной математической модели выбрана усеченная модель Вольтерра второго порядка. Она включает ядро первого порядка K_1 , описывающее линейную составляющую, и ядро второго порядка K_2 , описывающее квадратичную нелинейность. Показано, что задача идентификации сводится к нахождению этих ядер по известным входному и выходному сигналам. Для дальнейшей численной реализации ядра необходимо аппроксимировать конечным числом параметров. Подробное описание соответствующих численных методов приведено во второй главе.

2 Численные методы идентификации параметров модели Вольтерра

2.1 Общая постановка задачи идентификации

В первой главе была введена динамическая система типа «черный ящик», для которой известны только входной сигнал и выходной сигнал. Внутренняя структура объекта при этом не используется явно: система рассматривается как оператор, преобразующий входное воздействие $x(t)$ в выходную реакцию $y(t)$. Для описания такого преобразования в работе применяется усеченная интегрально-степенная модель Вольтерра второго порядка. Выбор именно второго порядка связан с тем, что такая модель уже позволяет учитывать нелинейную составляющую преобразования, но при этом остается достаточно удобной для численной реализации и программной обработки.

Пусть на интервале наблюдения $[0, T]$ задана пара функций $x(t)$ и $y(t)$. Входной сигнал $x(t)$ считается известным воздействием на систему, а $y(t)$ – известным выходом, зарегистрированным в результате наблюдения или вычислительного эксперимента. Требуется подобрать такие ядра Вольтерра $K_1(s)$ и $K_2(s_1, s_2)$, чтобы восстановленный выходной сигнал $\hat{y}(t)$ был близок к исходному выходному сигналу $y(t)$. В рассматриваемой постановке модель имеет вид

$$\begin{aligned} \hat{y}(t) = & \int_0^t K_1(s)x(t-s)ds + \\ & + \int_0^t \int_0^t K_2(s_1, s_2)x(t-s_1)x(t-s_2)ds_1ds_2, \quad t \in [0, T]. \end{aligned} \quad (19)$$

Первое слагаемое в (19) отвечает за линейную часть реакции системы, а второе – за квадратичную нелинейную часть. В линейном случае достаточно было бы найти только ядро K_1 , которое является аналогом импульсной характеристики. В нелинейном случае появляется ядро второго порядка K_2 , описывающее влияние произведений значений входного сигнала в разные моменты прошлого

времени. Подобное представление соответствует общей идее ряда Вольтерра как разложения нелинейного оператора с памятью по однородным интегральным операторам [3, 4].

С математической точки зрения задача идентификации состоит в восстановлении неизвестных функций K_1 и K_2 . Однако такая задача является бесконечномерной: ядра представляют собой функции, а не конечное число параметров. Поэтому для программной реализации необходимо перейти от функциональной постановки к конечномерной. Для этого ядра приближаются конечными разложениями по выбранной системе базисных функций. После такой аппроксимации задача сводится к нахождению конечного набора коэффициентов.

На практике входной и выходной сигналы в программном комплексе будут задаваться не непрерывными функциями, а таблицами значений. Пусть известны точки

$$t_0, t_1, \dots, t_N, \quad x_k = x(t_k), \quad y_k = y(t_k), \quad k = 0, 1, \dots, N. \quad (20)$$

Если в исходном файле присутствует отдельный столбец времени, то значения t_k считываются из него и приводятся к интервалу $[0; 1]$. Если столбец времени отсутствует, временная сетка строится автоматически по номеру отсчета. Такая схема позволяет использовать один алгоритм как для модельных данных, так и для экспериментальных наборов сигналов.

В рамках данной работы рассматриваются два численных подхода к нахождению коэффициентов модели: метод коллокации и метод наименьших квадратов. Оба метода используют одну и ту же исходную модель (19), но отличаются способом построения системы для неизвестных коэффициентов. Метод коллокации требует выполнения модельного равенства в конечном наборе узлов, а метод наименьших квадратов минимизирует суммарную ошибку на множестве точек наблюдения. Поэтому эти методы удобно рассматривать совместно: сначала вводится общее конечномерное представление ядер, а затем показывается, как на его основе строятся две различные вычислительные схемы.

2.2 Аппроксимация ядер модели

Для перехода к конечномерной задаче введем систему базисных функций

$$B_0(s), B_1(s), B_2(s), \dots, \quad (21)$$

заданную на интервале $[0, T]$. В качестве такой системы могут выступать степенные функции, сплайны, тригонометрические функции, ортогональные многочлены и другие базисы. В данной работе основное внимание уделяется полиномам Чебышева, однако сначала удобно записать аппроксимацию в общем виде.

Ядро первого порядка будем искать в виде конечной суммы

$$\tilde{K}_{1,m}(s) = \sum_{i=0}^{m-1} A_i B_i(s), \quad (22)$$

где A_i – неизвестные коэффициенты, а m – число базисных функций, используемых для аппроксимации K_1 . Ядро второго порядка приближается двумерным разложением

$$\tilde{K}_{2,m_1,m_2}(s_1, s_2) = \sum_{i=0}^{m_1-1} \sum_{j=0}^{m_2-1} C_{ij} B_i(s_1) B_j(s_2), \quad (23)$$

где C_{ij} – неизвестные коэффициенты ядра второго порядка. Параметры m_1 и m_2 задают число базисных функций по каждой переменной двумерного ядра.

При таком выборе приближений модель (19) принимает вид

$$\begin{aligned} \hat{y}(t) = & \sum_{i=0}^{m-1} A_i \int_0^t B_i(s) x(t-s) ds + \\ & + \sum_{i=0}^{m_1-1} \sum_{j=0}^{m_2-1} C_{ij} \int_0^t \int_0^t B_i(s_1) B_j(s_2) x(t-s_1) x(t-s_2) ds_1 ds_2. \end{aligned} \quad (24)$$

В формуле (24) неизвестными являются только коэффициенты A_i и C_{ij} . Все интегральные множители определяются известным входным сигналом и выбранными базисными функциями. Поэтому задача становится линейной

относительно неизвестных параметров, несмотря на то что сама модель является нелинейной относительно входного сигнала.

Для компактной записи введем обозначения

$$\beta_{ik}(t) = \int_0^t B_i(s)x(t-s)ds, \quad (25)$$

$$\gamma_{ij}(t) = \int_0^t \int_0^t B_i(s_1)B_j(s_2)x(t-s_1)x(t-s_2)ds_1ds_2. \quad (26)$$

Тогда восстановленный выходной сигнал можно представить в виде

$$\hat{y}(t) = \sum_{i=0}^{m-1} A_i \beta_i(t) + \sum_{i=0}^{m_1-1} \sum_{j=0}^{m_2-1} C_{ij} \gamma_{ij}(t). \quad (27)$$

Эта формула является ключевой для дальнейшего построения численных методов. Она показывает, что после выбора базиса исходная задача идентификации ядер Вольтерра сводится к задаче нахождения коэффициентов линейной комбинации заранее вычисленных функций.

Общее число неизвестных коэффициентов равно

$$M = m + m_1 m_2. \quad (28)$$

Если используется одинаковый порядок аппроксимации $m = m_1 = m_2$, то число неизвестных становится равным $m + m^2$. При увеличении m точность приближения потенциально возрастает, однако одновременно увеличивается размер системы, ухудшается обусловленность и возрастает вычислительная нагрузка. Поэтому выбор параметров m , m_1 и m_2 является важной частью вычислительного эксперимента.

2.3 Полиномы Чебышева как базисные функции

Полиномы Чебышева широко применяются в задачах приближения функций благодаря рекуррентному вычислению и благоприятным аппроксимационным свойствам [9, 10].

В качестве базисных функций в данной работе используются полиномы Чебышева первого рода. Такой выбор связан с тем, что полиномы Чебышева хорошо подходят для аппроксимации гладких функций на конечном интервале и часто применяются в спектральных и проекционных численных методах. Они позволяют компактно представлять функции при сравнительно небольшом числе коэффициентов.

Полиномы Чебышева первого рода $T_n(z)$ определяются на стандартном интервале $[-1, 1]$. Первые два полинома имеют вид

$$T_0(z) = 1, \quad T_1(z) = z. \quad (29)$$

Последующие полиномы вычисляются по рекуррентному соотношению

$$T_{n+1}(z) = 2zT_n(z) - T_{n-1}(z), \quad n = 1, 2, \dots \quad (30)$$

Если переменная s изменяется на интервале $[0, T]$, то для использования полиномов Чебышева необходимо выполнить линейное преобразование к стандартному интервалу:

$$z = \frac{2s}{T} - 1. \quad (31)$$

Тогда базисная функция на $[0, T]$ определяется выражением

$$B_i(s) = T_i\left(\frac{2s}{T} - 1\right). \quad (32)$$

С учетом (32) аппроксимация ядра первого порядка принимает вид

$$\tilde{K}_{1,m}(s) = \sum_{i=0}^{m-1} A_i T_i\left(\frac{2s}{T} - 1\right), \quad (33)$$

а аппроксимация ядра второго порядка записывается как

$$\tilde{K}_{2,m_1,m_2}(s_1, s_2) = \sum_{i=0}^{m_1-1} \sum_{j=0}^{m_2-1} C_{ij} T_i\left(\frac{2s_1}{T} - 1\right) T_j\left(\frac{2s_2}{T} - 1\right). \quad (34)$$

Такое представление удобно для программной реализации. Значения полиномов можно вычислять рекуррентно, не используя символьные

преобразования. Кроме того, для ядра второго порядка применяется тензорная структура базиса: базисная функция по двум переменным строится как произведение двух одномерных базисных функций.

Следует отметить, что использование полиномов Чебышева не отменяет общей постановки задачи. При необходимости в программном комплексе можно заменить базис на другой, например на сплайны или обычные степенные полиномы. Поэтому в архитектуре программы целесообразно отделять модуль вычисления базисных функций от модулей методов идентификации. Такой подход позволит в дальнейшем расширять программный комплекс без полной переработки вычислительного ядра.

2.4 Вычисление интегральных коэффициентов модели

Численное вычисление интегралов является центральной операцией при решении интегральных уравнений и построении их дискретных моделей [11]. После выбора базиса необходимо вычислить значения интегральных коэффициентов, входящих в матрицу системы. Для каждого момента времени t_k вычисляются величины

$$\beta_{ik} = \int_0^{t_k} B_i(s)x(t_k - s)ds, \quad i = 0, 1, \dots, m - 1, \quad (35)$$

а также

$$\gamma_{ijk} = \int_0^{t_k} \int_0^{t_k} B_i(s_1)B_j(s_2)x(t_k - s_1)x(t_k - s_2)ds_1ds_2. \quad (36)$$

Именно эти величины являются коэффициентами при неизвестных A_i и C_{ij} в алгебраической системе.

Если входной сигнал задан аналитически, то интегралы (35) и (36) в отдельных случаях могут быть вычислены аналитически. Такой подход удобен при проверке метода на модельных примерах, поскольку позволяет исключить дополнительную погрешность численного интегрирования. Именно поэтому на раннем этапе исследования полезно использовать простые тестовые сигналы,

например комбинации синусов, косинусов и экспонент.

В прикладной программе чаще всего входной сигнал задан таблицей значений. В этом случае для вычисления интегралов необходимо использовать численные квадратурные формулы. Если временная сетка равномерная, простейшим вариантом является применение формулы трапеций или составных квадратурных формул. При наличии более жестких требований к точности можно использовать квадратуры Гаусса или предварительную интерполяцию входного сигнала.

Пусть шаг равномерной сетки равен h . Тогда интеграл первого порядка можно приближенно заменить суммой

$$\beta_{ik} \approx h \sum_{r=0}^k w_r B_i(t_r) x(t_k - t_r), \quad (37)$$

где w_r – веса выбранной квадратурной формулы. Аналогично двойной интеграл можно аппроксимировать двойной суммой

$$\gamma_{ijk} \approx h^2 \sum_{r=0}^k \sum_{q=0}^k w_r w_q B_i(t_r) B_j(t_q) x(t_k - t_r) x(t_k - t_q). \quad (38)$$

Формулы (37) и (38) важны для программной реализации, поскольку они показывают, как перейти от непрерывной модели к обработке массивов чисел. В дальнейшем при разработке программного комплекса эти вычисления могут быть вынесены в отдельный модуль формирования матрицы системы.

При вычислении $x(t_k - t_r)$ возможны два случая. Если значения $t_k - t_r$ совпадают с узлами исходной сетки, то используются уже имеющиеся значения массива входного сигнала. Если же применяются квадратурные узлы, не совпадающие с сеткой исходных данных, то требуется интерполяция. Для этого можно использовать линейную интерполяцию, сплайн-интерполяцию или сглаживающую аппроксимацию. Выбор способа интерполяции влияет на точность восстановления ядер и должен быть согласован с характером входных данных.

2.5 Формирование матричной модели

После дискретизации исходная задача преобразуется в систему линейных алгебраических уравнений. Устойчивость вычисления коэффициентов зависит от обусловленности матрицы и выбранного способа ее разложения [12].

Для дальнейшего изложения удобно записать задачу в матричной форме. Объединим неизвестные коэффициенты в один вектор

$$q = (A_0, A_1, \dots, A_{m-1}, C_{00}, C_{01}, \dots, C_{m_1-1, m_2-1})^T. \quad (39)$$

Каждому моменту времени t_k соответствует строка матрицы

$$B_k = (\beta_{0k}, \dots, \beta_{m-1,k}, \gamma_{00k}, \gamma_{01k}, \dots, \gamma_{m_1-1, m_2-1,k}). \quad (40)$$

Тогда модельное равенство в точке t_k может быть записано как

$$B_k q \approx y_k. \quad (41)$$

Для набора точек получается матричное соотношение

$$Bq \approx y, \quad (42)$$

где B – матрица коэффициентов, q – вектор неизвестных параметров, а y – вектор значений выходного сигнала.

Матрица B является центральным объектом в обоих методах. В методе коллокации из всех точек выбирается конечный набор узлов, по которым формируется квадратная или почти квадратная система. В методе наименьших квадратов в матрицу включается большее число строк, соответствующих всем или многим точкам наблюдения. Поэтому вычислительная часть обоих методов может быть построена на едином модуле формирования матрицы, а различие между методами будет заключаться в способе выбора строк и способе решения полученной системы.

Такой подход хорошо согласуется с принципами программной инженерии. В разрабатываемом приложении модуль загрузки данных не должен зависеть от

конкретного метода идентификации, а модуль визуализации не должен знать, каким способом были получены коэффициенты. Это позволяет заменить или доработать математический алгоритм, не переписывая интерфейс программы.

2.6 Метод коллокации

2.6.1 Идея метода коллокации

Метод коллокации относится к численным методам, в которых исходное уравнение или модельное равенство требуют выполнить не на всем непрерывном интервале, а в конечном наборе специально выбранных точек. Эти точки называются узлами коллокации. Для интегральных уравнений Вольтерра метод коллокации является одним из распространенных подходов, поскольку позволяет свести интегральную задачу к системе алгебраических уравнений конечного размера [13, 14, 15, 16].

Применительно к задаче идентификации параметров модели Вольтерра метод коллокации имеет следующий смысл. Известно, что истинный выход $y(t)$ и восстановленный выход $\hat{y}(t)$ должны быть близки на всем интервале $[0, T]$. Вместо требования равенства для всех t выбирается конечный набор узлов

$$\tau_0, \tau_1, \dots, \tau_N \in [0, T]. \quad (43)$$

В этих узлах накладываются условия

$$\hat{y}(\tau_k) = y(\tau_k), \quad k = 0, 1, \dots, N. \quad (44)$$

Если число условий совпадает с числом неизвестных коэффициентов, то получается квадратная система линейных алгебраических уравнений.

Такой подход удобен тем, что позволяет свести непрерывную задачу восстановления ядер к конечному набору алгебраических соотношений. В программной реализации это дает возможность заранее определить размерность расчетной матрицы, сформировать ее элементы по выбранным узлам и передать полученную систему стандартному численному алгоритму решения.

2.6.2 Получение системы уравнений

Подставим в условия (44) разложения ядер (22) и (23). Тогда для каждого узла τ_k получаем равенство

$$\begin{aligned} y(\tau_k) = & \sum_{i=0}^{m-1} A_i \int_0^{\tau_k} B_i(s)x(\tau_k - s)ds + \\ & + \sum_{i=0}^{m_1-1} \sum_{j=0}^{m_2-1} C_{ij} \int_0^{\tau_k} \int_0^{\tau_k} B_i(s_1)B_j(s_2)x(\tau_k - s_1)x(\tau_k - s_2)ds_1ds_2. \end{aligned} \quad (45)$$

Введя обозначения

$$\beta_{ik} = \int_0^{\tau_k} B_i(s)x(\tau_k - s)ds, \quad (46)$$

$$\gamma_{ijk} = \int_0^{\tau_k} \int_0^{\tau_k} B_i(s_1)B_j(s_2)x(\tau_k - s_1)x(\tau_k - s_2)ds_1ds_2, \quad (47)$$

получаем систему

$$y_k = \sum_{i=0}^{m-1} A_i \beta_{ik} + \sum_{i=0}^{m_1-1} \sum_{j=0}^{m_2-1} C_{ij} \gamma_{ijk}, \quad k = 0, 1, \dots, N. \quad (48)$$

Если $N + 1 = M$, где $M = m + m_1 m_2$, то система (48) является квадратной. Ее можно записать в матричной форме

$$B_c q = y_c. \quad (49)$$

Здесь B_c – коллокационная матрица, строки которой построены по узлам τ_k , а y_c – вектор значений выходного сигнала в этих узлах.

2.6.3 Выбор узлов коллокации

Качество метода коллокации существенно зависит от выбора узлов. В простейшем варианте можно использовать равномерную сетку

$$\tau_k = \frac{kT}{N}, \quad k = 0, 1, \dots, N. \quad (50)$$

Такой выбор прост и удобен для программной реализации. Если входной и выходной сигналы заданы на равномерной сетке, то узлы коллокации можно совместить с частью исходных точек.

В более точных схемах могут использоваться неравномерные узлы, например узлы, связанные с ортогональными полиномами. Для полиномов Чебышева естественным вариантом являются узлы Чебышева на $[-1, 1]$, которые затем переносятся на интервал $[0, T]$. Однако в программном комплексе на первом этапе целесообразно предусмотреть простой и понятный вариант: выбор равномерных узлов или выбор заданного числа точек из исходной сетки. Это позволит проверить корректность всей цепочки вычислений, а затем при необходимости усложнить схему выбора узлов.

Если данные содержат шум, точное выполнение условий $\hat{y}(\tau_k) = y(\tau_k)$ может быть нежелательным. В этом случае коллокация может давать решение, слишком сильно зависящее от отдельных точек. Поэтому при работе с зашумленными сигналами полезно сравнивать результат коллокации с методом наименьших квадратов.

2.6.4 Алгоритм метода коллокации

Вычислительная схема метода коллокации для данной задачи может быть записана следующим образом:

- задать параметры аппроксимации m, m_1, m_2 ;
- определить число неизвестных $M = m + m_1 m_2$;
- выбрать M узлов коллокации на интервале $[0, T]$;
- вычислить значения базисных функций в узлах интегрирования;
- вычислить коэффициенты β_{ik} и γ_{ijk} ;
- сформировать матрицу B_c и вектор y_c ;
- решить систему $B_c q = y_c$;
- восстановить коэффициенты A_i и C_{ij} из вектора q ;
- вычислить $\hat{y}(t_k)$ на всей сетке наблюдения;

- построить невязку и оценить ошибку.

Важным преимуществом этой схемы является ее наглядность. Она почти напрямую повторяет математическую постановку: модельное равенство выполняется в выбранных точках. Поэтому метод удобно объяснять и демонстрировать в рамках выпускной квалификационной работы.

2.6.5 Достоинства и ограничения метода коллокации

К достоинствам метода коллокации можно отнести простоту идеи, связь с исходной интегральной моделью и возможность получать систему уравнений относительно коэффициентов ядер. Метод хорошо подходит для модельных задач, в которых входной и выходной сигналы заданы точно, а также для ситуаций, когда требуется проверить корректность переноса математической схемы в программу.

Ограничения метода связаны прежде всего с устойчивостью. Если матрица B_c плохо обусловлена, то небольшие ошибки в исходных данных могут привести к большим ошибкам в найденных коэффициентах. Кроме того, при неудачном выборе узлов возможна ситуация, когда модель хорошо совпадает с выходом в узлах коллокации, но хуже ведет себя между ними. Поэтому результат метода необходимо оценивать не только по точкам, использованным при построении системы, но и по всей сетке наблюдения.

В рамках программного комплекса метод коллокации будет полезен как основной прямой способ идентификации. Он позволяет получить коэффициенты ядер K_1 и K_2 , построить восстановленный сигнал и сравнить его с исходным выходом. При этом пользователь сможет изменять параметры m , m_1 , m_2 и число узлов, наблюдая изменение невязки.

Практическое применение метода требует согласованного выбора числа базисных функций и числа узлов. Если узлов недостаточно, система не содержит всей необходимой информации о поведении модели. Если же число узлов чрезмерно увеличивается без изменения структуры аппроксимации, то расчет

может стать более чувствительным к численным погрешностям. Поэтому в программе параметры метода рассматриваются как настраиваемые величины, влияющие на точность и устойчивость результата.

2.7 Метод наименьших квадратов

2.7.1 Идея метода наименьших квадратов

Метод наименьших квадратов является стандартным средством оценивания параметров по избыточному набору наблюдений; его теория и вычислительные алгоритмы подробно изложены в [17].

Метод наименьших квадратов применяется в задачах, где необходимо подобрать параметры модели по набору наблюдений. В отличие от метода коллокации, здесь не требуется точного выполнения равенства $\hat{y}(t_k) = y(t_k)$ в каждой выбранной точке. Вместо этого ищется такой набор коэффициентов, при котором суммарное квадратичное отклонение восстановленного сигнала от исходного минимально.

Пусть имеется $N + 1$ точек наблюдения, причем количество точек больше числа неизвестных коэффициентов. Тогда для каждого t_k модель дает значение $\hat{y}(t_k, q)$, зависящее от вектора параметров q . Невязка в точке t_k имеет вид

$$r_k(q) = y_k - \hat{y}(t_k, q). \quad (51)$$

Критерий метода наименьших квадратов задается функционалом

$$J(q) = \sum_{k=0}^N r_k^2(q) = \sum_{k=0}^N (y_k - \hat{y}(t_k, q))^2 \rightarrow \min. \quad (52)$$

Поскольку модель (27) линейна относительно неизвестных коэффициентов, задача (52) является линейной задачей наименьших квадратов.

2.7.2 Матричная форма метода

Используя матричное обозначение $Bq \approx y$, функционал (52) можно записать как

$$J(q) = \|Bq - y\|_2^2 \rightarrow \min. \quad (53)$$

Необходимое условие минимума приводит к нормальным уравнениям

$$B^T Bq = B^T y. \quad (54)$$

Если матрица B имеет полный столбцовый ранг, то решение нормальных уравнений единственно. Вектор q задает коэффициенты A_i и C_{ij} , минимизирующие сумму квадратов отклонений.

В вычислительной практике нормальные уравнения не всегда являются наиболее устойчивым способом решения, поскольку матрица $B^T B$ может быть хуже обусловлена, чем исходная матрица B . Поэтому в программной реализации можно использовать QR-разложение или сингулярное разложение. Для выпускной квалификационной работы важно не только получить формулу (54), но и отметить, что конкретный численный способ решения должен учитывать устойчивость вычислений.

2.7.3 Применение МНК к модели Вольтерра

Для применения метода наименьших квадратов к модели Вольтерра необходимо выполнить те же подготовительные действия, что и для метода коллокации: выбрать базис, вычислить интегральные коэффициенты β_{ik} и γ_{ijk} , сформировать матрицу B . Отличие состоит в том, что в матрицу включаются не только специальные узлы, а все выбранные точки наблюдения или достаточно большое их подмножество.

Строка матрицы для точки t_k имеет вид

$$B_k = (\beta_{0k}, \dots, \beta_{m-1,k}, \gamma_{00k}, \dots, \gamma_{m_1-1,m_2-1,k}). \quad (55)$$

После формирования всех строк получается переопределенная система

$$\begin{pmatrix} B_0 \\ B_1 \\ \vdots \\ B_N \end{pmatrix} q \approx \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_N \end{pmatrix}. \quad (56)$$

В отличие от квадратной системы коллокации, такая система обычно не имеет точного решения. Метод наименьших квадратов выбирает решение, которое является наилучшим в среднем квадратичном смысле.

2.7.4 Алгоритм метода наименьших квадратов

Алгоритм МНК для идентификации параметров модели Вольтерра можно представить следующим образом:

- загрузить массивы значений x_k и y_k ;
- выбрать параметры m , m_1 , m_2 и длину памяти;
- построить базисные функции на интервале $[0, T]$;
- для каждой точки наблюдения вычислить β_{ik} и γ_{ijk} ;
- сформировать матрицу B и вектор y ;
- решить задачу $\|Bq - y\|_2^2 \rightarrow \min$;
- восстановить коэффициенты A_i и C_{ij} ;
- вычислить восстановленный выходной сигнал;
- построить график невязки и вычислить численные ошибки.

Такой алгоритм хорошо подходит для программной реализации, поскольку он использует все доступные данные. Это особенно важно, если входной и выходной сигналы получены из файла Excel и содержат большое число отсчетов.

2.7.5 Преимущества и ограничения МНК

При наличии плохо обусловленной матрицы решение может быть чувствительным к шуму. В работах по дискретным обратным задачам [18] для стабилизации подобных вычислений применяются масштабирование,

сингулярное разложение и регуляризация.

Основное преимущество метода наименьших квадратов заключается в устойчивости к локальным ошибкам. Если отдельные значения выходного сигнала содержат шум, то МНК не пытается точно пройти через каждую точку, а подбирает параметры по всему набору данных. Поэтому при обработке экспериментальных сигналов этот метод часто оказывается более устойчивым, чем точная коллокационная схема.

Другим преимуществом является возможность использовать произвольное число точек. Если данных много, то система становится переопределенной, и это может улучшить качество оценки параметров. Кроме того, МНК удобно применять для сравнения различных порядков модели: можно рассчитывать ошибку для разных значений m , m_1 , m_2 и выбирать вариант с наилучшим соотношением точности и устойчивости.

Ограничения метода также необходимо учитывать. Если модель слишком сложная, то МНК может подстраиваться под шум. Если матрица B плохо обусловлена, то найденные коэффициенты могут иметь большие значения, хотя восстановленный сигнал будет выглядеть приемлемо. Поэтому в некоторых случаях требуется регуляризация, ограничение порядка модели или предварительная нормализация данных.

2.8 Невязка и критерии качества идентификации

После нахождения коэффициентов модели необходимо оценить, насколько хорошо восстановленный сигнал совпадает с исходным. Основной величиной для такой оценки является невязка

$$r(t) = y(t) - \hat{y}(t). \quad (57)$$

В дискретном виде она записывается как

$$r_k = y_k - \hat{y}_k, \quad k = 0, 1, \dots, N. \quad (58)$$

График невязки позволяет увидеть, на каких участках интервала модель

работает хуже. Это важно, потому что одно численное значение ошибки не всегда показывает характер расхождения между сигналами.

В качестве численного критерия можно использовать максимальную норму

$$\varepsilon_{\max} = \max_{0 \leq k \leq N} |r_k|. \quad (59)$$

Такая ошибка показывает наибольшее отклонение модели от исходного сигнала. В материалах по коллокационной постановке для задачи идентификации также используется контроль невязки в максимальной норме, что удобно для оценки качества аппроксимации на всем интервале.

Другой распространенный критерий – среднеквадратичная ошибка

$$\varepsilon_{RMSE} = \sqrt{\frac{1}{N+1} \sum_{k=0}^N r_k^2}. \quad (60)$$

Эта величина особенно удобна для метода наименьших квадратов, поскольку напрямую связана с минимизируемым функционалом (52). Для сравнения результатов на сигналах разного масштаба можно использовать относительную ошибку

$$\varepsilon_{rel} = \frac{\sqrt{\sum_{k=0}^N r_k^2}}{\sqrt{\sum_{k=0}^N y_k^2}}. \quad (61)$$

В программном комплексе целесообразно выводить несколько показателей: среднеквадратичную ошибку, максимальное отклонение и график невязки. Такой набор результатов позволит пользователю не только получить одно число, но и визуальную оценку качества идентификации.

2.9 Сравнение метода коллокации и метода наименьших квадратов

Метод коллокации и метод наименьших квадратов решают одну и ту же задачу, но используют разные критерии построения модели. В коллокации модельное равенство выполняется в выбранных узлах. В МНК минимизируется

суммарная ошибка по набору точек. Поэтому методы могут давать разные коэффициенты даже при одинаковом базисе и одинаковых исходных данных.

Метод коллокации удобен, когда данные заданы точно и требуется получить решение, непосредственно связанное с исходным интегральным равенством. Он хорошо подходит для модельных примеров и проверки корректности математической схемы. Метод наименьших квадратов предпочтителен, когда данных много или они содержат шум. Он использует больше информации о сигнале и обычно дает более сглаженное приближение.

Основные различия двух методов приведены в таблице 1.

Таблица 1 – Сравнение метода коллокации и метода наименьших квадратов

Критерий	Метод коллокации	Метод наименьших квадратов
Основная идея	выполнение равенства в выбранных узлах	минимизация суммарной квадратичной ошибки
Тип системы	чаще квадратная система	чаще переопределенная система
Использование данных	используется набор узлов	может использоваться вся сетка наблюдений
Чувствительность к шуму	более высокая	обычно ниже
Проверка качества	невязка на всем интервале после решения	невязка является частью критерия
Программная реализация	требуется выбор узлов	требуется решение задачи МНК

Для выпускной квалификационной работы наличие двух методов является полезным. Это позволяет не ограничиваться одной вычислительной схемой, а сравнить результаты: построить графики восстановленного сигнала, вычислить ошибки и проанализировать влияние параметров m , m_1 , m_2 . Кроме того, такая структура хорошо соответствует программному комплексу: пользователь выбирает метод, после чего программа выполняет соответствующий алгоритм и выводит результаты в едином формате.

2.10 Выбор параметров модели

К основным параметрам модели относятся количества базисных функций первого ядра m и второго ядра m_1, m_2 , а для метода наименьших квадратов также количество используемых точек. Эти параметры определяют размерность задачи и влияют на качество результата.

Если параметры аппроксимации слишком малы, модель не сможет описать сложную зависимость между входом и выходом. В этом случае график восстановленного сигнала будет заметно отличаться от исходного выхода, а невязка будет иметь систематический характер. Если же параметры слишком велики, число неизвестных возрастает, матрица может стать плохо обусловленной, а решение может стать неустойчивым. На практике это может проявляться в очень больших значениях отдельных коэффициентов при относительно небольшой ошибке восстановления.

Выбор параметров должен выполняться экспериментально. В программном комплексе пользователь сможет изменять параметры, запускать расчет и сравнивать результаты. Для дипломной работы это важно, поскольку позволяет показать не только работу одного фиксированного алгоритма, но и исследование поведения модели при разных настройках.

2.11 Алгоритмическая схема идентификации

Общая алгоритмическая схема идентификации параметров модели Вольтерра включает следующие этапы:

- загрузить входной и выходной сигналы из файла или сформировать модельный пример;
- проверить корректность данных и построить графики $x(t)$ и $y(t)$;
- задать параметры модели m, m_1, m_2 и число используемых точек;
- построить базисные функции на интервале $[0, T]$;
- вычислить интегральные коэффициенты β_{ik} и γ_{ijk} ;

- сформировать матрицу системы;
- выбрать метод решения: коллокацию или МНК;
- найти вектор коэффициентов q ;
- разделить вектор q на коэффициенты A_i и C_{ij} ;
- вычислить восстановленный выход $\hat{y}(t_k)$;
- построить график невязки и вычислить показатели ошибки;
- вывести результаты пользователю и при необходимости сохранить их в файл.

Эта схема непосредственно переносится в структуру программного комплекса. Загрузка данных, построение базиса, формирование матрицы, решение системы и визуализация результатов могут быть реализованы как отдельные модули. Такой подход упрощает разработку и последующее сопровождение программы.

2.12 Выводы по второй главе

Во второй главе рассмотрены численные методы идентификации параметров модели Вольтерра второго порядка. Показано, что после аппроксимации ядер по конечной системе базисных функций задача восстановления K_1 и K_2 сводится к нахождению конечного набора коэффициентов. Для этого вводятся коэффициенты β_{ik} и γ_{ijk} , формирующие матрицу системы.

Подробно рассмотрены два метода идентификации: метод коллокации и метод наименьших квадратов. Метод коллокации основан на выполнении модельного равенства в выбранных узлах и приводит к системе линейных алгебраических уравнений. Метод наименьших квадратов основан на минимизации суммарной квадратичной невязки и позволяет использовать большое число точек наблюдения. Оба метода применимы к одной и той же матричной модели, что удобно для программной реализации.

Также рассмотрены критерии качества идентификации: максимальная

невязка, среднеквадратичная ошибка и относительная ошибка. Эти величины будут использоваться в программном комплексе для анализа результата. Полученные в данной главе формулы и алгоритмические схемы являются основой для последующего проектирования и реализации вычислительного ядра приложения.

3 Проектирование программного комплекса

После определения математической модели динамической системы и рассмотрения численных методов идентификации необходимо перейти к проектированию программного комплекса. На данном этапе формируется представление о том, какие функции должна выполнять программа, какие данные она должна получать на входе, какие результаты должны быть доступны пользователю и каким образом вычислительная часть должна быть связана с графическим интерфейсом.

Разрабатываемое приложение предназначено не только для выполнения отдельных вычислений, но и для организации полного цикла работы с моделью: загрузки входных данных, выбора метода идентификации, задания параметров модели, запуска расчета, просмотра графиков и сохранения результатов. Поэтому при проектировании важно выделить отдельные логические части системы и определить связи между ними. Такой подход позволяет упростить дальнейшую реализацию, тестирование и развитие программы.

При проектировании программного комплекса использовались общие принципы разработки программных систем: выделение требований, разделение программы на модули, отделение пользовательского интерфейса от вычислительного ядра, а также представление структуры программы с помощью диаграмм UML [19, 20]. В главе предусмотрены диаграмма вариантов использования, диаграмма компонентов, диаграмма классов и диаграмма деятельности. Подробные спецификации основных вариантов использования приведены в приложении А.

3.1 Назначение и область применения программного комплекса

Программный комплекс предназначен для идентификации параметров динамической системы, рассматриваемой в виде модели «черного ящика» с одним входом и одним выходом. Пользователь задает или загружает входной сигнал

$x(t)$ и выходной сигнал $y(t)$, после чего программа выполняет подбор параметров интегрально-степенной модели Вольтерра второго порядка. Результатом работы программы являются коэффициенты аппроксимации ядер, восстановленный выходной сигнал, график невязки и численное значение ошибки.

Основная область применения разрабатываемого комплекса связана с анализом систем, для которых известны экспериментальные или расчетные данные о входе и выходе, но внутренняя структура объекта либо неизвестна, либо не используется явно. В таком случае объект рассматривается как оператор, преобразующий входной сигнал в выходной. Такой подход удобен при исследовании технических, физических и информационных систем, где построение полной аналитической модели затруднено или избыточно для прикладной задачи.

В рамках данной выпускной квалификационной работы программный комплекс ориентирован на математическую модель, не привязанную к конкретной предметной области. Это означает, что программа должна работать не с медицинскими или радиотехническими величинами как таковыми, а с числовыми сигналами. Такая постановка делает приложение более универсальным: один и тот же вычислительный модуль может применяться к различным наборам данных при условии, что они представлены в виде согласованных последовательностей значений $x(t)$ и $y(t)$.

С точки зрения пользователя программный комплекс должен обеспечивать следующие возможности:

- загрузку входного и выходного сигналов из файла;
- отображение исходных данных в табличной форме;
- построение графиков входного и выходного сигналов;
- выбор метода идентификации параметров модели;
- задание численных параметров метода;
- расчет коэффициентов модели;
- построение восстановленного выходного сигнала;

- вычисление и визуализацию невязки;
- сохранение результатов для последующего анализа.

Таким образом, назначение программы заключается в автоматизации вычислительного эксперимента по идентификации параметров динамической системы. При этом пользователь освобождается от ручного составления систем уравнений и выполнения промежуточных расчетов, а основные результаты представляются в наглядном виде.

3.2 Анализ требований к программному комплексу

На этапе анализа требований необходимо определить, какие задачи должен решать программный комплекс и какими свойствами он должен обладать. Требования целесообразно разделить на функциональные и нефункциональные. Функциональные требования описывают действия, которые должна выполнять система, а нефункциональные требования задают ограничения на качество, удобство использования, надежность и расширяемость программы.

Пользователем программного комплекса является исследователь или студент, выполняющий идентификацию параметров модели по имеющимся входным и выходным данным. Предполагается, что пользователь обладает базовым пониманием математической постановки задачи, но не должен вручную выполнять все вычисления. Поэтому интерфейс должен быть достаточно простым: основные операции должны быть доступны через меню, кнопки и вкладки.

В качестве главного сценария использования можно выделить следующий порядок действий. Пользователь запускает программу, загружает файл с данными, проверяет отображение сигналов, выбирает один из двух методов идентификации, задает параметры модели, выполняет расчет и анализирует полученные графики. При необходимости пользователь сохраняет результаты в файл.

Для наглядного представления функциональных возможностей программного комплекса построена диаграмма вариантов использования, представленная на рисунке 1. На ней в качестве основного действующего лица

выступает пользователь, а основными вариантами использования являются загрузка данных, выбор метода, выполнение идентификации, просмотр результатов и сохранение результатов.



Рисунок 1 – Диаграмма вариантов использования программного комплекса

Диаграмма показывает, что пользователь взаимодействует с программой через ограниченный набор основных операций. Вариант использования «Загрузить входные данные» уточняется вариантами загрузки таблицы Excel и демонстрационного набора, а просмотр результатов расширяется просмотром графиков, коэффициентов и ядер модели. Подробные спецификации вариантов

использования приведены в приложении А, что позволяет не перегружать основной текст повторяющимися таблицами.

3.3 Функциональные требования

Функциональные требования определяют основные операции, которые должны быть реализованы в программном комплексе. Их состав сформирован с учетом общих требований к содержанию технического задания на программное изделие [21]. Они непосредственно связаны с задачами идентификации динамической системы и с работой пользователя в графическом интерфейсе.

Основные функциональные требования представлены в таблице 2.

Таблица 2 – Функциональные требования к программному комплексу

Номер	Требование	Описание
1	Загрузка входных данных	загрузка входного и выходного сигналов из файла табличного формата
2	Отображение данных	вывод времени, входного и выходного сигналов в табличной форме
3	Визуализация исходных сигналов	построение графиков $x(t)$ и $y(t)$
4	Выбор метода идентификации	выбор метода коллокации или метода наименьших квадратов
5	Задание параметров модели	ввод количества базисных функций m, m_1, m_2
6	Расчет коэффициентов	вычисление коэффициентов аппроксимации ядер модели Вольтерра
7	Построение восстановленного сигнала	вычисление выходного сигнала $\hat{y}(t)$ по найденным коэффициентам

Продолжение таблицы 2

Номер	Требование	Описание
8	Расчет невязки	вычисление $r(t) = y(t) - \hat{y}(t)$ и показателей ошибки
9	Вывод результатов	отображение таблиц коэффициентов и графиков
10	Визуализация ядер	построение графика $K_1(s)$ и поверхности $K_2(s_1, s_2)$
11	Сохранение результатов	сохранение численных результатов и изображений графиков

Данные требования отражают минимальный набор функций, необходимый для выполнения вычислительного эксперимента. В дальнейшем программный комплекс может быть расширен, например, за счет добавления новых форматов входных данных, дополнительных способов визуализации или новых методов идентификации.

3.4 Нефункциональные требования

Нефункциональные требования описывают свойства программного комплекса, не связанные напрямую с отдельными действиями пользователя, но определяющие качество системы в целом. При их формировании учитывались характеристики модели качества программного продукта ISO/IEC 25010 [22]. Для разрабатываемого приложения важны удобство использования, корректность вычислений, возможность расширения и простота сопровождения.

К основным нефункциональным требованиям относятся следующие:

- удобство интерфейса: основные действия пользователя должны быть доступны из главного окна программы;
- наглядность результатов: исходные и восстановленные сигналы должны отображаться в графической форме;

- модульность: загрузка данных, вычисления, визуализация и экспорт должны быть реализованы как отдельные логические части программы;
- расширяемость: структура проекта должна позволять добавлять новые методы идентификации без полной переработки интерфейса;
- повторяемость расчетов: при одинаковых входных данных и параметрах программа должна выдавать одинаковые результаты;
- устойчивость к ошибкам ввода: программа должна проверять наличие данных и корректность основных параметров;
- совместимость: приложение должно работать в операционной системе Windows, так как разрабатывается как настольная программа;
- сопровождаемость: код должен быть разделен на классы и модули с понятным назначением.

Особое значение имеет разделение пользовательского интерфейса и математического ядра. Если вычислительный алгоритм будет жестко связан с элементами формы, то дальнейшая доработка программы станет затруднительной. Поэтому при проектировании принято решение выделить отдельные классы для хранения сигналов, параметров идентификации, результатов расчета и реализации методов.

3.5 Формат входных и выходных данных

Входными данными для программного комплекса являются дискретные значения входного и выходного сигналов. В общем случае данные могут быть представлены в виде трех столбцов:

$$t_k, \quad x(t_k), \quad y(t_k), \quad k = 1, 2, \dots, N, \quad (62)$$

где t_k – момент времени, $x(t_k)$ – значение входного сигнала, $y(t_k)$ – значение выходного сигнала, N – число отсчетов.

Если в файле отсутствует отдельный столбец времени, временная сетка может быть сформирована автоматически по номеру отсчета. В простейшем

случае принимается

$$t_k = k, \quad k = 0, 1, 2, \dots, N - 1. \quad (63)$$

При необходимости в дальнейшем может быть добавлен параметр шага дискретизации Δt , тогда временная сетка будет иметь вид

$$t_k = k\Delta t. \quad (64)$$

На текущем этапе проектирования в качестве основного формата входных данных выбран файл Excel. Такой выбор обусловлен тем, что табличные данные удобно подготавливать, просматривать и корректировать средствами электронных таблиц. Кроме того, использование Excel-файла позволяет быстро проводить эксперименты с различными наборами сигналов без изменения кода программы.

Поддерживаемые варианты структуры входного файла представлены в таблице 3.

Таблица 3 – Варианты структуры входного файла

Формат	Описание
Два столбца	первый столбец содержит значения $x(t)$, второй столбец содержит значения $y(t)$; время формируется автоматически
Три столбца	первый столбец содержит значения t , второй – $x(t)$, третий – $y(t)$

Выходными данными программного комплекса являются:

- коэффициенты аппроксимации ядра первого порядка;
- коэффициенты аппроксимации ядра второго порядка;
- значения восстановленного выходного сигнала;
- значения невязки;
- численное значение ошибки;

- графики исходных и восстановленных сигналов.

Для сохранения результатов целесообразно использовать текстовый табличный формат, например CSV. Он удобен тем, что может быть открыт как в электронных таблицах, так и в текстовом редакторе. В дальнейшем при необходимости может быть реализован экспорт в Excel или формирование отдельного отчета.

3.6 Выбор средств разработки

При выборе средств разработки учитывались следующие факторы: необходимость создания настольного приложения с графическим интерфейсом, работа в операционной системе Windows, наличие средств для построения графиков, возможность чтения Excel-файлов и поддержка библиотек линейной алгебры. С учетом этих требований в качестве основного языка программирования выбран C#, а в качестве платформы – .NET.

Для реализации графического интерфейса выбран Windows Forms. Данная технология подходит для разработки учебного и инженерного настольного приложения, так как позволяет быстро создавать окна, кнопки, таблицы, меню и другие элементы интерфейса. Для выпускной квалификационной работы это является важным преимуществом, поскольку основное внимание должно быть уделено математической модели, алгоритмам идентификации и анализу результатов, а не созданию сложного интерфейса.

В проекте предполагается использовать следующие программные средства и библиотеки:

- C# – основной язык программирования;
- .NET – платформа выполнения приложения;
- Windows Forms – технология построения графического интерфейса;
- ClosedXML – библиотека для чтения Excel-файлов;
- ScottPlot – библиотека для построения графиков;
- MathNet.Numerics – библиотека для выполнения операций линейной

алгебры;

- Git – система контроля версий проекта.

Такой набор средств позволяет разделить работу на несколько независимых направлений: разработку интерфейса, реализацию ввода-вывода, построение графиков и создание вычислительного ядра. При необходимости отдельные библиотеки могут быть заменены без изменения общей архитектуры программы.

3.7 Проектирование архитектуры программного комплекса

Архитектура программного комплекса должна отражать разделение приложения на основные функциональные модули. В разрабатываемой программе можно выделить следующие части:

- модуль пользовательского интерфейса;
- модуль загрузки входных данных;
- модуль хранения сигналов и параметров;
- математическое ядро;
- модуль метода коллокации;
- модуль метода наименьших квадратов;
- модуль визуализации результатов;
- модуль экспорта результатов.

Модуль пользовательского интерфейса отвечает за взаимодействие с пользователем. Через него выполняется выбор файла, ввод параметров, запуск расчета и отображение результатов. Модуль загрузки данных преобразует содержимое Excel-файла во внутреннее представление сигналов. Математическое ядро получает подготовленные данные и параметры модели, после чего передает управление выбранному методу идентификации.

Ключевой особенностью архитектуры является наличие общего интерфейса для методов идентификации. Это позволяет рассматривать метод коллокации и метод наименьших квадратов как взаимозаменяемые реализации одной общей операции: по входным данным и параметрам модели получить

результат идентификации. Такой подход облегчает добавление новых методов в будущем.

Для визуального представления архитектуры программного комплекса построена диаграмма компонентов, приведенная на рисунке 2. Она показывает основные модули приложения и связи между ними.

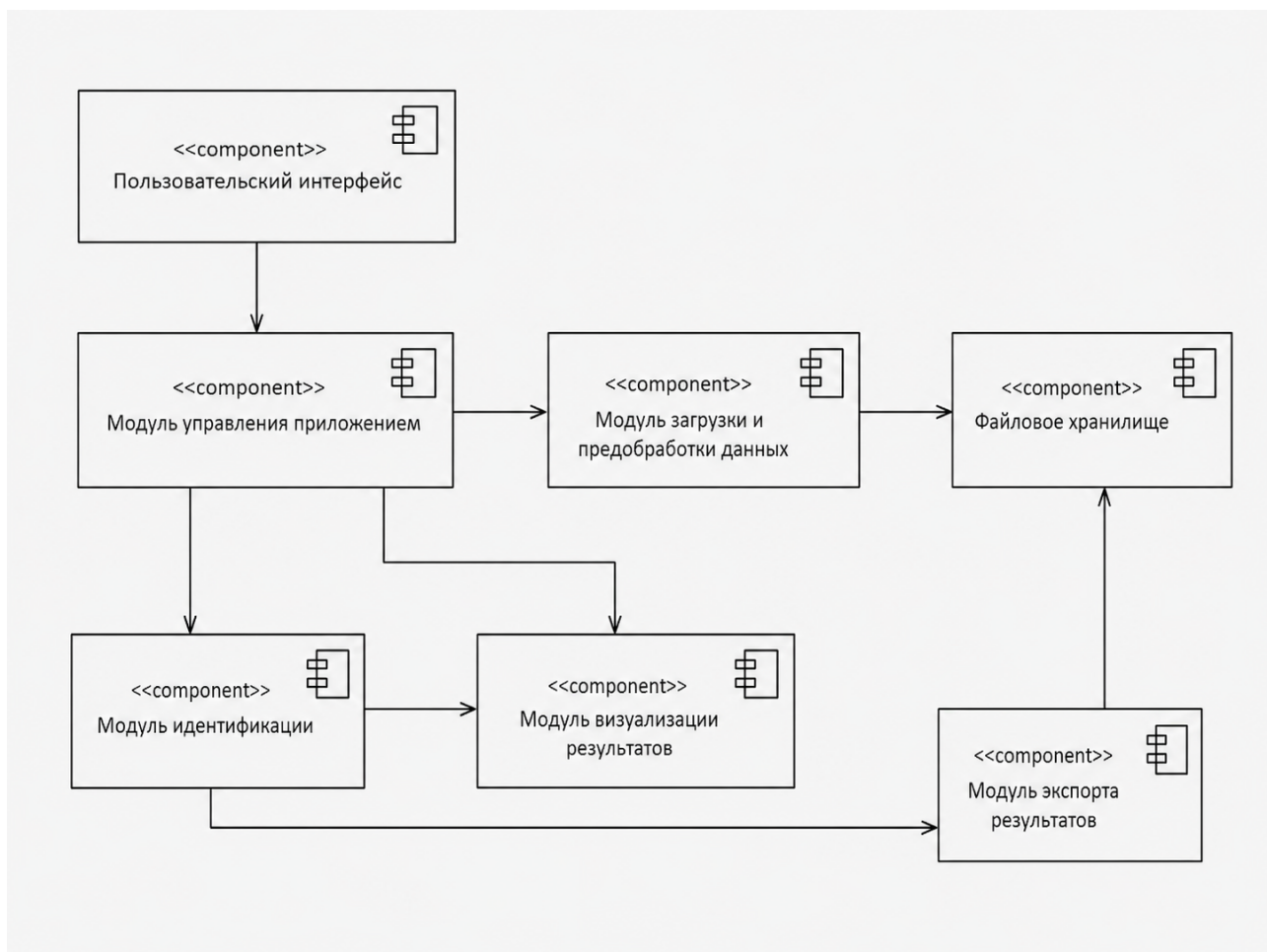


Рисунок 2 – Диаграмма компонентов программного комплекса

На диаграмме выделен компонент пользовательского интерфейса, через который пользователь запускает загрузку данных, расчет и просмотр результатов. Модуль управления приложением связывает интерфейс с остальными частями системы и передает подготовленные параметры в модуль идентификации. Модуль загрузки и предобработки данных отвечает за чтение табличных файлов и подготовку временной шкалы. Модуль идентификации содержит вычислительные процедуры для определения коэффициентов модели. Модуль визуализации

результатов строит графики сигналов, невязки и ядер, а модуль экспорта результатов обеспечивает сохранение данных и изображений во внешние файлы. Файловое хранилище на диаграмме обозначает внешние Excel-, CSV- и графические файлы, с которыми взаимодействует программа.

3.8 Проектирование пользовательского интерфейса

Пользовательский интерфейс должен обеспечивать выполнение всех основных действий без необходимости обращения к исходному коду программы. В качестве основы выбран подход с одним главным окном, содержащим меню, область параметров и вкладки с данными и результатами.

В верхней части главного окна располагается меню, через которое пользователь загружает файл, формирует демонстрационный набор, сохраняет результаты или завершает работу. Ниже находится область параметров модели, в которой выбирается метод идентификации и задается количество базисных функций m , m_1 , m_2 . Для коллокации число узлов вычисляется автоматически, а для метода наименьших квадратов задается количество используемых точек. Основная часть окна разделена на вкладки, что позволяет не перегружать интерфейс большим количеством элементов одновременно.

Предполагается использовать следующие вкладки:

- «Входные данные» – таблица значений и графики исходных сигналов;
- «Результаты» – график исходного и восстановленного выходного сигнала, а также график невязки;
- «Коэффициенты» – таблицы коэффициентов ядер первого и второго порядка.

Такое построение интерфейса соответствует логике работы пользователя. Сначала он проверяет исходные данные, затем выполняет расчет и анализирует результаты. Разделение информации по вкладкам делает окно более понятным и удобным для работы.

Общий вид главного окна реализованной программы представлен на

рисунке 3. Вместо отдельного изображения конструктора Visual Studio в работе используется снимок рабочего окна приложения, поскольку именно он показывает фактическое расположение элементов интерфейса после запуска программы.

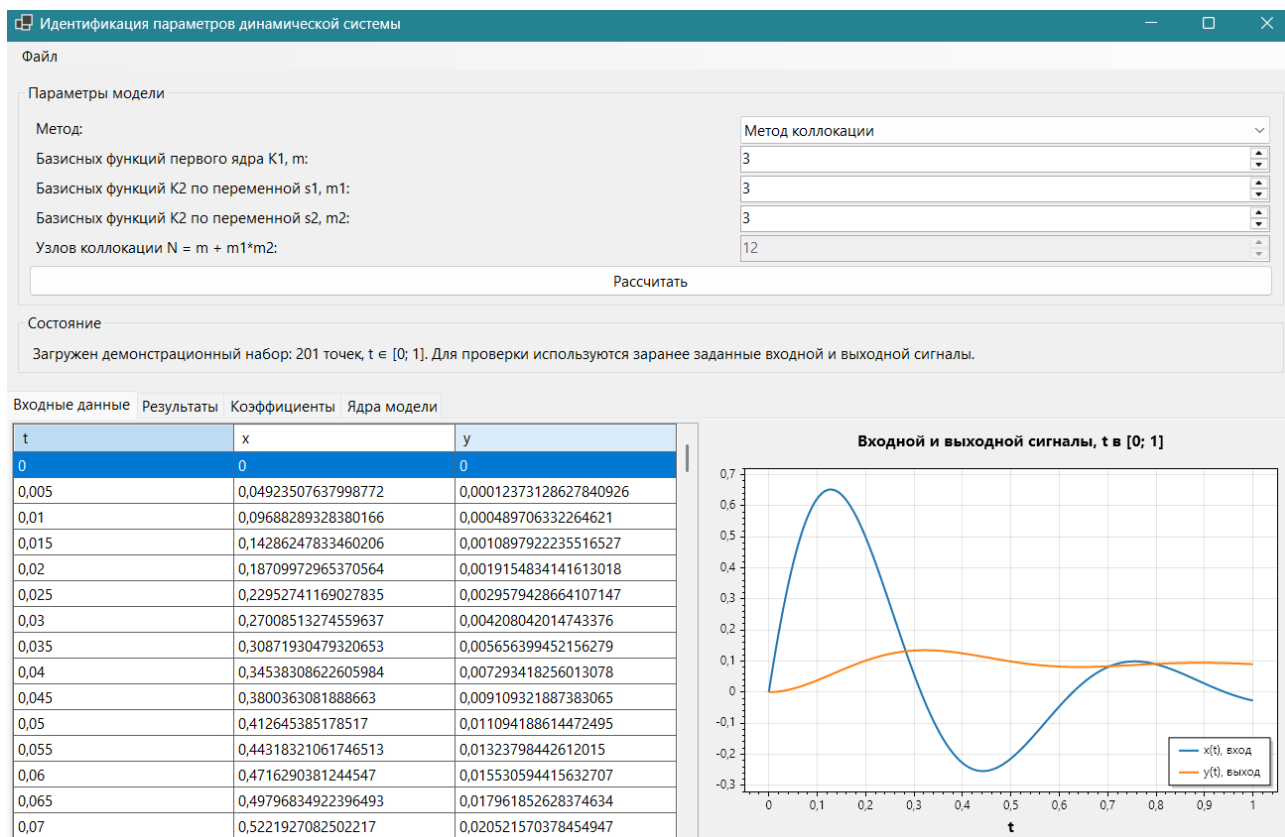


Рисунок 3 – Главное окно программного комплекса

Основные области окна соответствуют последовательности действий пользователя: загрузке данных, настройке параметров, запуску расчета и анализу результатов. В верхней части располагаются меню и параметры модели, ниже выводится состояние программы, а центральная область отведена под вкладки с входными данными, результатами, коэффициентами и ядрами модели.

3.9 Проектирование структуры классов и модулей

Для обеспечения сопровождаемости и расширяемости программы необходимо заранее определить основные классы и их назначение. Предлагаемая структура проекта разделяется на несколько логических областей: пользовательский интерфейс, модели данных, ввод-вывод и математическое

ядро.

К основным классам программного комплекса относятся:

- MainForm – главное окно приложения;
- SignalPoint – отдельная точка сигнала, содержащая значения t , $x(t)$, $y(t)$;
- SignalData – набор точек входного и выходного сигналов;
- IdentificationParameters – параметры выбранного метода и модели;
- IdentificationResult – результат идентификации;
- ExcelSignalReader – класс загрузки данных из Excel-файла;
- Identifier – общий интерфейс метода идентификации;
- CollocationIdentifier – реализация метода коллокации;
- LeastSquaresIdentifier – реализация метода наименьших квадратов;
- ChebyshevBasis – вычисление базисных функций Чебышева;
- VolterraModelEvaluator – вычисление восстановленного выходного сигнала;
- ResultExporter – сохранение результатов.

Подобная структура позволяет отделить математические операции от интерфейса. Главное окно не должно содержать подробной реализации метода коллокации или метода наименьших квадратов. Его задача – получить параметры от пользователя, передать их в математическое ядро и отобразить результат. В свою очередь, математическое ядро не должно зависеть от элементов формы, кнопок и таблиц.

Для наглядного представления структуры классов построена диаграмма классов, представленная на рисунке 4. На ней показаны основные классы, их поля, методы и связи между ними.

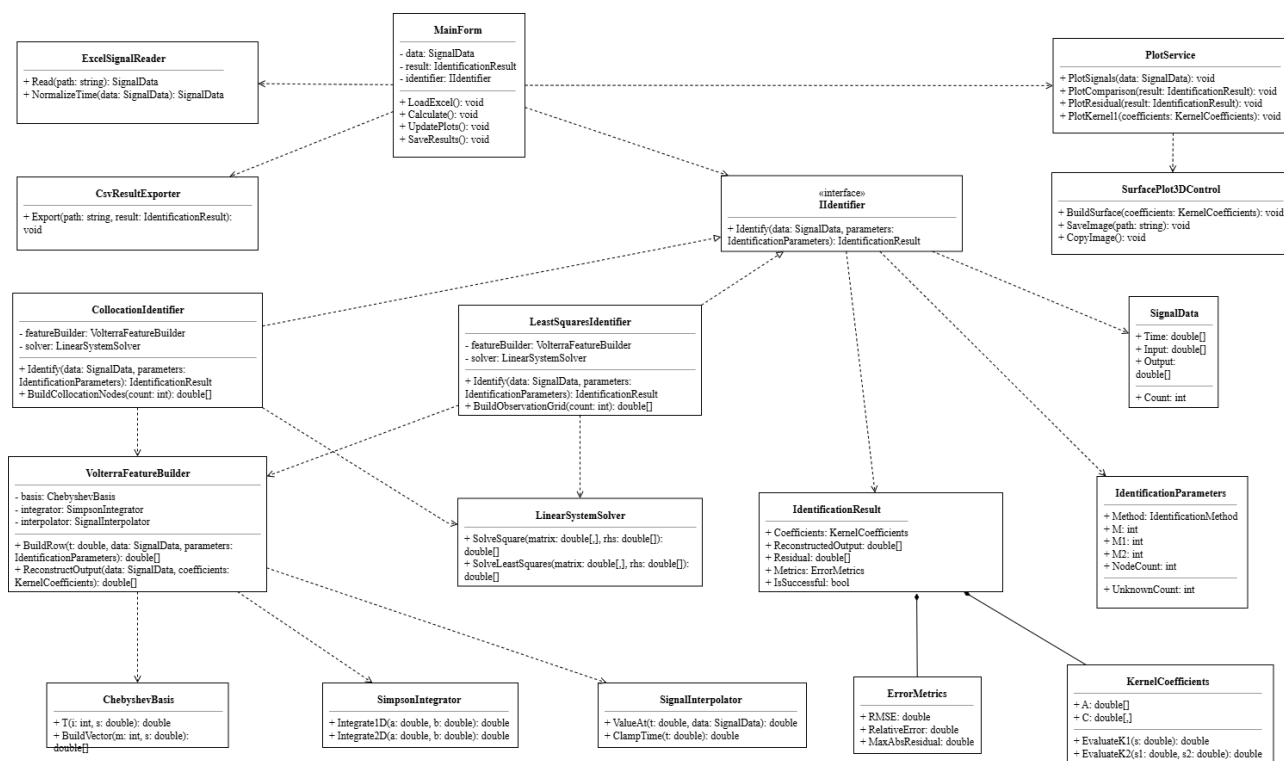


Рисунок 4 – Диаграмма классов программного комплекса

На диаграмме классов особое внимание уделяется интерфейсу `Identifier`, поскольку он задает общий способ подключения различных методов идентификации. Классы `CollocationIdentifier` и `LeastSquaresIdentifier` реализуют этот интерфейс и используют общие вспомогательные классы математического ядра. Классы данных отделены от классов визуализации и ввода-вывода, что уменьшает связанность программы и упрощает дальнейшее сопровождение.

3.10 Проектирование алгоритма работы программного комплекса

Общий алгоритм работы программного комплекса соответствует последовательности действий пользователя и внутренней логике обработки данных. Сначала выполняется загрузка или формирование входных данных. Затем пользователь выбирает метод идентификации и задает параметры модели. После этого программа формирует вычислительную задачу, решает ее выбранным методом и строит результаты.

В общем виде алгоритм работы можно представить следующей последовательностью:

- запуск приложения;
- загрузка входных данных из файла;
- проверка корректности данных;
- отображение таблицы и графиков исходных сигналов;
- выбор метода идентификации;
- ввод параметров модели;
- формирование системы уравнений или задачи минимизации;
- вычисление коэффициентов модели;
- построение восстановленного выходного сигнала;
- вычисление невязки и ошибки;
- отображение графиков и таблиц результатов;
- сохранение результатов при необходимости.

Внутри вычислительного ядра основное отличие между методами проявляется на этапе формирования и решения математической задачи. Для метода коллокации строится система уравнений, соответствующая выполнению модельного равенства в выбранных узлах. Для метода наименьших квадратов формируется переопределенная система, решение которой минимизирует суммарную квадратичную ошибку.

Для наглядного описания последовательности действий построена диаграмма деятельности, приведенная на рисунке 5. Она показывает переходы между основными этапами работы программы, а также ветвление при выборе метода идентификации.

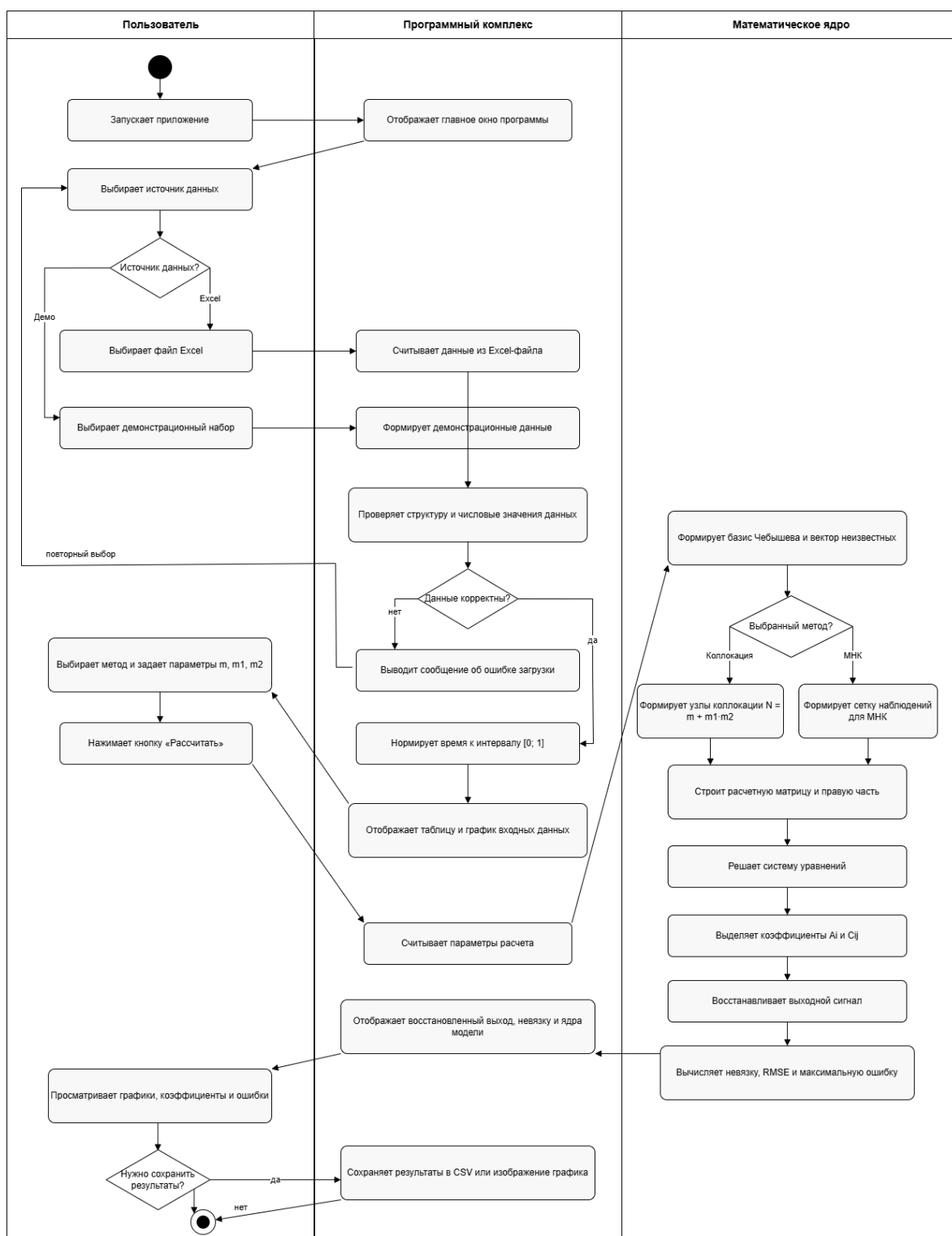


Рисунок 5 – Диаграмма деятельности программного комплекса

Ветвление на диаграмме соответствует выбору метода коллокации или метода наименьших квадратов, после чего результаты обрабатываются единым модулем. Выделение дорожек «Пользователь», «Программный комплекс» и «Математическое ядро» позволяет показать, какие действия выполняются

пользователем, какие – интерфейсной частью программы, а какие относятся к вычислительным процедурам.

3.11 Проектирование обработки ошибок

При работе с программным комплексом возможны ошибки, связанные с некорректными входными данными или параметрами расчета. На этапе проектирования необходимо предусмотреть основные ситуации, которые должны обрабатываться программой.

К таким ситуациям относятся:

- отсутствие выбранного файла;
- неподдерживаемый формат файла;
- отсутствие числовых данных в таблице;
- несовпадение длины входного и выходного сигналов;
- слишком малое количество точек для выбранных параметров модели;
- некорректные значения параметров аппроксимации;
- невозможность решить систему уравнений из-за плохой обусловленности.

При возникновении подобных ошибок программа должна выводить понятное сообщение, не завершая работу аварийно. Для пользователя важно не только получить информацию о факте ошибки, но и понять, какие действия необходимо выполнить: выбрать другой файл, изменить параметры модели, уменьшить порядок аппроксимации или проверить входные данные.

Обработка ошибок должна быть реализована как на уровне интерфейса, так и на уровне вычислительных модулей. Интерфейс проверяет наличие файла и заполненность параметров, а математическое ядро проверяет возможность построения и решения вычислительной задачи.

3.12 Выводы по третьей главе

В третьей главе была выполнена предварительная проработка структуры программного комплекса идентификации параметров динамической системы.

Определено назначение приложения, выделены основные функциональные и нефункциональные требования, описан формат входных и выходных данных, а также обоснован выбор средств разработки.

Предложена модульная архитектура программного комплекса, включающая пользовательский интерфейс, модуль загрузки данных, математическое ядро, модули методов идентификации, визуализацию и экспорт результатов. Такой подход позволяет отделить вычислительную часть от графического интерфейса и упростить дальнейшее развитие программы.

Определены состав UML-диаграмм и структура пользовательского интерфейса. Выполненное проектирование стало основой программной реализации, описанной в четвертой главе.

4 Реализация программного комплекса

После завершения проектирования была выполнена программная реализация комплекса идентификации параметров динамической системы. Основной задачей данного этапа являлось объединение средств загрузки экспериментальных данных, численных методов, процедур восстановления ядер Вольтерра и средств визуализации в одном настольном приложении. В результате разработана программа, позволяющая выполнить полный вычислительный эксперимент без ручного формирования матриц и промежуточных расчетов.

Программный комплекс реализован на языке C# для платформы .NET 8, официальная документация которой определяет средства создания, сборки и автономной публикации приложений [23]. В качестве технологии построения графического интерфейса применена Windows Forms [24]. Чтение электронных таблиц выполняется с помощью библиотеки ClosedXML [25], операции линейной алгебры – с помощью MathNet.Numerics [26], а двумерные графики строятся средствами ScottPlot [27]. Трехмерная поверхность второго ядра отображается специализированным элементом управления, реализованным средствами GDI+ и классов пространства имен System.Drawing [28].

4.1 Структура программной реализации

Исходный код разделен на логические каталоги, соответствующие назначению отдельных частей приложения. Такое разделение уменьшает связанность классов и позволяет изменять вычислительный алгоритм без существенной переработки пользовательского интерфейса. Основные модули программного комплекса представлены в таблице 4.

Таблица 4 – Основные модули программного комплекса

Модуль	Назначение
Core	классы хранения сигналов, параметров и результатов идентификации
IO	загрузка данных из Excel, формирование демонстрационного набора и экспорт результатов
MathModel	вычисление базисных функций, построение матриц, решение систем и восстановление модели
Controls	отображение трехмерной поверхности второго ядра
MainForm	взаимодействие с пользователем, запуск методов и вывод результатов

Класс `SignalPoint` хранит одну тройку значений t , $x(t)$, $y(t)$, а класс `SignalData` представляет упорядоченный набор таких точек. Параметры вычислительного эксперимента передаются через класс `IdentificationParameters`. В нем задаются выбранный метод, количество базисных функций m , m_1 , m_2 , число точек метода наименьших квадратов и количество интервалов численного интегрирования.

Результат расчета хранится в объекте `IdentificationResult`. Он содержит вектор коэффициентов A_i , двумерный массив C_{ij} , значения восстановленного выхода, невязку и показатели точности. Благодаря этому главное окно получает готовый объект результата и не выполняет математические операции самостоятельно.

Оба метода идентификации реализуют общий интерфейс `Identifier`. Интерфейс содержит операцию идентификации, принимающую данные и параметры модели. Классы `CollocationIdentifier` и `LeastSquaresIdentifier` имеют одинаковый способ вызова, поэтому выбор метода в интерфейсе не изменяет общую последовательность обработки.

4.2 Загрузка и предварительная подготовка данных

Для чтения файлов электронных таблиц реализован класс `ExcelSignalReader`. Программа анализирует числовые значения первой рабочей страницы книги и поддерживает два варианта входного файла. Если строка содержит три числовых значения, они интерпретируются как $t, x(t), y(t)$. Если присутствуют только два значения, они интерпретируются как $x(t), y(t)$, а временная координата формируется по номеру строки.

После загрузки выполняется подготовка временной шкалы. Точки сортируются по времени, повторяющиеся отметки объединяются путем усреднения, после чего время преобразуется к интервалу $[0; 1]$:

$$\tilde{t}_i = \frac{t_i - t_{\min}}{t_{\max} - t_{\min}}. \quad (65)$$

где $\tilde{t}_i$ – нормированное время; t_i – исходная отметка времени; $t_{\min}, t_{\max}$ – границы исходного интервала.

Амплитуды входного и выходного сигналов при этом не изменяются. Такое решение соответствует исходной математической постановке, в которой узлы коллокации и аргументы ядер принадлежат интервалу $[0; 1]$.

Для вычисления значений сигналов между исходными отсчетами используется линейная интерполяция. Класс `SignalInterpolator` находит соседние точки временной сетки и вычисляет промежуточное значение по формуле

$$f(t) = f(t_i) + \frac{t - t_i}{t_{i+1} - t_i} (f(t_{i+1}) - f(t_i)). \quad (66)$$

где $f(t)$ обозначает интерполируемый входной или выходной сигнал.

В приложении предусмотрен встроенный демонстрационный набор из 201 точки. Он используется для проверки программы без подготовки внешнего файла. Главное окно после загрузки демонстрационного набора представлено на рисунке 6.

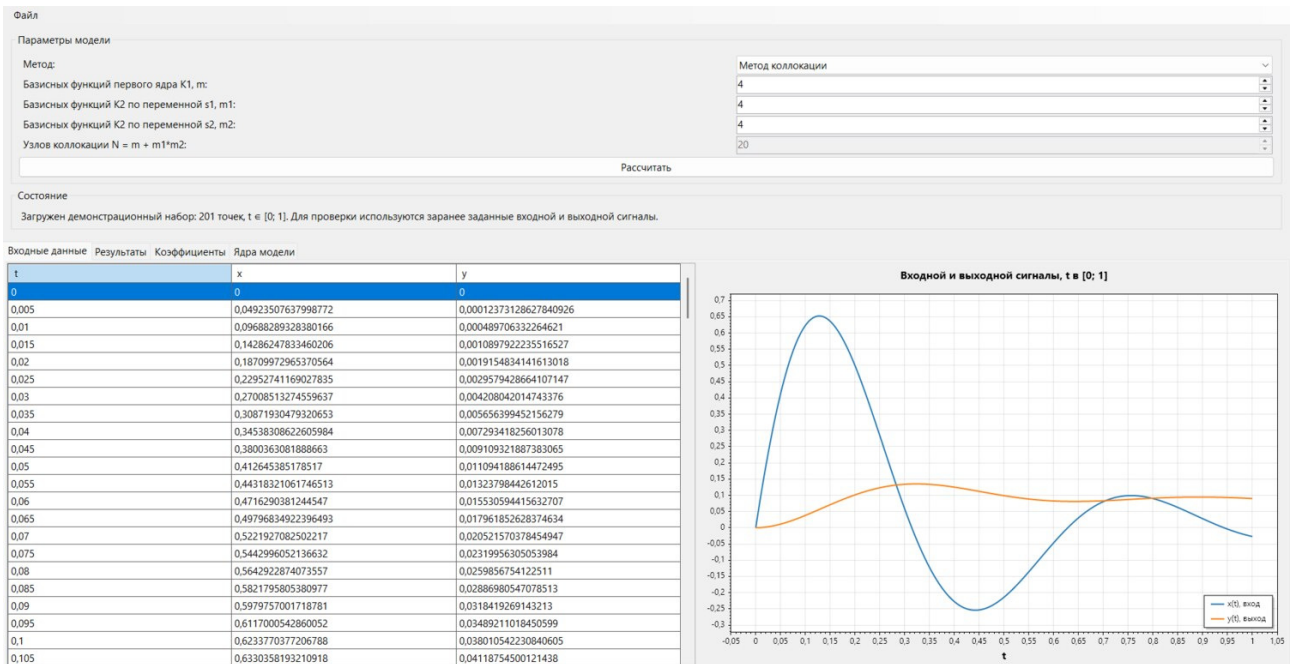


Рисунок 6 – Главное окно программы после загрузки демонстрационных данных

На вкладке «Входные данные» одновременно отображаются таблица значений и графики $x(t)$, $y(t)$. Пользователь может проверить количество точек, временной интервал и общий характер сигналов до запуска идентификации.

4.3 Реализация математического ядра

Вычислительная часть программы включает построение базисных функций, расчет интегральных коэффициентов, формирование матрицы и восстановление выходного сигнала. Многочлены Чебышева первого рода вычисляются в классе `ChebyshevBasis` по рекуррентной формуле, рассмотренной во второй главе. Для каждого значения времени формируется набор интегральных коэффициентов

$$\beta_i(t) = \int_0^t T_i(s)x(t-s) ds. \quad (67)$$

Интеграл в формуле (67) вычисляется составной формулой Симпсона. Интервал $[0; t]$ делится на четное число частей, после чего значения подынтегральной функции суммируются с весами 1, 4, 2, 4, ..., 1. В текущей версии по умолчанию используется 160 интервалов. Выбранное значение обеспечивает приемлемый баланс между точностью и временем расчета.

При разделяемом базисе двойной интегральный коэффициент может быть представлен произведением одномерных коэффициентов:

$$\gamma_{ij}(t) = \beta_i(t)\beta_j(t). \quad (68)$$

Поэтому строка расчетной матрицы формируется последовательно из значений

$$\mathbf{q}(t) = (\beta_0(t), \dots, \beta_{m-1}(t), \gamma_{00}(t), \dots, \gamma_{m_1-1, m_2-1}(t)). \quad (69)$$

Порядок элементов вектора (69) согласован с порядком неизвестных коэффициентов. Сначала в решении располагаются m коэффициентов первого ядра, затем построчно записываются $m_1 m_2$ коэффициентов массива второго ядра. После решения системы класс ResultFactory разделяет общий вектор на массивы A_i и C_{ij} .

Восстановление выходного сигнала выполняет класс VolterraModelEvaluator. Для каждой точки исходной временной сетки повторно формируется вектор признаков, после чего определяется скалярное произведение с найденным вектором коэффициентов. Невязка и показатели качества вычисляются по формулам, приведенным в подразделе 2.8. Ключевые фрагменты исходного кода, связанные с построением базиса, формированием расчетной матрицы, решением системы и восстановлением выхода, приведены в приложении Г.

4.4 Программная реализация методов идентификации

4.4.1 Реализация метода коллокации

Класс CollocationIdentifier сначала приводит временную ось к интервалу $[0; 1]$. Число неизвестных коэффициентов определяется автоматически:

$$N = m + m_1 m_2. \quad (70)$$

Затем формируются узлы

$$t_k = \frac{k}{N}, \quad k = 1, 2, \dots, N. \quad (71)$$

Для каждого узла вычисляется строка матрицы и значение выходного сигнала. В результате получается квадратная система из N уравнений относительно N коэффициентов.

Перед решением столбцы матрицы масштабируются по евклидовой норме. Это уменьшает влияние различий в величинах линейных и квадратичных признаков. Для решения используется сингулярное разложение. Малые сингулярные числа, не превышающие заданный порог, отсекаются, после чего строится псевдообратное решение. Такой подход предотвращает появление неопределенных числовых значений при плохо обусловленной матрице. После решения выполняется обратное масштабирование коэффициентов.

4.4.2 Реализация метода наименьших квадратов

В классе `LeastSquaresIdentifier` используется число точек, превышающее количество неизвестных. Для демонстрационного набора в расчет включаются все 201 точка. Матрица имеет M строк и N столбцов, где $M > N$.

После масштабирования столбцов формируется система регуляризованных нормальных уравнений:

$$(Q^T Q + \lambda I) \mathbf{a} = Q^T \mathbf{y}. \quad (72)$$

где Q – расчетная матрица; $\mathbf{a}$ – вектор коэффициентов; $\mathbf{y}$ – вектор выходных значений; λ – малый параметр регуляризации; I – единичная матрица.

Параметр регуляризации выбирается относительно среднего диагонального элемента матрицы $Q^T Q$. Основным способом решения является разложение Холецкого. Если оно неприменимо, программа последовательно использует QR-разложение и сингулярное разложение. Наличие нескольких вариантов решения повышает устойчивость программы к особенностям входных данных.

4.5 Реализация пользовательского интерфейса

Главная форма содержит меню «Файл», панель параметров, область состояния и четыре вкладки. Через меню выполняется загрузка Excel-файла, формирование демонстрационного набора, сохранение результатов в CSV и завершение работы.

В верхней части формы пользователь выбирает метод и задает значения m , m_1 , m_2 . При выборе метода коллокации число узлов вычисляется автоматически по формуле (70). Для метода наименьших квадратов поле используется для задания числа точек, однако при загрузке данных по умолчанию устанавливается весь доступный набор.

Вкладка «Результаты» содержит график сравнения $y(t)$, $\hat{y}(t)$ и отдельный график невязки. Разные стили линий позволяют различать сигналы даже при почти полном совпадении. Для двумерных графиков доступны масштабирование и сохранение изображения.

На вкладке «Коэффициенты» значения A_i представлены вместе с обозначениями соответствующих базисных функций. Коэффициенты C_{ij} выводятся в виде матрицы, строки которой соответствуют индексу i , а столбцы – индексу j . Такое представление непосредственно соответствует двойной сумме аппроксимации второго ядра.

Вкладка «Ядра модели» содержит график $K_1(s)$ и трехмерную поверхность $K_2(s_1, s_2)$. Специализированный элемент `SurfacePlot3DControl` позволяет вращать поверхность левой кнопкой мыши и изменять масштаб колесом. По правой кнопке мыши доступны сохранение текущего ракурса в PNG, JPEG или BMP и копирование изображения в буфер обмена.

4.6 Порядок работы с программой

Работа с программным комплексом выполняется в следующей последовательности:

- запустить исполняемый файл программы;
- выбрать команду загрузки Excel-файла или демонстрационного набора;
- проверить таблицу и графики входных данных;
- выбрать метод идентификации;
- задать значения m , m_1 , m_2 ;
- нажать кнопку «Рассчитать»;
- проанализировать восстановленный выход, невязку, коэффициенты и ядра;
- при необходимости сохранить таблицу результатов или изображения графиков.

Ошибки загрузки и расчета перехватываются в пользовательском интерфейсе. При недостаточном числе точек, отсутствии числовых данных или некорректных параметрах выводится сообщение с описанием причины и рекомендуемым действием.

4.7 Подготовка программы к эксплуатации

Для переноса приложения на другой компьютер была выполнена публикация для 64-разрядной версии Windows. Использован режим автономной публикации, при котором необходимые компоненты среды .NET включаются в публикуемую папку. Поэтому запуск программы на другом компьютере не требует отдельной установки .NET 8.

Результатом публикации является исполняемый файл и набор служебных библиотек. Проверка опубликованной версии показала, что загрузка демонстрационных данных, выполнение обоих методов, построение графиков и сохранение результатов работают так же, как при запуске из среды Visual Studio. Создание отдельного установщика в рамках работы не выполнялось, поскольку переносимой опубликованной версии достаточно для демонстрации и эксплуатации программного комплекса.

4.8 Выводы по четвертой главе

В четвертой главе описана программная реализация комплекса идентификации параметров динамической системы. Рассмотрены структура исходного кода, классы хранения данных и результатов, загрузка электронных таблиц, нормирование временной шкалы, интерполяция сигналов и численное вычисление интегральных коэффициентов.

Реализованы методы коллокации и наименьших квадратов, использующие общий интерфейс и единый формат результата. Показаны способы формирования расчетной матрицы, решения линейных систем, восстановления выходного сигнала и построения ядер первого и второго порядков.

Разработанный пользовательский интерфейс объединяет ввод параметров, запуск вычислений, просмотр коэффициентов, двумерных графиков и трехмерной поверхности. Предусмотрены сохранение результатов, обработка ошибочных ситуаций и автономная публикация приложения для Windows. Проверка реализованных функций и анализ численных результатов выполняются в пятой главе.

5 Тестирование и анализ результатов работы программного комплекса

После завершения программной реализации была выполнена проверка соответствия комплекса сформулированным требованиям и оценена точность численных методов. Цель тестирования состояла не только в подтверждении корректной работы элементов интерфейса, но и в проверке полного вычислительного цикла: от загрузки сигналов до построения коэффициентов, ядер и восстановленного выхода.

Качество программного комплекса оценивалось по функциональной пригодности, надежности обработки входных данных, корректности визуализации и точности получаемого результата. Такой набор характеристик согласуется с моделью качества программного продукта, выделяющей функциональную пригодность, надежность, удобство использования и переносимость [22].

5.1 Методика и условия тестирования

Испытания выполнялись на финальной версии приложения, опубликованной для 64-разрядной операционной системы Windows. Проверка включала три группы тестов:

- функциональные тесты основных пользовательских операций;
- негативные тесты загрузки и обработки некорректных данных;
- вычислительный эксперимент на наборе с заранее известным способом формирования выходного сигнала.

Функциональные тесты проводились по сценарию «действие – ожидаемый результат – фактический результат». Для численной проверки использовались одинаковые параметры аппроксимации обоих методов. Точность оценивалась по среднеквадратичной, относительной и максимальной абсолютной ошибкам. Дополнительно анализировались форма невязки, отсутствие неопределенных

числовых значений и возможность построения всех предусмотренных графиков.

В основной части приведены сводные результаты, непосредственно характеризующие работоспособность комплекса. Подробный протокол испытаний и дополнительные изображения вынесены в приложения Б и В, что позволяет не перегружать текст вспомогательным иллюстративным материалом.

5.2 Функциональное тестирование

Функциональное тестирование выполнялось после завершения основных этапов реализации. Состав проверок формировался с учетом общих требований к программе и методике испытаний программных средств [29]. Проверялись загрузка данных, работа обоих методов, отображение результатов, сохранение файлов и обработка ошибочных ситуаций. Сводные результаты представлены в таблице 5. Подробный протокол приведен в приложении В.

Таблица 5 – Результаты функционального тестирования

Проверяемая функция	Ожидаемый результат	Результат
Загрузка файла x, y	формирование времени и отображение сигналов	соответствует
Загрузка файла t, x, y	чтение времени и приведение его к $[0; 1]$	соответствует
Демонстрационный набор	отображение 201 точки и исходных графиков	соответствует
Метод коллокации	получение конечных коэффициентов и графиков	соответствует
Метод наименьших квадратов	решение переопределенной системы	соответствует
Сохранение CSV	создание файла с сигналами, ошибками и коэффициентами	соответствует

Продолжение таблицы 5

Проверяемая функция	Ожидаемый результат	Результат
Сохранение трехмерного графика	создание изображения текущего ракурса	соответствует
Некорректный файл	вывод сообщения без аварийного завершения	соответствует

Результаты функционального тестирования подтверждают, что реализованные операции выполняются в соответствии с требованиями, сформулированными в третьей главе.

5.3 Проведение вычислительного эксперимента

Для численной проверки использован демонстрационный пример на интервале $t \in [0; 1]$. Входной сигнал и исходные ядра задавались функциями

$$x(t) = e^{-3t} \sin(10t), \quad K_1(s) = \cos\left(\frac{s}{2}\right), \quad K_2(s_1, s_2) = \sin(s_1 + 2s_2). \quad (73)$$

Выходной сигнал был сформирован численным вычислением интегральной модели второго порядка. Объем набора составил 201 точку. Для обоих методов использовались одинаковые параметры:

$$m = 4, \quad m_1 = 4, \quad m_2 = 4. \quad (74)$$

Таким образом, количество неизвестных коэффициентов было равно

$$N = 4 + 4 \cdot 4 = 20. \quad (75)$$

5.3.1 Результаты метода коллокации

Для метода коллокации сформировано 20 узлов и 20 уравнений. Сравнение исходного и восстановленного выходов, а также распределение невязки представлены на рисунке 7.

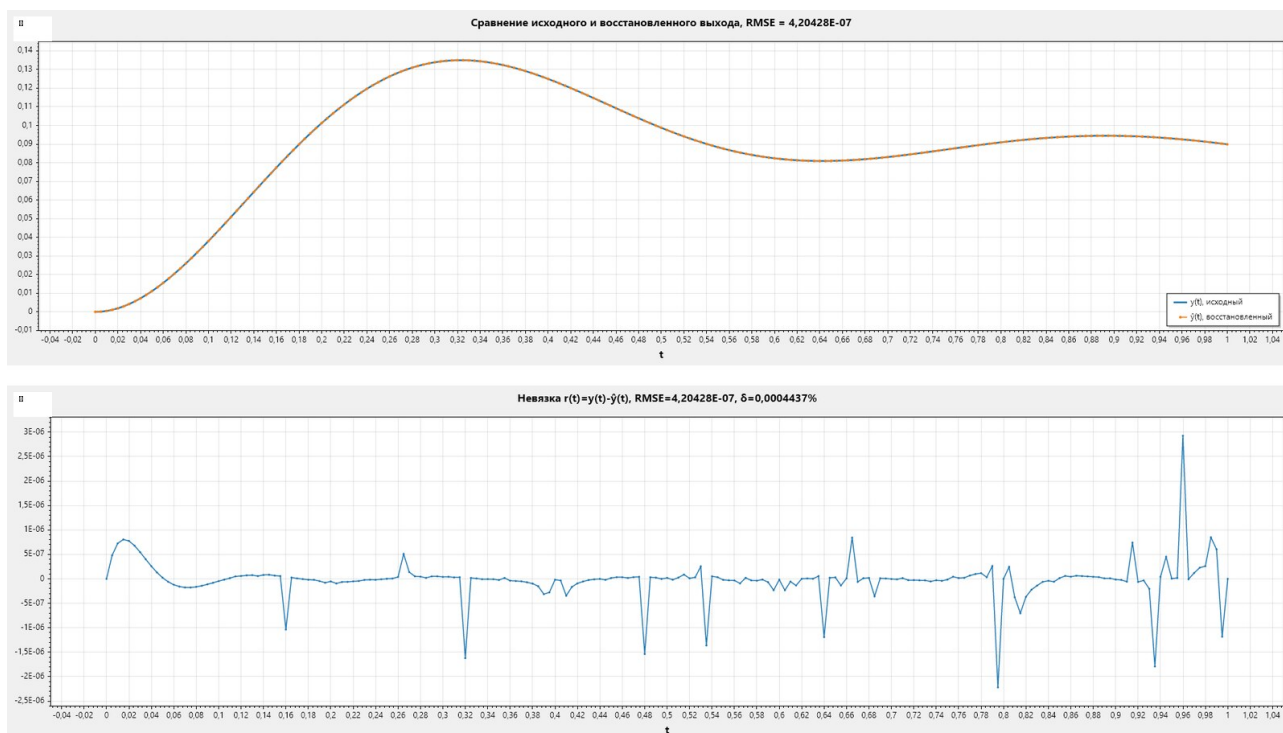


Рисунок 7 – Результаты идентификации методом коллокации

Получено значение среднеквадратичной ошибки $RMSE = 4,20428 \cdot 10^{-7}$, относительная ошибка составила 0,0004437 процента, а максимальная абсолютная ошибка – $2,92652 \cdot 10^{-6}$. Кривые исходного и восстановленного выходов практически совпадают. На графике невязки заметны локальные колебания, однако их абсолютная величина не превышает нескольких миллионных долей.

5.3.2 Результаты метода наименьших квадратов

При использовании метода наименьших квадратов в расчет включены все 201 значение выходного сигнала. Сравнение выходов и невязка представлены на рисунке 8.

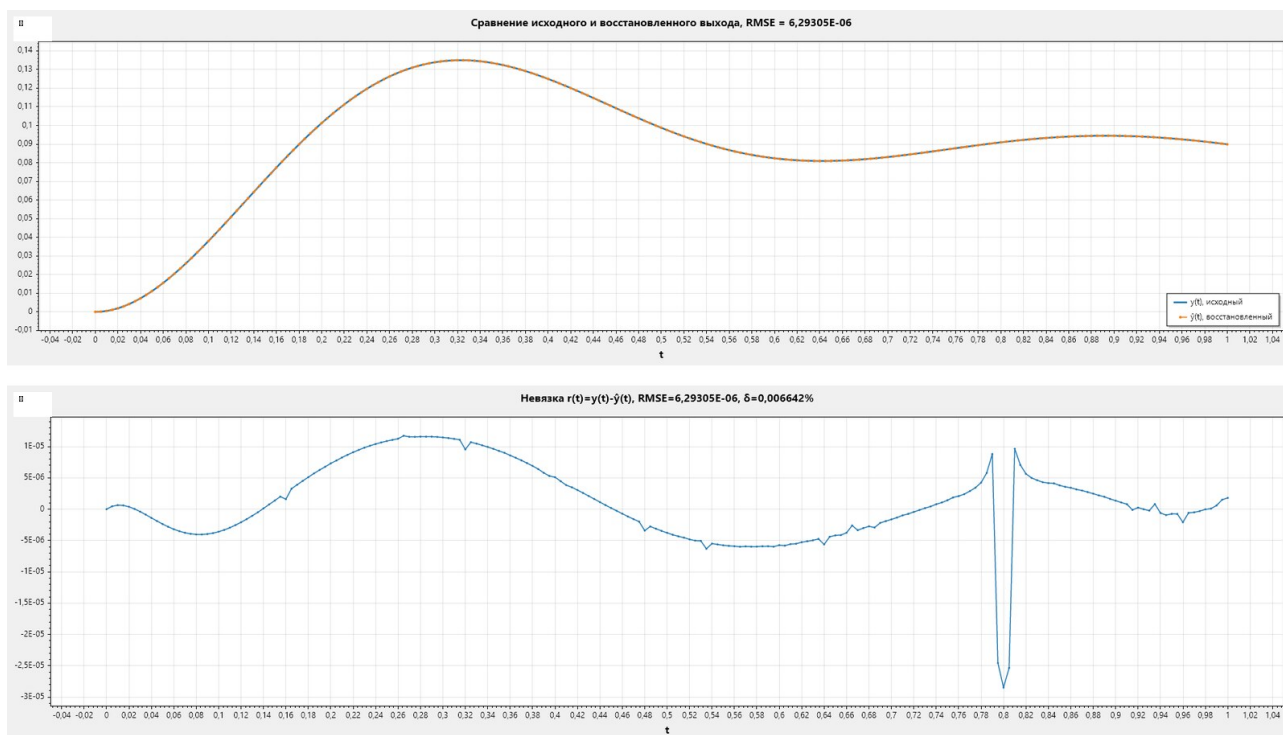
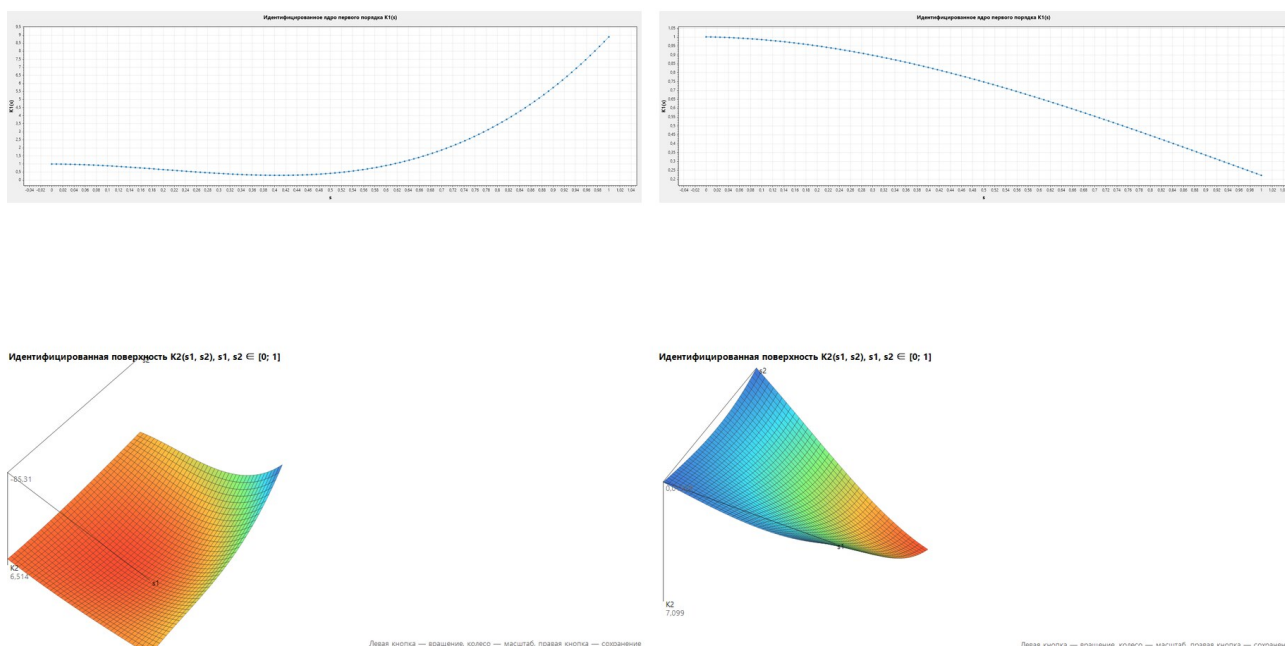


Рисунок 8 – Результаты идентификации методом наименьших квадратов

Для метода наименьших квадратов получено $RMSE = 6,29305 \cdot 10^{-6}$. Относительная ошибка равна 0,006642 процента, максимальная абсолютная ошибка – $2,84470 \cdot 10^{-5}$. Ошибка остается малой по сравнению с диапазоном выходного сигнала, поэтому две кривые на графике также визуальнo совпадают.

Полученные приближенные ядра двух методов представлены на рисунке 9. Полные таблицы коэффициентов приведены в приложении Б.



а) Метод коллокации

б) Метод наименьших квадратов

Рисунок 9 – Идентифицированные ядра модели

Оба метода формируют конечные коэффициенты и непрерывные приближения ядер. При этом численные значения коэффициентов и форма поверхностей различаются, поскольку методы используют разные критерии построения решения: выполнение модельного равенства в узлах коллокации и минимизацию суммарной квадратичной ошибки на расширенном наборе точек. Основным критерием качества вычислительного эксперимента являлась точность воспроизведения выходного сигнала.

5.3.3 Сравнение методов

Для сопоставления результатов были использованы одинаковые входные данные и одинаковая структура аппроксимации ядер. Сравнение выполнялось по числу неизвестных, числу расчетных уравнений, среднеквадратичной ошибке, относительной ошибке и максимальному абсолютному отклонению.

Такое сопоставление показывает, насколько различается поведение методов при одинаковой размерности модели. Отдельно учитывались не только значения ошибок, но и количество уравнений, используемых при построении решения.

Сводные показатели вычислительного эксперимента приведены в таблице 6.

Таблица 6 – Сравнение результатов методов идентификации

Показатель	Коллокация	МНК
Количество неизвестных	20	20
Количество уравнений	20	201
RMSE	$4,20428 \cdot 10^{-7}$	$6,29305 \cdot 10^{-6}$
Относительная ошибка, процент	0,0004437	0,006642
Максимальная ошибка	$2,92652 \cdot 10^{-6}$	$2,84470 \cdot 10^{-5}$

На рассматриваемом наборе метод коллокации показал меньшую ошибку. Это объясняется тем, что демонстрационные данные сформированы без случайного шума, а число узлов точно соответствует числу неизвестных. Метод наименьших квадратов также обеспечивает высокую точность и использует все доступные точки, что является преимуществом при работе с экспериментальными данными, содержащими погрешности.

5.4 Анализ достоверности и ограничений результатов

Полученные малые значения ошибки подтверждают правильность согласованной работы программных модулей на демонстрационном наборе. При этом результат вычислительного эксперимента следует интерпретировать с учетом условий его проведения. Исходный выход был сформирован той же интегральной моделью второго порядка, которая используется при идентификации, а в данные не добавлялся случайный шум. Поэтому данный тест в первую очередь проверяет программную реализацию формул, порядок коэффициентов и корректность восстановления выхода.

Высокая точность на одном наборе не означает одинаковой точности для произвольных экспериментальных сигналов. На результат влияют число и расположение точек, уровень шума, выбор количества базисных функций, свойства входного воздействия и обусловленность расчетной матрицы. В задачах

идентификации параметры модели должны выбираться с учетом объема и качества исходных данных [1, 2].

Используемая модель не содержит свободного члена нулевого порядка. Поэтому при значительном постоянном смещении выходного сигнала значение модели в начальный момент может отличаться от экспериментального. Кроме того, усеченный ряд второго порядка ориентирован на системы, для которых вклад членов более высоких порядков не является определяющим. Эти ограничения должны учитываться при подготовке новых контрольных наборов.

Для дальнейшей верификации целесообразно сформировать несколько Excel-файлов с различными входными воздействиями. Один набор может повторять демонстрационные данные, второй – содержать другой гладкий входной сигнал, третий – включать контролируемый аддитивный шум. Такие испытания позволят исследовать устойчивость методов и обоснованно выбрать рекомендуемые значения параметров для разных условий применения.

5.5 Выводы по пятой главе

В пятой главе разработана методика тестирования программного комплекса, выполнены функциональные и негативные проверки, а также проведен вычислительный эксперимент для методов коллокации и наименьших квадратов.

По результатам функционального тестирования подтверждена корректность загрузки данных, задания параметров, выполнения идентификации, визуализации коэффициентов и ядер, а также сохранения результатов. Проверка ошибочных ситуаций показала, что приложение выводит диагностические сообщения и не завершает работу аварийно.

На демонстрационном наборе оба метода обеспечили высокую точность восстановления выходного сигнала. При выбранных параметрах метод коллокации показал меньшее значение среднеквадратичной ошибки, а метод наименьших квадратов использовал расширенный набор наблюдений. Проведенный анализ определил границы применимости полученных результатов

и направления последующего тестирования на дополнительных наборах данных.

Заключение

В ходе выполнения выпускной квалификационной работы разработан программный комплекс идентификации параметров динамической системы, рассматриваемой в виде модели «черного ящика» с одним входом и одним выходом. В качестве математической основы использована усеченная интегрально-степенная модель Вольтерра второго порядка, включающая ядра $K_1(s)$ и $K_2(s_1, s_2)$.

В теоретической части рассмотрены свойства динамических систем с памятью, модель «черного ящика», ряды Вольтерра и постановка задачи восстановления ядер по известным входному и выходному сигналам. Для перехода к конечномерной задаче ядра представлены разложениями по многочленам Чебышева. Рассмотрены метод коллокации и метод наименьших квадратов, а также критерии оценки точности восстановленного выходного сигнала.

В ходе проектирования были определены функциональные и нефункциональные требования, формат входных и выходных данных, архитектура приложения и состав основных классов. Программный комплекс реализован на языке C# для платформы .NET 8 с использованием Windows Forms. Реализованы загрузка Excel-файлов, формирование демонстрационных данных, численное вычисление интегральных коэффициентов, решение расчетных систем, вывод коэффициентов, построение двумерных графиков и интерактивной трехмерной поверхности второго ядра.

В программе реализованы два метода идентификации. Для метода коллокации количество узлов автоматически определяется как $N = m + m_1 m_2$. Метод наименьших квадратов использует расширенный набор точек и регуляризованное решение нормальных уравнений. Для повышения численной устойчивости применяются масштабирование столбцов матрицы и методы

разложения матриц.

В пятой главе функциональное тестирование подтвердило корректную работу загрузки данных, расчетных модулей, визуализации и экспорта. При параметрах $m = 4$, $m_1 = 4$, $m_2 = 4$ на демонстрационном наборе метод коллокации обеспечил значение $RMS E = 4,20428 \cdot 10^{-7}$, а метод наименьших квадратов – $RMS E = 6,29305 \cdot 10^{-6}$. В обоих случаях восстановленный выходной сигнал практически совпал с исходным.

Таким образом, поставленная цель достигнута.

Разработанный программный комплекс автоматизирует вычислительный эксперимент по идентификации параметров модели Вольтерра и предоставляет средства анализа численных результатов. Дальнейшее развитие программы может быть связано с добавлением новых базисных систем, методов регуляризации, пакетной обработкой нескольких наборов данных и расширением средств сравнения вычислительных экспериментов.

Список использованных источников

1. Ljung, L. System Identification: Theory for the User / L. Ljung. – 2nd ed. – Upper Saddle River : Prentice Hall PTR, 1999. – 609 p.
2. Billings, S. A. Nonlinear System Identification: NARMAX Methods in the Time, Frequency, and Spatio-Temporal Domains / S. A. Billings. – Chichester : Wiley, 2013. – 576 p.
3. Carassale, L. Modeling Nonlinear Systems by Volterra Series / L. Carassale, A. Kareem // Journal of Engineering Mechanics. – 2010. – Vol. 136, No. 6. – P. 801–818.
4. Zhang, H. Volterra Series: Introduction and Application / H. Zhang. – ECEN 665: RF Communication Circuits and Systems, 2008.
5. Volterra, V. Theory of Functionals and of Integral and Integro-Differential Equations / V. Volterra. – Mineola : Dover Publications, 2005. – 288 p.
6. Boyd, S. Fading Memory and the Problem of Approximating Nonlinear Operators with Volterra Series / S. Boyd, L. O. Chua // IEEE Transactions on Circuits and Systems. – 1985. – Vol. 32, No. 11. – P. 1150–1161.
7. Rugh, W. J. Nonlinear System Theory: The Volterra/Wiener Approach / W. J. Rugh. – Baltimore : Johns Hopkins University Press, 1981. – 330 p.
8. Schetzen, M. The Volterra and Wiener Theories of Nonlinear Systems / M. Schetzen. – New York : Wiley, 1980. – 618 p.
9. Mason, J. C. Chebyshev Polynomials / J. C. Mason, D. C. Handscomb. – Boca Raton : Chapman and Hall/CRC, 2003. – 360 p.
10. Trefethen, L. N. Approximation Theory and Approximation Practice / L. N. Trefethen. – Philadelphia : SIAM, 2013. – 305 p.
11. Верлань, А. Ф. Интегральные уравнения: методы, алгоритмы, программы / А. Ф. Верлань, В. С. Сизиков. – Киев : Наукова думка, 1986. – 544 с.

12. Golub, G. H. Matrix Computations / G. H. Golub, C. F. Van Loan. – 4th ed. – Baltimore : Johns Hopkins University Press, 2013. – 756 p.
13. Brunner, H. Collocation Methods for Volterra Integral and Related Functional Differential Equations / H. Brunner. – Cambridge : Cambridge University Press, 2004. – 597 p.
14. Diogo, T. Collocation Solutions of a Weakly Singular Volterra Integral Equation / T. Diogo, P. Lima // TEMA Tendencias em Matematica Aplicada e Computacional. – 2007. – Vol. 8, No. 2. – P. 229–238.
15. Brunner, H. The Piecewise Polynomial Collocation Method for Nonlinear Weakly Singular Volterra Equation / H. Brunner, A. Pedaş, G. Vainikko // Mathematics of Computation. – 1999. – Vol. 68, No. 227. – P. 1079–1095.
16. Tynda, A. N. Mixed Spline-Collocation Method for Approximate Solution of Weakly Singular Volterra Integral Equations / A. N. Tynda // Proceedings of Middle-Volga Mathematical Society. – 2005. – Vol. 7, No. 1. – P. 351–358.
17. Lawson, C. L. Solving Least Squares Problems / C. L. Lawson, R. J. Hanson. – Philadelphia : SIAM, 1995. – 337 p.
18. Hansen, P. C. Discrete Inverse Problems: Insight and Algorithms / P. C. Hansen. – Philadelphia : SIAM, 2010. – 213 p.
19. Sommerville, I. Software Engineering / I. Sommerville. – 10th ed. – Boston : Pearson, 2016. – 816 p.
20. Booch, G. The Unified Modeling Language User Guide / G. Booch, J. Rumbaugh, I. Jacobson. – 2nd ed. – Addison-Wesley, 2005. – 496 p.
21. ГОСТ 19.201–78. Единая система программной документации. Техническое задание. Требования к содержанию и оформлению. – Введ. 1980–01–01.
22. ISO/IEC 25010:2023. Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Product quality model. – Geneva : International Organization for Standardization, 2023.
23. Microsoft. .NET documentation [Электронный ресурс]. – URL:

<https://learn.microsoft.com/dotnet/> (дата обращения: 19.06.2026).

24. Microsoft. Windows Forms overview [Электронный ресурс]. – URL: <https://learn.microsoft.com/dotnet/desktop/winforms/overview/> (дата обращения: 19.06.2026).

25. ClosedXML. Documentation [Электронный ресурс]. – URL: <https://docs.closedxml.io/> (дата обращения: 19.06.2026).

26. Math.NET Numerics. Documentation [Электронный ресурс]. – URL: <https://numerics.mathdotnet.com/> (дата обращения: 19.06.2026).

27. ScottPlot. Documentation [Электронный ресурс]. – URL: <https://scottplot.net/> (дата обращения: 19.06.2026).

28. Microsoft. System.Drawing namespace [Электронный ресурс]. – URL: <https://learn.microsoft.com/dotnet/api/system.drawing/> (дата обращения: 19.06.2026).

29. ГОСТ 19.301–79. Единая система программной документации. Программа и методика испытаний. Требования к содержанию и оформлению. – Введ. 1981–01–01.

Приложение А
(рекомендуемое)
Спецификации вариантов использования

Таблица А.1 – Спецификация варианта использования «Загрузить входные данные»

Поле	Содержание
Наименование	Загрузить входные данные
Действующее лицо	пользователь
Цель	загрузить значения входного и выходного сигналов из файла Excel
Предусловия	приложение запущено; файл содержит не менее пяти числовых строк
Основной сценарий	пользователь выбирает команду «Открыть Excel», указывает файл, программа считывает данные, приводит время к интервалу $[0; 1]$, отображает таблицу и графики
Альтернативный сценарий	при отсутствии числовых данных или неподдерживаемой структуре выводится сообщение об ошибке
Постусловия	данные доступны для выбора параметров и выполнения идентификации

Таблица А.2 – Спецификация варианта использования «Задать параметры модели»

Поле	Содержание
Наименование	Задать параметры модели
Действующее лицо	пользователь
Цель	определить количество базисных функций первого и второго ядер
Предусловия	приложение запущено
Основной сценарий	пользователь вводит m, m_1, m_2 ; программа проверяет положительность значений и вычисляет число неизвестных
Альтернативный сценарий	при слишком большом числе коэффициентов выводится рекомендация уменьшить параметры
Постусловия	параметры сохранены в элементах формы и используются при расчете

Таблица А.3 – Спецификация варианта использования «Выбрать метод идентификации»

Поле	Содержание
Наименование	Выбрать метод идентификации

Продолжение таблицы А.3

Поле	Содержание
Действующее лицо	пользователь
Цель	выбрать метод коллокации или метод наименьших квадратов
Предусловия	приложение запущено
Основной сценарий	пользователь выбирает метод в раскрывающемся списке; форма изменяет назначение поля числа точек
Постусловия	при запуске расчета создается соответствующая реализация интерфейса Pidentifier

Таблица А.4 – Спецификация варианта использования «Выполнить идентификацию»

Поле	Содержание
Наименование	Выполнить идентификацию
Действующее лицо	пользователь
Цель	определить коэффициенты ядер и восстановить выходной сигнал
Предусловия	данные загружены; параметры модели заданы корректно
Основной сценарий	пользователь нажимает «Рассчитать»; программа формирует матрицу, решает задачу выбранным методом, вычисляет выход, невязку и ошибки
Альтернативный сценарий	при недостатке точек или невозможности решения выводится сообщение с рекомендацией изменить параметры
Постусловия	таблицы коэффициентов и графики обновлены

Таблица А.5 – Спецификация варианта использования «Просмотреть результаты»

Поле	Содержание
Наименование	Просмотреть результаты
Действующее лицо	пользователь
Цель	проанализировать восстановленный выход, невязку, коэффициенты и ядра
Предусловия	расчет завершен без ошибки

Продолжение таблицы А.5

Поле	Содержание
Основной сценарий	пользователь переключается между вкладками «Результаты», «Коэффициенты» и «Ядра модели», изменяет масштаб и ракурс графиков
Постусловия	результаты доступны для визуального анализа и сохранения

Таблица А.6 – Спецификация варианта использования «Сохранить результаты»

Поле	Содержание
Наименование	Сохранить результаты
Действующее лицо	пользователь
Цель	сохранить численные данные или изображение графика
Предусловия	расчет завершен
Основной сценарий	пользователь выбирает сохранение CSV либо вызывает контекстное меню графика и указывает имя файла
Альтернативный сценарий	при ошибке доступа к файлу выводится сообщение
Постусловия	в выбранной папке создан файл результатов или изображения

Приложение Б
(рекомендуемое)

Дополнительные результаты вычислительного эксперимента

Входные данные Результаты Коэффициенты Ядра модели

Коэффициенты первого ядра K1(s)
K1(s) = ∑ A_iT_i(s). В таблице указаны базисная функция, обозначение и значение A_i.

Базисная функция	Коэффициент	Значение
T ₀ (s)	A ₀	-5,14895655208
T ₁ (s)	A ₁	15,1230064897
T ₂ (s)	A ₂	-6,15039647367
T ₃ (s)	A ₃	5,06508480923

Матрица коэффициентов второго ядра K2(s1, s2)
K2(s1, s2) = ∑ ∑ C_{ij}T_i(s1)T_j(s2). Строки соответствуют i, столбцы — j.

i \ j	j = 0	j = 1	j = 2	j = 3
i = 0	35,95176975	-45,35630806	44,61178649	-5,711694663
i = 1	-45,35630805	-16,11731743	-51,19665404	-16,55713592
i = 2	44,61178649	-51,19665403	53,72882013	-6,880411065
i = 3	-5,711694663	-16,55713592	-6,880411065	3,30873719

Рисунок Б.1 – Коэффициенты ядер, полученные методом коллокации

Коэффициенты первого ядра K1(s)
K1(s) = ∑ A_iT_i(s). В таблице указаны базисная функция, обозначение и значение A_i.

Базисная функция	Коэффициент	Значение
T ₀ (s)	A ₀	0,446493496587
T ₁ (s)	A ₁	0,236840845478
T ₂ (s)	A ₂	-0,555034483786
T ₃ (s)	A ₃	0,0942027207532

Матрица коэффициентов второго ядра K2(s1, s2)
K2(s1, s2) = ∑ ∑ C_{ij}T_i(s1)T_j(s2). Строки соответствуют i, столбцы — j.

i \ j	j = 0	j = 1	j = 2	j = 3
i = 0	0,4107659268	1,736611477	0,2426839428	-0,8994182024
i = 1	1,736611484	1,649143401	1,520740796	0,1095697743
i = 2	0,2426839391	1,520740671	0,1207514538	-0,4126261
i = 3	-0,8994186899	0,1095696599	-0,4126262579	0,3230158275

Рисунок Б.2 – Коэффициенты ядер, полученные методом наименьших квадратов

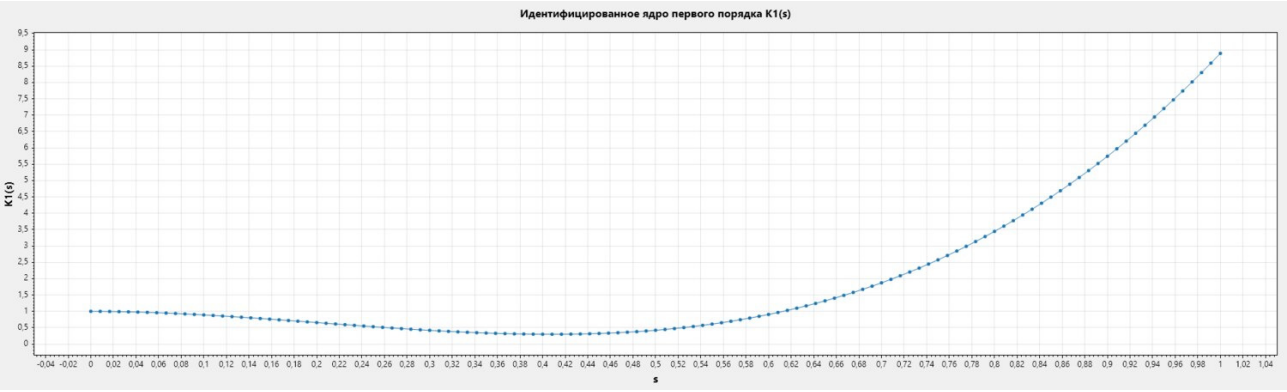
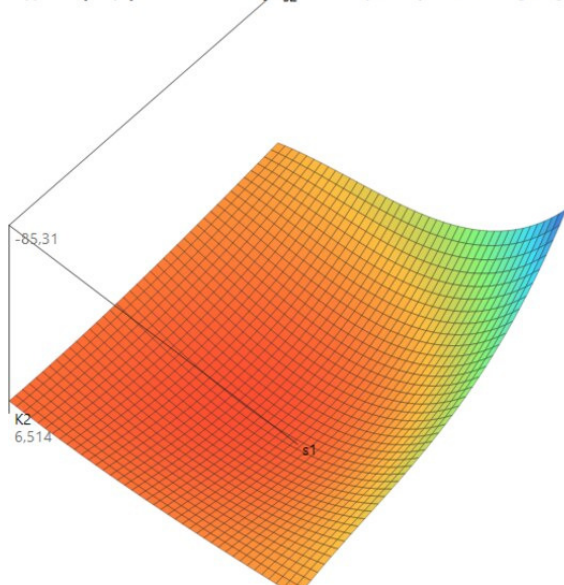


Рисунок Б.3 – Ядро первого порядка, полученное методом коллокации

Идентифицированная поверхность $K_2(s_1, s_2)$, $s_1, s_2 \in [0; 1]$



Левая кнопка — вращение, колесо — масштаб, правая кнопка — сохранение

Рисунок Б.4 – Поверхность второго ядра, полученная методом коллокации

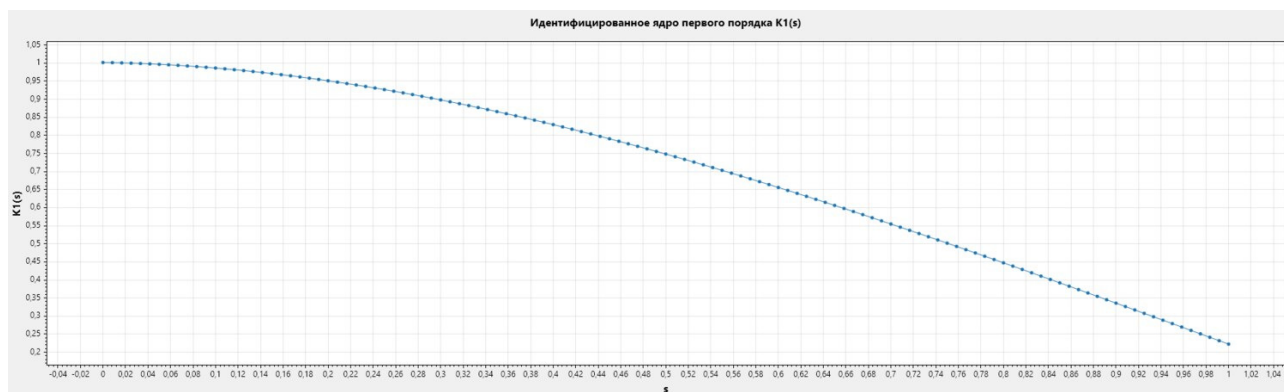
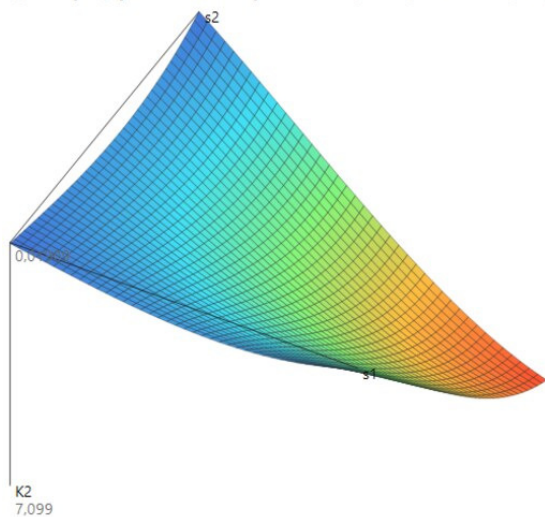


Рисунок Б.5 – Ядро первого порядка, полученное методом наименьших квадратов

Идентифицированная поверхность $K2(s1, s2)$, $s1, s2 \in [0; 1]$



Левая кнопка — вращение, колесо — масштаб, правая кнопка — сохранение

Рисунок Б.6 – Поверхность второго ядра, полученная методом наименьших квадратов

Приложение В
(рекомендуемое)
Протокол функционального тестирования

Таблица В.1 – Протокол функционального тестирования

Проверка	Действия	Полученный результат
Запуск приложения	открыть исполняемый файл	главное окно отображается без ошибок
Загрузка двух столбцов	выбрать файл со столбцами x, y	время сформировано автоматически, данные отображены
Загрузка трех столбцов	выбрать файл со столбцами t, x, y	время считано и приведено к $[0; 1]$
Демонстрационные данные	выбрать соответствующую команду меню	загружена 201 точка, построены исходные графики
Изменение параметров	задать $m = 4, m_1 = 4, m_2 = 4$	число узлов коллокации автоматически равно 20
Коллокация	выбрать метод и нажать «Рассчитать»	получены 20 коэффициентов, графики и показатели ошибки
Метод наименьших квадратов	выбрать метод и использовать все точки	получены коэффициенты и восстановленный выход
Просмотр коэффициентов	открыть соответствующую вкладку	вектор A_i и матрица C_{ij} отображаются полностью
Просмотр ядер	открыть вкладку «Ядра модели»	построены график K_1 и поверхность K_2
Поворот поверхности	перемещать мышь с нажатой левой кнопкой	ракурс поверхности изменяется
Сохранение поверхности	выбрать команду контекстного меню	изображение сохранено в выбранном формате
Сохранение CSV	выбрать команду меню и путь	создан файл с сигналами, коэффициентами и ошибками
Недостаток точек	задать параметры, превышающие объем данных	выведено сообщение о необходимости изменить параметры
Некорректный файл	загрузить таблицу без требуемых числовых данных	выведено сообщение, приложение продолжает работу

Приложение Г

(обязательное)

Ключевые фрагменты программного кода

Листинг Г.1 – Интерфейс метода идентификации

```
public interface IIdentifier
{
    IdentificationResult Identify(
        SignalData data,
        IdentificationParameters parameters);
}
```

Листинг Г.2 – Вычисление многочленов Чебышева

```
public static double T(int n, double x)
{
    if (n < 0)
        throw new ArgumentOutOfRangeException(nameof(n));

    x = Math.Clamp(x, -1.0, 1.0);

    if (n == 0)
        return 1.0;
    if (n == 1)
        return x;

    double t0 = 1.0;
    double t1 = x;

    for (int k = 2; k <= n; k++)
    {
        double t2 = 2.0 * x * t1 - t0;
        t0 = t1;
        t1 = t2;
    }

    return t1;
}
```

Листинг Г.3 – Формирование строки расчетной матрицы

```
public static double[] BuildFeatureVectorAtTime(
    SignalInterpolator interpolator,
    IdentificationParameters parameters,
    double time)
{

```

```

time = Math.Clamp(time, 0.0, 1.0);
int order = Math.Max(
    parameters.FirstKernelBasisCount,
    Math.Max(parameters.SecondKernelBasisCountS1,
        parameters.SecondKernelBasisCountS2));

double[] beta = ComputeBetaValues(
    interpolator,
    time,
    order,
    parameters.QuadratureIntervals);

var features = new double[parameters.ParameterCount];
int column = 0;

for (int i = 0; i < parameters.FirstKernelBasisCount; i++)
    features[column++] = beta[i];

for (int i = 0; i < parameters.SecondKernelBasisCountS1; i++)
    for (int j = 0; j < parameters.SecondKernelBasisCountS2; j++)
        features[column++] = beta[i] * beta[j];

return features;
}

```

Листинг Г.4 – Реализация метода коллокации

```

public IdentificationResult Identify(
    SignalData data,
    IdentificationParameters parameters)
{
    SignalData unitTimeData =
        SignalPreprocessor.NormalizeTimeToUnitInterval(data);

    NormalizedSignalData normalized =
        SignalNormalizer.NormalizeForIdentification(
            unitTimeData,
            parameters.NormalizeSignals);

    double[] nodes =
        DesignMatrixBuilder.GetCollocationTimes(parameters);
}

```

```

var (matrix, target, _) =
    DesignMatrixBuilder.Build(
        normalized.Data,
        parameters,
        nodes);

double[] solution =
    LinearAlgebra.SolveSquareSystem(matrix, target);

return ResultFactory.Create(
    unitTimeData,
    normalized.Data,
    parameters,
    solution,
    normalized.Info,
    nodes.Length);
}

```

Листинг Г.5 – Восстановление выходного сигнала

```

public static double[] Evaluate(
    SignalData data,
    IdentificationParameters parameters,
    double[] linearCoefficients,
    double[,] quadraticCoefficients)
{
    var output = new double[data.Count];
    var interpolator = new SignalInterpolator(data);

    for (int pointIndex = 0; pointIndex < data.Count; pointIndex++)
    {
        double time = data.Points[pointIndex].Time;
        double[] features =
            DesignMatrixBuilder.BuildFeatureVectorAtTime(
                interpolator,
                parameters,
                time);

        int column = 0;
        double value = 0.0;
    }
}

```

```

        for (int i = 0; i < linearCoefficients.Length; i++)
            value += linearCoefficients[i] * features[column++];

        for (int i = 0; i < quadraticCoefficients.GetLength(0); i++)
            for (int j = 0; j < quadraticCoefficients.GetLength(1); j++)
                value += quadraticCoefficients[i, j] * features[column++];

        output[pointIndex] = value;
    }

    return output;
}

```

Листинг Г.6 – Расчет показателей ошибки и подготовка результатов

```

public static class ErrorMetricsCalculator
{
    public static ErrorMetrics Calculate(
        IReadOnlyList<double> expected,
        IReadOnlyList<double> actual)
    {
        if (expected == null)
            throw new ArgumentNullException(nameof(expected));
        if (actual == null)
            throw new ArgumentNullException(nameof(actual));
        if (expected.Count != actual.Count)
            throw new ArgumentException(
                "Arrays must have the same length");
        if (expected.Count == 0)
            throw new ArgumentException(
                "At least one point is required");

        double squaredErrorSum = 0.0;
        double squaredSignalSum = 0.0;
        double maxAbsoluteResidual = 0.0;

        for (int i = 0; i < expected.Count; i++)
        {
            double residual = expected[i] - actual[i];
            double absoluteResidual = Math.Abs(residual);

```

```

        squaredErrorSum += residual * residual;
        squaredSignalSum += expected[i] * expected[i];

        if (absoluteResidual > maxAbsoluteResidual)
            maxAbsoluteResidual = absoluteResidual;
    }

    double rmse = Math.Sqrt(squaredErrorSum / expected.Count);
    double denominator = Math.Sqrt(squaredSignalSum);
    double relativeError = denominator > 0.0
        ? Math.Sqrt(squaredErrorSum) / denominator
        : 0.0;

    return new ErrorMetrics(
        rmse,
        relativeError,
        maxAbsoluteResidual);
}
}

public static class SignalPreprocessor
{
    public static SignalData NormalizeTimeToUnitInterval(SignalData data)
    {
        if (data == null)
            throw new ArgumentNullException(nameof(data));
        if (data.Count < 2)
            throw new InvalidOperationException(
                "Signal must contain at least two points");

        double minTime = data.Points[0].Time;
        double maxTime = data.Points[data.Count - 1].Time;
        double length = maxTime - minTime;

        if (length <= 0.0)
            throw new InvalidOperationException(
                "Time values must be increasing");

        var points = new List<SignalPoint>(data.Count);

```

```

    for (int i = 0; i < data.Count; i++)
    {
        double normalizedTime =
            (data.Points[i].Time - minTime) / length;

        points.Add(new SignalPoint(
            normalizedTime,
            data.Points[i].Input,
            data.Points[i].Output));
    }

    return new SignalData(points);
}

public static SignalData ValidateAndCopy(SignalData data)
{
    if (data == null)
        throw new ArgumentNullException(nameof(data));
    if (data.Count < 5)
        throw new InvalidOperationException(
            "Input file must contain at least five rows");

    var points = new List<SignalPoint>(data.Count);
    double previousTime = double.NegativeInfinity;

    for (int i = 0; i < data.Count; i++)
    {
        SignalPoint point = data.Points[i];

        if (double.IsNaN(point.Time) ||
            double.IsInfinity(point.Time) ||
            double.IsNaN(point.Input) ||
            double.IsInfinity(point.Input) ||
            double.IsNaN(point.Output) ||
            double.IsInfinity(point.Output))
        {
            throw new InvalidOperationException(
                "The table contains nonnumeric values");
        }
    }
}

```

```
        if (point.Time <= previousTime)
            throw new InvalidOperationException(
                "Time values must be strictly increasing");

        points.Add(point);
        previousTime = point.Time;
    }

    return new SignalData(points);
}

}
```