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

Кафедра «Прикладная математика и информатика»
(наименование)

01.03.02 «Прикладная математика и информатика»
(код и наименование направления подготовки / специальности)

Компьютерные технологии и математическое моделирование
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему Исследование численных алгоритмов в задачах распространения сигналов

Обучающийся И.С. Еремин  (личная подпись)
(Инициалы Фамилия)

Руководитель д.ф.-м.н., доцент С.В. Талалов
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Консультант к.ф.н., доцент Н. В. Ященко
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2025

Аннотация

В выпускной квалификационной работе исследуются численные алгоритмы решения задачи распространения сигналов через линейный фильтр с заданной частотной характеристикой. Основной фокус работы направлен на решение обратной задачи фильтрации – восстановление входного сигнала по известному выходному, что математически сводится к интегральному уравнению Фредгольма первого рода.

В работе проведен анализ теоретических основ теории сигналов, интегральных уравнений и методов регуляризации некорректных задач. Выведено интегральное уравнение для линейного фильтра с характеристикой $\rho(\omega) = (1 - |\omega|/\omega_0)\theta(\omega_0 - |\omega|)$, где $\theta(\omega)$ – функция Хевисайда, а ω_0 – частота отсечки.

Разработан алгоритм решения задачи, основанный на методе аппроксимации функций полиномами Лежандра, который позволяет свести интегральное уравнение к системе линейных алгебраических уравнений. Рассмотрены вопросы определения порядка аппроксимации, анализа обусловленности получаемой системы и применения методов регуляризации для обеспечения устойчивости решения.

Практическая значимость работы заключается в разработке эффективного алгоритма восстановления входного сигнала, который может быть использован в системах связи, обработке медицинских изображений, акустическом анализе и других областях, где требуется восстановление сигналов, искаженных при прохождении через линейные фильтры.

Ключевые слова: интегральные уравнения, обработка сигналов, некорректные задачи, регуляризация, полиномы Лежандра, линейные фильтры, численные методы.

Abstract

The final qualification work studies numerical algorithms for solving the problem of signal propagation through a linear filter with a given frequency response. The main focus of the work is on solving the inverse filtering problem - restoring the input signal from a known output signal, which mathematically reduces to the Fredholm integral equation of the first kind.

The work analyzes the theoretical foundations of signal theory, integral equations and methods for regularizing ill-posed problems. An integral equation is derived for a linear filter with the characteristic $\rho(\omega) = (1 - |\omega|/\omega_0)\theta(\omega_0 - |\omega|)$, where $\theta(\omega)$ is the Heaviside function, and ω_0 is the cutoff frequency.

An algorithm for solving the problem is developed based on the method of approximating functions by Legendre polynomials, which allows reducing the integral equation to a system of linear algebraic equations. The issues of determining the order of approximation, analyzing the conditionality of the resulting system and applying regularization methods to ensure the stability of the solution are considered.

The practical significance of the work lies in the development of an effective algorithm for restoring the input signal, which can be used in communication systems, medical image processing, acoustic analysis and other areas where it is necessary to restore signals distorted when passing through linear filters.

Key words: integral equations, signal processing, ill-posed problems, regularization, Legendre polynomials, linear filters, numerical methods.

Содержание

Введение

Актуальность определяется теоретическими аспектами исследования некорректных задач и их практическим применением в системах обработки сигналов.

Цифровая обработка сигналов требует восстановления исходного сигнала по искаженным данным. Системы связи компенсируют искажения каналов передачи; медицинская томография восстанавливает внутреннюю структуру объекта по проекциям; геофизика определяет структуру земной коры по сейсмическим наблюдениям. Эти задачи моделируются интегральными уравнениями Фредгольма первого рода.

Разработки А.Н. Тихонова, В.К. Иванова, М.М. Лаврентьева продвинули методы регуляризации некорректных задач, но проблема решения интегральных уравнений Фредгольма первого рода не получила окончательного решения. Существующие методы требуют больших вычислительных ресурсов и настройки параметров регуляризации, затрудняя практическое применение.

Данная работа изучает линейный фильтр с характеристикой $\rho(\omega) = (1 - |\omega|/\omega_0)\theta(\omega_0 - |\omega|)$, где $\theta(\omega)$ - функция Хевисайда, ω_0 - частота отсечки. Фильтр подавляет высокочастотные составляющие сигнала, сохраняя низкочастотные компоненты. Разработка численных методов решения соответствующего интегрального уравнения улучшит восстановление сигналов в прикладных задачах.

Использование разложения функций в ряд по полиномам Лежандра позволяет свести решение интегрального уравнения к системе линейных алгебраических уравнений, что значительно упрощает численную реализацию алгоритма. Такой подход дает возможность точнее контролировать ошибки аппроксимации и более эффективно применять методы регуляризации [38].

Цель данной работы – исследование численных алгоритмов решения задачи распространения сигналов на основе интегрального уравнения Фредгольма первого рода.

Для достижения поставленной цели сформулированы следующие задачи:

– проанализировать существующие методы решения интегральных уравнений Фредгольма первого рода, выявить их преимущества и недостатки применительно к задачам обработки сигналов. Особое внимание требуется уделить проблеме некорректности таких уравнений и методам её преодоления.

– вывести аналитическое выражение интегрального уравнения, связывающего исходный сигнал f(t) и выходной сигнал g(t) для линейного фильтра с характеристикой $\rho(\omega) = (1 - |\omega|/\omega_0)\theta(\omega_0 - |\omega|)$. Определить структуру ядра и его особенности.

– разработать блок-схему алгоритма решения задачи на основе аппроксимации ядра интегрального уравнения выбранным ядром порядка N и сведения к системе линейных алгебраических уравнений.

– создать программную реализацию алгоритма с возможностью анализа числа обусловленности матрицы СЛАУ и применения различных методов регуляризации.

– исследовать влияние ключевых параметров (точности ε, частоты отсечки ω_0, параметра регуляризации) на точность восстановления сигнала и вычислительную эффективность.

– провести сравнительный анализ эффективности разработанного алгоритма с существующими методами решения подобных задач.

– сформулировать практические рекомендации по выбору оптимальных параметров алгоритма для различных условий применения.

В работе применяются следующие методы исследования [20]:

– аналитические методы преобразования Фурье для вывода интегрального уравнения Фредгольма из частотной характеристики фильтра.

– аппроксимация функций с помощью полиномов Лежандра для сведения интегрального уравнения к системе линейных алгебраических уравнений.

– численные методы решения систем линейных алгебраических уравнений.

– методы анализа устойчивости решений, включая оценку числа обусловленности матрицы.

– метод регуляризации для повышения устойчивости решения.

Применение полиномов Лежандра обусловлено их ортогональностью на отрезке [-1, 1], что соответствует нормированному интервалу измерений сигнала. Метод разложения по полиномам Лежандра позволяет учесть особенности исследуемой задачи и добиться более высокой точности аппроксимации.

Объектом исследования выступают процессы распространения сигналов через линейные фильтры с заданной частотной характеристикой.

Предметом исследования являются численные методы решения интегральных уравнений Фредгольма первого рода, возникающих при анализе прохождения сигналов через линейные фильтры.

Практическая значимость работы заключается в разработке эффективного алгоритма восстановления входного сигнала по известному выходному сигналу фильтра с заданной частотной характеристикой.

Результаты исследования могут найти применение в системах связи, обработке медицинских изображений, акустическом анализе и других областях, где требуется восстановление сигналов, искаженных при прохождении через линейные фильтры.

1 Теоретические основы исследования задач распространения сигналов

1.1 Математические основы теории сигналов

Сигналом называют любую физическую величину, изменяющуюся во времени и несущую информацию. Математически сигнал представляется функцией одной или нескольких переменных. Для одномерных сигналов используется функция времени s(t), а многомерные сигналы (например, изображения) задаются функциями нескольких пространственных координат.

Сигналы классифицируются по различным признакам. По характеру изменения во времени выделяют детерминированные и случайные сигналы. Детерминированный сигнал полностью определен математическим выражением, а случайный – известен лишь с определенной вероятностью. По характеру функции времени различают непрерывные, дискретные и цифровые сигналы. Часто в литературе встречается термин «аналоговый сигнал», который является синонимом непрерывного сигнала, так как при аналоговой передаче физическая величина (например, напряжение) непрерывно изменяется пропорционально передаваемой информации [40].

Непрерывный сигнал определен для всех значений времени t и принимает любые значения из некоторого интервала. Дискретный сигнал задан только в фиксированные моменты времени$t_n = n\Delta t$, где n – целое число, а Δt– интервал дискретизации. Цифровой сигнал отличается от дискретного тем, что кроме дискретизации по времени он квантуется по уровню. Такие сигналы обрабатываются цифровыми устройствами и используются в современных системах связи [9].

Особый класс сигналов – периодические сигналы, повторяющиеся через равные промежутки времени T: s(t + T) = s(t). С точки зрения спектрального анализа, периодические сигналы имеют дискретный спектр – набор гармонических составляющих с частотами, кратными основной частоте $f_0 =$

$1/T$. Известный пример – прямоугольные импульсы, широко применяемые в цифровой электронике для представления бинарной информации [1].

Энергетические характеристики сигналов включают мощность, энергию и энергетический спектр. Мощность определяет среднюю энергию сигнала за единицу времени. Энергетический спектр показывает распределение энергии сигнала по частотам – эта характеристика особенно важна при анализе прохождения сигнала через линейные фильтры, поскольку позволяет определить, какие частотные компоненты будут усилены или ослаблены. Основные классы сигналов и их характеристики приведены в таблице 1.

Таблица 1 – Основные классы сигналов и их характеристики

Класс сигналов	Определение	Особенности спектра	Примеры
Непрерывные	Определены для всех t	Может быть как непрерывным, так и дискретным	Речь, музыка
Дискретные	Определены для $t = n\Delta t$	Периодический с периодом $1/\Delta t$	ИКМ сигналы
Периодические	$s(t+T) = s(t)$	Дискретный с шагом $1/T$	Гармоники
Финитные	$s(t) = 0$ при $t \notin [t_1, t_2]$	Непрерывный	Импульсы
Случайные	Заданы статистически	Определяется через спектральную плотность мощности	Шум

Теорема Котельникова-Найквиста играет ключевую роль в цифровой обработке сигналов. Она утверждает, что сигнал с ограниченным спектром ($F(\omega) = 0$ при $|\omega| > \omega m$) может быть точно восстановлен по своим отсчетам, взятым с частотой $fd > 2fm$. Это результат устанавливает нижнюю границу частоты дискретизации, необходимую для точного представления аналогового сигнала в цифровой форме. На практике для учета неидеальности реальных фильтров частоту дискретизации выбирают с запасом, используя коэффициент 2,2-2,5 вместо теоретического минимума 2. Спектральные методы анализа сигналов – базовый инструмент в теории сигналов [24]. Они основаны на представлении сигналов в виде суммы или интеграла простейших

гармонических колебаний. Преобразование Фурье сопоставляет функции времени другую функцию – ее спектр, который описывает амплитуды и фазы гармонических составляющих исходного сигнала [2].

Прямое преобразование Фурье для непрерывного сигнала f(t) дает его спектр F(ω), а обратное преобразование восстанавливает сигнал по его спектру. Комплексный спектр сигнала F(ω) содержит информацию об амплитудном и фазовом спектрах. Для практического применения особенно важна теорема о свертке, связывающая свертку функций во временной области с произведением их спектров в частотной области. Операция свертки двух функций имеет глубокий физический смысл. Например, свертка входного сигнала с импульсной характеристикой линейной системы дает выходной сигнал этой системы - эта операция лежит в основе математического описания линейной фильтрации – одного из основных методов обработки сигналов [30].

В цифровой обработке сигналов используется дискретное преобразование Фурье (ДПФ). В отличие от непрерывного преобразования, ДПФ применяется к последовательностям конечной длины и дает набор комплексных коэффициентов, описывающих спектр дискретного сигнала. Алгоритм быстрого преобразования Фурье (БПФ) – эффективный метод вычисления ДПФ, значительно сокращающий количество требуемых арифметических операций [27].

Для обработки нестационарных сигналов, спектральный состав которых меняется со временем (речь, музыка, многие биологические сигналы), используются методы частотно-временного анализа. Оконное преобразование Фурье – простейший из таких методов – заключается в применении преобразования Фурье к различным участкам сигнала, выделенным с помощью окна, которое сдвигается вдоль оси времени. Недостаток этого метода – фиксированное разрешение по времени и частоте. Более совершенные инструменты – вейвлет-преобразование и преобразование Вигнера-Вилля.

Линейная фильтрация – фундаментальная операция обработки сигналов. Фильтры модифицируют спектральный состав сигналов, усиливая или ослабляя определенные частотные компоненты. По виду амплитудно-частотной характеристики (АЧХ) различают фильтры низких частот (ФНЧ), высоких частот (ФВЧ), полосовые и режекторные фильтры. ФНЧ пропускает низкие частоты и подавляет высокие, ФВЧ действует противоположным образом [33]. Полосовой фильтр выделяет частоты в заданном диапазоне, а режекторный – подавляет их [3]. Типы цифровых фильтров и их применение приведены в таблице 2.

Таблица 2 – Типы цифровых фильтров и их применение

Тип фильтра	Характеристика	Применение
ФНЧ	Пропускает частоты ниже частоты среза	Сглаживание сигналов, устранение высокочастотного шума
ФВЧ	Пропускает частоты выше частоты среза	Выделение быстрых изменений сигнала, устранение постоянной составляющей
Полосовой	Пропускает частоты в заданном диапазоне	Выделение сигналов определенной частоты (например, голоса в телефонии)
Режекторный	Подавляет частоты в заданном диапазоне	Устранение сетевой наводки 50/60 Гц
Гребенчатый	Имеет периодическую АЧХ	Разделение сигналов с периодическим спектром

Идеальные фильтры с прямоугольной АЧХ физически нереализуемы, поскольку требуют бесконечной импульсной характеристики и нулевой задержки сигнала. На практике используют различные аппроксимации: фильтры Баттерворта, имеющие максимально плоскую АЧХ в полосе пропускания; фильтры Чебышева, обеспечивающие более крутой спад АЧХ ценой появления пульсаций в полосе пропускания или задерживания; эллиптические фильтры, у которых пульсации присутствуют в обеих полосах, но обеспечивается самый крутой спад.

Цифровые фильтры делятся на фильтры с конечной импульсной характеристикой (КИХ-фильтры) и с бесконечной импульсной характеристикой (БИХ-фильтры). КИХ-фильтры всегда устойчивы, могут иметь строго линейную фазо-частотную характеристику (ФЧХ) и удобны для реализации на многопроцессорных системах. БИХ-фильтры более экономичны по числу вычислений, но могут быть неустойчивы и всегда имеют нелинейную ФЧХ. Для физической реализации аналоговых фильтров используются различные схемы на основе резисторов, конденсаторов и катушек индуктивности (пассивные фильтры) или с добавлением операционных усилителей (активные фильтры). Цифровые фильтры реализуются программно или аппаратно как последовательность арифметических операций.

Для анализа линейных систем широко используется z-преобразование – дискретный аналог преобразования Лапласа. Оно переводит разностные уравнения, описывающие цифровые фильтры, в алгебраические, что значительно упрощает их анализ и синтез. Передаточная функция H(z) полностью характеризует линейную систему с дискретным временем. Расположение полюсов и нулей передаточной функции на комплексной z-плоскости определяет устойчивость системы и форму ее частотной характеристики [6].

Помимо фильтрации, к базовым операциям обработки сигналов относятся модуляция и демодуляция. Модуляция – процесс изменения параметров несущего сигнала (обычно высокочастотного) в соответствии с передаваемой информацией (низкочастотным модулирующим сигналом). Различают амплитудную (АМ), частотную (ЧМ), фазовую (ФМ) модуляцию и их разновидности. Модуляция позволяет передавать сигналы по каналам с ограниченной полосой пропускания, разделять несколько сигналов в общем канале связи (частотное и временное мультиплексирование), улучшать помехоустойчивость и энергетическую эффективность передачи [4].

Случайные сигналы и шумы играют важную роль в теории сигналов. В отличие от детерминированных сигналов, случайные сигналы могут быть описаны только статистически – через вероятностные характеристики: функцию распределения, плотность вероятности, моменты (среднее значение, дисперсия и др.). Для случайных процессов вводятся понятия стационарности (неизменности статистических характеристик во времени), эргодичности (возможности замены усреднения по множеству реализаций усреднением по времени для одной реализации), корреляционной функции (меры статистической связи значений процесса в различные моменты времени) и спектральной плотности мощности (распределения средней мощности случайного процесса по частотам).

В теории сигналов занимает белый шум – случайный процесс с равномерной спектральной плотностью мощности во всем диапазоне частот. Хотя идеальный белый шум физически нереализуем (требует бесконечной мощности), он служит удобной математической моделью для многих реальных шумовых процессов в ограниченной полосе частот. Прохождение случайных сигналов через линейные системы – важная задача, возникающая при анализе систем связи, радиолокации и других приложений. При этом спектральная плотность мощности выходного случайного процесса равна произведению спектральной плотности мощности входного процесса на квадрат модуля частотной характеристики системы.

1.2 Интегральные уравнения в задачах обработки сигналов

Интегральные уравнения – класс уравнений, в которых неизвестная функция входит под знак интеграла - эти уравнения возникают во многих прикладных задачах, включая задачи обработки сигналов. В зависимости от пределов интегрирования интегральные уравнения делятся на уравнения Фредгольма (с постоянными пределами интегрирования) и уравнения

Вольтерра (с переменным верхним пределом). Уравнения Фредгольма первого рода имеют вид [5]:

$$\int_a^b K(s,t)f(t)dt = g(s), \quad (1)$$

где K(s,t) – ядро интегрального уравнения,

f(t) – искомая функция,

g(s) – известная функция (правая часть уравнения).

Интегрирование производится по некоторой области D. В одномерном случае эта область представляет отрезок [a,b]. Уравнения Фредгольма второго рода содержат неизвестную функцию также вне интеграла [25]:

$$f(s) - \lambda\int_a^b K(s,t)f(t)dt = g(s), \quad (2)$$

где λ – числовой параметр.

Если λ = 0, то уравнение решается тривиально: f(s) = g(s). Уравнения Вольтерра первого рода записываются в виде, где интегрирование производится по переменному интервалу от a до s :

$$\int_a^s K(s,t)f(t)dt = g(s),. \quad (3)$$

Уравнения Вольтерра второго рода аналогичны уравнениям Фредгольма второго рода, но с переменным верхним пределом интегрирования. Если ядро K(s,t) зависит только от разности аргументов: K(s,t) = K(s-t), то интегральное уравнение называется уравнением свертки. Такие уравнения особенно часто встречаются в задачах обработки сигналов, поскольку описывают воздействие линейных стационарных систем. Уравнение свертки первого рода имеет вид:

$$\int_a^b K(s-t)f(t)dt = g(s). \quad (4)$$

В теории сигналов это уравнение возникает при решении задачи восстановления входного сигнала f(t) по известному выходному сигналу g(s) и известной импульсной характеристике системы K(s-t). Уравнения Фредгольма первого рода относятся к классу некорректно поставленных задач. Некорректность в смысле Адамара означает, что хотя бы одно из следующих условий не выполняется [6]:

– решение существует;

– решение единственно;

– решение устойчиво по отношению к малым изменениям исходных данных.

Именно некорректность задачи решения уравнения Фредгольма первого рода создает основные трудности в задачах обработки сигналов, связанных с восстановлением входного сигнала по известному выходному. Даже небольшие погрешности измерения выходного сигнала g(s) или неточности в определении ядра K(s,t) могут приводить к значительным ошибкам в восстановленном сигнале f(t). Физически некорректность объясняется тем, что интегральный оператор сглаживает высокочастотные компоненты входного сигнала. При обратной задаче – восстановлении входного сигнала по выходному – требуется усиление высоких частот, что приводит к усилению шума и неустойчивости решения [18].

Классификация интегральных уравнений приведена в таблице 3.

Таблица 3 – Классификация интегральных уравнений

Тип уравнения	Математическая форма	Характеристики	Примеры в обработке сигналов
Фредгольма I рода	$\int K(s,t)f(t)dt = g(s)$	Некорректная задача	Восстановление сигнала по его искаженной версии
Фредгольма II рода	$f(s) - \lambda\int K(s,t)f(t)dt = g(s)$	Обычно корректная задача	Моделирование распространения сигналов в средах с отражениями
Вольтерра I рода	$\int_{[a,s]} K(s,t)f(t)dt = g(s)$	Может быть сведена к дифференциальному уравнению	Задачи идентификации причинных систем
Вольтерра II рода	$f(s) - \lambda\int_{[a,s]} K(s,t)f(t)dt = g(s)$	Корректная задача	Моделирование причинных систем с обратной связью
Уравнение свертки	$\int K(s-t)f(t)dt = g(s)$	Удобно решать в частотной области	Восстановление сигнала при известной передаточной функции

Методы решения интегральных уравнений Фредгольма первого рода можно разделить на прямые методы и методы регуляризации. Прямые методы включают различные способы дискретизации уравнения, приводящие к системе линейных алгебраических уравнений (СЛАУ). Простейший метод – метод квадратур, в котором интеграл заменяется конечной суммой с использованием квадратурных формул (прямоугольников, трапеций, Симпсона и др.). Метод коллокации состоит в требовании точного выполнения уравнения в конечном числе точек (узлов коллокации).

Метод Галеркина (метод взвешенных невязок) основан на ортогонализации невязки уравнения по отношению к некоторой системе базисных функций. Эти методы приводят к СЛАУ, однако из-за некорректности исходной задачи получаемая система оказывается плохо обусловленной – малые изменения правой части могут вызывать большие

изменения решения. Оценить степень обусловленности матрицы СЛАУ можно с помощью числа обусловленности – отношения наибольшего сингулярного числа матрицы к наименьшему. Большое число обусловленности (например, 10^6 и более) указывает на плохую обусловленность системы и необходимость применения специальных методов для ее решения.

Методы регуляризации направлены на преодоление проблемы некорректности. Основная идея – заменить исходную некорректную задачу близкой к ней корректной задачей, решение которой при стремлении погрешности исходных данных к нулю приближается к решению исходной задачи. Наиболее известен метод регуляризации Тихонова, в котором вместо минимизации невязки $\|Kf - g\|^2$ минимизируется функционал [7]:

$$J_\alpha[f] = |Kf - g|^2 + \alpha|Lf|^2, \quad (5)$$

где $\alpha > 0$ – параметр регуляризации;

L – некоторый оператор (обычно дифференциальный).

Второе слагаемое является стабилизирующим функционалом, который вносит дополнительную информацию о решении (например, о его гладкости). Выбор параметра регуляризации α представляет отдельную проблему. При слишком малых α решение будет неустойчивым, а при слишком больших – сильно сглаженным, далеким от точного. Существуют различные методы выбора α: метод невязки, метод L-кривой, метод обобщенной перекрестной проверки, принцип квазиоптимальности. Упрощенный метод регуляризации заключается в сведении интегрального уравнения к СЛАУ и последующем добавлении к матрице системы диагональной матрицы αI. Это эквивалентно добавлению к функционалу невязки стабилизирующего функционала $\alpha\|f\|^2$.

Другой подход к решению некорректных задач – метод усеченного сингулярного разложения (TSVD). В этом методе при решении СЛАУ отбрасываются компоненты решения, соответствующие малым сингулярным

числам матрицы системы. Это позволяет уменьшить влияние погрешностей на решение, хотя и приводит к его сглаживанию.

Эффективным подходом к решению интегральных уравнений свертки является использование преобразования Фурье. Применение преобразования Фурье к обеим частям уравнения позволяет превратить свертку в произведение спектров [8]:

$$F[K \cdot f] = F[K] \cdot F[f] = F[g]. \quad (6)$$

Откуда формально получаем:

$$F[f] = \frac{F[g]}{F[K]}. \quad (7)$$

И, применяя обратное преобразование Фурье:

$$f = F^{-1}\left[\frac{F[g]}{F[K]}\right]. \quad (8)$$

Однако этот подход также сталкивается с проблемой некорректности: при наличии в спектре K(t) близких к нулю значений или при зашумленности g(t) решение может оказаться неустойчивым. Для преодоления этой проблемы используются различные модификации метода, включающие регуляризацию в частотной области – например, применение сглаживающих фильтров к спектру F[g] перед делением на F[K]. Метод винеровской фильтрации основан на минимизации среднеквадратичной ошибки восстановления сигнала при наличии шума. Фильтр Винера учитывает спектральные характеристики как сигнала, так и шума, что позволяет достичь оптимального компромисса между точностью восстановления сигнала и подавлением шума.

В задачах обработки изображений и сигналов также применяются итерационные методы решения интегральных уравнений. Метод

последовательных приближений, метод наименьших квадратов с регуляризацией, метод сопряженных градиентов позволяют эффективно решать большие системы уравнений, возникающие при дискретизации интегральных уравнений. Метод аппроксимации ядра выражает ядро интегрального уравнения K(s,t) через конечный набор функций, что позволяет свести интегральное уравнение к СЛАУ. Особый случай – вырожденное ядро, представимое в виде конечной суммы [9]:

$$K(s,t) = \sum_{i=1}^{n} a_i(s) b_i(t). \quad (9)$$

Для такого ядра решение интегрального уравнения сводится к решению системы из n линейных уравнений.

Для приближения произвольного ядра вырожденным можно использовать разложение в ряд по некоторой полной системе функций, усеченное до n членов. В частности, разложение по полиномам Лежандра удобно для задач, в которых область интегрирования – отрезок [-1, 1].

Особый класс интегральных уравнений – интегральные уравнения с разностным ядром, зависящим только от разности аргументов: K(s,t) = K(s-t). Для таких уравнений эффективны спектральные методы, основанные на использовании преобразования Фурье. Этот подход особенно актуален в задачах обработки сигналов, где ядро представляет импульсную характеристику линейной системы.

Методы решения интегральных уравнений Фредгольма первого рода приведены в таблице 4.

Таблица 4 – Методы решения интегральных уравнений Фредгольма первого рода

Метод	Принцип	Преимущества	Недостатки
Метод квадратур	Замена интеграла конечной суммой	Простота реализации	Неустойчивость решения
Метод регуляризации Тихонова	Минимизация сглаживающего функционала	Устойчивость решения	Сложность выбора параметра регуляризации
Метод TSVD	Отбрасывание компонент решения, соответствующих малым сингулярным числам	Эффективное подавление шума	Потеря высокочастотных компонент решения
Спектральные методы	Использование преобразования Фурье	Эффективность для уравнений свертки	Неустойчивость при наличии нулей в спектре ядра
Итерационные методы	Последовательное уточнение решения	Возможность учета априорных ограничений	Медленная сходимость для плохо обусловленных задач
Метод аппроксимации ядра	Представление ядра в виде конечной суммы	Сведение к СЛАУ малой размерности	Ошибки аппроксимации ядра

Помимо интегральных уравнений с действительными ядрами, в задачах обработки сигналов встречаются интегральные преобразования с комплексными ядрами. Например, преобразование Фурье, преобразование Лапласа, Z-преобразование - эти преобразования играют важную роль в анализе и синтезе систем обработки сигналов. Для решения интегральных уравнений также применяются методы проекции. Они основаны на аппроксимации искомой функции линейной комбинацией базисных функций. Выбор базиса существенно влияет на эффективность метода. Часто используются полиномиальные базисы (полиномы Чебышева, Лежандра, Лагерра), тригонометрические функции, вейвлеты.

Выбор конкретного метода решения интегрального уравнения зависит от многих факторов: типа ядра, гладкости правой части, требуемой точности, доступных вычислительных ресурсов. Для задач обработки сигналов учитываются также особенности физической природы сиг

Для задач обработки сигналов учитываются также особенности физической природы сигналов, наличие шума, требования к быстродействию алгоритма. Ряд задач обработки сигналов приводит к интегральным уравнениям с особенностями. Например, при наличии разрывов в ядре или правой части уравнения требуются специальные методы численного интегрирования, учитывающие эти особенности. Методы выделения особенностей, адаптивного выбора узлов квадратурных формул, сингулярного разложения помогают повысить точность решения [37].

Байесовский подход к решению некорректных задач основан на представлении решения и данных как случайных величин с известными априорными распределениями. Решение находится как апостериорное распределение, максимизирующее вероятность при заданных наблюдениях - этот подход естественным образом включает регуляризацию через априорные предположения о решении [4].

В системах реального времени требуются эффективные алгоритмы решения интегральных уравнений, способные работать с потоковыми данными. Рекурсивные методы, позволяющие обновлять решение при поступлении новых данных без полного пересчета, имеют в этом случае значительное преимущество. Проблема некорректности решения интегральных уравнений Фредгольма первого рода имеет глубокие математические корни. Теорема Римана-Лебега утверждает, что преобразование Фурье функции из L^1 стремится к нулю на бесконечности - это означает, что высокочастотные компоненты спектра ядра интегрального оператора могут быть сколь угодно малы, делая задачу обращения этого оператора некорректной [35].

Особенности решения интегральных уравнений в различных прикладных задачах приведены в таблице 5.

Таблица 5 – Особенности решения интегральных уравнений в различных прикладных задачах

Область применения	Тип уравнения	Методы решения	Специфические проблемы
Обработка изображений	Фредгольма I рода (восстановление размытых изображений)	Винеровская фильтрация, регуляризация Тихонова	Большая размерность, анизотропное размытие
Томография	Фредгольма I рода (преобразование Радона)	Метод обратной проекции с фильтрацией	Неполные данные, геометрические искажения
Спектроскопия	Фредгольма I рода	Регуляризация, метод максимальной энтропии	Широкий динамический диапазон спектров
Геофизика	Вольтерра I рода	Деконволюция, методы оптимальной фильтрации	Нестационарность сред, многомерность
Акустика	Фредгольма II рода (интегральные уравнения рассеяния)	Метод граничных элементов	Высокая частота колебаний, сложная геометрия

В задачах цифровой обработки сигналов интегральные уравнения часто дискретизируются, что приводит к системам линейных алгебраических уравнений. Устойчивость решения таких систем определяется числом обусловленности матрицы – отношением максимального сингулярного числа к минимальному. Большое число обусловленности указывает на плохую обусловленность задачи и необходимость применения методов регуляризации. Метод регуляризации Тихонова основан на минимизации функционала, включающего невязку и стабилизирующий функционал. Выбор стабилизирующего функционала определяется априорной информацией о решении. Для гладких решений используют функционалы, содержащие производные искомой функции; для разрывных решений – функционалы, допускающие разрывы (например, основанные на вариации функции).

Важной задачей при использовании методов регуляризации является выбор параметра регуляризации. Метод невязки основан на принципе согласования погрешности и невязки: параметр регуляризации выбирается так, чтобы невязка была сравнима с уровнем шума в данных. Метод L-кривой анализирует зависимость нормы решения от нормы невязки при различных значениях параметра регуляризации. Принцип обобщенной перекрестной проверки основан на статистическом подходе к оценке ошибки предсказания.

Для задач с большой размерностью эффективны итерационные методы регуляризации, такие как метод наименьших квадратов с регуляризацией и метод сопряженных градиентов с ранним остановом. В этих методах число итераций играет роль параметра регуляризации: слишком малое число итераций приводит к сильному сглаживанию решения, а слишком большое – к неустойчивости. Практическое применение методов решения интегральных уравнений в обработке сигналов включает задачи восстановления сигналов, искаженных при прохождении через линейные системы; реконструкцию изображений в томографии; деконволюцию в геофизике и спектроскопии; идентификацию систем по их импульсной или частотной характеристике [19].

Для некоторых типов интегральных уравнений существуют точные аналитические решения. Например, для интегральных уравнений с вырожденным ядром или для уравнений специального вида (уравнение Абеля, уравнение Вольтерра с ядром типа свертки). Однако для большинства практических задач приходится использовать численные методы.

Аппроксимация ядра интегрального уравнения с помощью полиномов Лежандра особенно удобна для задач, определенных на отрезке [-1, 1], поскольку эти полиномы ортогональны на данном отрезке с весом 1. Разложение функции g(x) по полиномам Лежандра имеет вид [10]:

$$g(x) = \sum_{n=0}^{\infty} g^{(n)} P_n(x), \tag{10}$$

где $P_n(x)$ – полиномы Лежандра.

Коэффициенты $g^{(n)}$вычисляются через скалярное произведение:

$$g^{(n)} = \frac{2n+1}{2} \int_{-1}^{1} g(x) P_n(x) dx. \quad (11)$$

Аппроксимация ядра с помощью полиномов Лежандра приводит к вырожденному ядру вида:

$$K(s,t) \approx \sum_{i=0}^{N} \sum_{j=0}^{N} k_{ij} P_i(s) P_j(t). \quad (12)$$

Это позволяет свести интегральное уравнение к системе линейных уравнений относительно коэффициентов разложения искомой функции. Выбор порядка аппроксимации N определяется требуемой точностью и степенью некорректности задачи. Слишком большие значения N могут приводить к неустойчивости решения из-за усиления влияния погрешностей в исходных данных. С другой стороны, слишком малые значения N дают грубую аппроксимацию, не позволяющую точно восстановить сигнал.

Современные методы решения интегральных уравнений в задачах обработки сигналов часто используют комбинацию различных подходов: спектральных методов, методов регуляризации, итерационных алгоритмов. Разработка адаптивных алгоритмов, автоматически выбирающих оптимальные параметры регуляризации и метод решения в зависимости от свойств конкретной задачи, представляет собой активную область исследований.

С развитием вычислительной техники стали доступны более сложные методы решения интегральных уравнений, требующие значительных вычислительных ресурсов. Применение параллельных вычислений, графических процессоров, специализированных аппаратных ускорителей позволяет решать задачи большой размерности в реальном времени, что особенно важно для приложений, связанных с обработкой сигналов.

В заключение отметим, что теория интегральных уравнений и методы их решения играют ключевую роль в задачах цифровой обработки сигналов. Некорректность обратных задач, таких как восстановление сигналов по их искаженным версиям, требует применения специальных методов, обеспечивающих устойчивость решения. Развитие эффективных алгоритмов решения интегральных уравнений остается актуальной задачей, имеющей важное практическое значение для многих областей науки и техники.

1.3 Методы регуляризации некорректных задач

Ж. Адамар определил корректность математической задачи через три условия: существование решения, его единственность и устойчивость относительно малых изменений исходных данных. Нарушение любого из этих условий делает задачу некорректно поставленной. Интегральные уравнения Фредгольма первого рода типично нарушают третье условие – малые изменения в правой части g(s) могут вызывать большие изменения в решении f(t). Физически некорректность связана с подавлением высокочастотных компонент при прямом преобразовании. Ядро K(s,t) действует как фильтр, сглаживающий входной сигнал; обратная задача требует усиления высоких частот, что усиливает шум и делает решение неустойчивым. Математически некорректность проявляется через быстрое убывание сингулярных чисел оператора интегрального уравнения [11].

В таблице 6 приведена rлассификация задач по степени некорректности.

Таблица 6 – Классификация задач по степени некорректности

Класс задачи	Скорость убывания сингулярных чисел	Примеры задач
Умеренно некорректные	Степенная: $\sigma_n \sim n^{-p}$, $0 < p < \infty$	Численное дифференцирование, обращение преобразования Абеля
Сильно некорректные	Экспоненциальная: $\sigma_n \sim e^{-kn}$	Аналитическое продолжение, обращение преобразования Лапласа
Существенно некорректные	Сверхэкспоненциальная: $\sigma_n \sim e^{-knr}$, $r > 1$	Обратная задача теплопроводности

На практике некорректность проявляется в плохой обусловленности матрицы СЛАУ, получаемой при дискретизации интегрального уравнения. Число обусловленности такой матрицы может достигать 10^{10}-10^{15}, делая прямое решение системы неустойчивым даже при использовании двойной точности вычислений [21].

Идея регуляризации – замена исходной некорректной задачи близкой корректной, решение которой стремится к точному при стремлении погрешности данных к нулю. Основные классические методы включают: метод регуляризации Тихонова, метод усеченного сингулярного разложения (TSVD), методы итеративной регуляризации, метод статистической регуляризации. Метод TSVD основан на сингулярном разложении оператора [12]

$$K = \sum_{i=1}^{\infty} \sigma_i (u_i, \cdot) v_i. \quad (13)$$

Формальное решение выражается через это разложение:

$$f = \sum_{i=1}^{\infty} \frac{(g, u_i)}{\sigma_i} v_i. \quad (14)$$

Некорректность проявляется в малых знаменателях σ_i. Метод TSVD ограничивает сумму первыми k членами:

$$f_k = \sum_{i=1}^{k} \frac{(g,u_i)}{\sigma_i} v_i. \quad (15)$$

Методы итеративной регуляризации используют свойство полурегулярности итерационных процессов: начальные итерации восстанавливают низкочастотные компоненты, высокочастотные появляются позже. Остановка процесса на определенном шаге дает регуляризованное решение. К таким методам относятся метод простой итерации, метод Ландвебера, метод сопряженных градиентов. Статистический подход основан на представлении задачи как фильтрации случайных процессов. Фильтр Винера минимизирует среднеквадратичную ошибку восстановления:

$$F_{Wiener}(\omega) = \frac{H^*(\omega) S_f(\omega)}{|H(\omega)|^2 S_f(\omega) + S_n(\omega)} \quad , \quad (16)$$

где H(ω) – передаточная функция,

$S_n(\omega)$ и $S^f(\omega)$ – спектральные плотности шума и сигнала.

Метод регуляризации Тихонова – наиболее известный подход к решению некорректных задач. Он основан на минимизации функционала

$$J_\alpha[f] = |Kf - g|^2 + \alpha |Lf|^2, \quad (17)$$

где $\alpha > 0$ – параметр регуляризации;

L – некоторый оператор.

Первое слагаемое отвечает за близость к исходным данным, второе – за регулярность (гладкость) решения.

Методы выбора параметра регуляризации приведены в таблице 7.

Таблица 7– Методы выбора параметра регуляризации

Метод	Принцип	Преимущества	Недостатки
Невязки	Согласование невязки с уровнем шума	Теоретическое обоснование	Требует оценки уровня шума
L-кривой	Анализ зависимости ‖f_α‖ от ‖Kf_α - g‖	Не требует оценки уровня шума	Не всегда дает четкий излом кривой
GCV	Минимизация ошибки предсказания	Хорошо работает для задач восстановления сигналов	Вычислительная сложность
Квазиоптимальности	Минимизация нормы разности решений для близких α	Простота реализации	Может давать неоптимальные результаты

Выбор оператора L зависит от априорной информации о решении: L = I – регуляризация нулевого порядка (минимизация нормы); L = d/dt – регуляризация первого порядка (минимизация производной); L = d^2/dt^2 – регуляризация второго порядка (минимизация кривизны). Для дискретного случая функционал Тихонова имеет вид:

$$J_\alpha[f] = |Kf - g|^2 + \alpha|Lf|^2 = (Kf - g)^T(Kf - g) + \alpha f^T L^T L f. \quad (18)$$

Минимизация приводит к системе $(K^T K + \alpha L^T L) f_\alpha = K^T g$.

В простейшем случае L = I система становится $(K^T K + \alpha I) f_\alpha = K^T g$. Для уравнений свертки можно использовать преобразование Фурье. Регуляризация Тихонова в частотной области [13]:

$$\hat{f}\alpha(\omega) = \frac{\hat{K^*}(\omega)\hat{g}(\omega)}{|\hat{K}(\omega)|^2 + \alpha|\hat{L}(\omega)|^2}. \quad (19)$$

Это выражение показывает, что метод работает как фильтр, подавляющий высокочастотные компоненты. Ключевая проблема – выбор параметра α. Малые значения дают неустойчивые решения, большие – сильно сглаженные. На практике используется принцип невязки: α выбирается так, чтобы

$$|Kf\alpha - g| = C\delta. \tag{20}$$

где δ – оценка погрешности правой части.

Другой подход – метод L-кривой, основанный на анализе зависимости ‖f_α‖ от ‖Kf_α - g‖. Для практических задач часто применяются упрощенные методы регуляризации – менее вычислительно затратные, чем полный метод Тихонова. Метод спектральной регуляризации вводит весовые коэффициенты, плавно подавляющие компоненты с малыми сингулярными числами:

$$f_\alpha = \sum_{i=1}^{\infty} \frac{\sigma_i^2}{\sigma_i^2+\alpha} \frac{(g,u_i)}{\sigma_i} v_i. \tag{21}$$

Это эквивалентно регуляризации Тихонова нулевого порядка. Фильтр Винера – статистический метод регуляризации, оптимальный при известных спектральных характеристиках сигнала и шума. При отсутствии точной информации используются оценки или упрощающие предположения. Метод квазирешений ищет решение, минимизирующее норму невязки ‖Kf - g‖ при ограничении ‖f‖ ≤ M. Итерационная регуляризация использует ранний останов итерационного процесса. Метод Ландвебера:

$$f_{k+1} = f_k + \tau K^T(g - Kf_k), \tag{22}$$

где τ – шаговый множитель (обычно $\tau < 2/\|K\|^2$).

Число итераций играет роль параметра регуляризации.

Таблица 8 – Упрощенные методы регуляризации

Метод	Принцип	Вычислительная сложность	Область применения
Спектральная регуляризация	Подавление компонент с малыми σ_i	Высокая (требует SVD)	Малоразмерные задачи
Фильтр Винера	Оптимальная фильтрация в частотной области	Низкая (для уравнений свертки)	Задачи с известными спектральными характеристиками
Метод квазирешений	Минимизация невязки при ограничении на норму решения	Средняя	Задачи с априорной оценкой нормы решения
Итерационная регуляризация	Ранний останов итерационного процесса	Низкая (для разреженных матриц)	Большеразмерные задачи

Эффективность методов регуляризации зависит от свойств задачи: степени некорректности, уровня шума, гладкости решения. Нет универсального метода для всех задач, поэтому практическое применение требует анализа специфики каждой конкретной задачи. Современные подходы включают нелинейные модели и методы оптимизации с ограничениями. Регуляризация с нормой l_1 дает разреженные решения (важно для сжатого считывания), полная вариация сохраняет резкие границы (полезно для изображений).

Интегральные уравнения Фредгольма первого рода представляют собой математический аппарат, естественным образом возникающий при решении обратных задач распространения сигналов. Данные уравнения относятся к классу некорректно поставленных задач, что создает существенные трудности при их численном решении.

Некорректность задачи проявляется в неустойчивости решения по отношению к малым изменениям исходных данных.

2 Разработка алгоритма решения задачи распространения сигналов

2.1 Математическая постановка задачи

Основу рассматриваемой задачи составляет линейный фильтр, преобразующий исходный сигнал f(t) в выходной сигнал g(t). Известно, что частотные спектры F(ω) и G(ω) функций f(t) и g(t) связаны соотношением

$G(\omega) = \rho(\omega)F(\omega)$, где $\rho(\omega) = (1 - |\omega|/\omega_0)\theta(\omega_0 - |\omega|)$ – частотная характеристика фильтра, а θ(ω) – функция Хевисайда. Частота отсечки ω_0 принадлежит интервалу [0.5, 5], а измерения сигнала проводятся в промежутке времени $[-1/\omega_0, 1/\omega_0]$ [14].

Такой фильтр можно классифицировать как фильтр низких частот (ФНЧ) с линейно убывающей частотной характеристикой. В отличие от идеального ФНЧ, имеющего прямоугольную АЧХ, рассматриваемый фильтр обеспечивает плавное подавление высокочастотных составляющих сигнала, что снижает эффект Гиббса при восстановлении сигнала.

Задача состоит в восстановлении исходного сигнала f(t) по известному выходному сигналу g(t) при заданной частотной характеристике фильтра ρ(ω). Эта задача относится к обратным задачам фильтрации и является некорректно поставленной, так как фильтр подавляет высокочастотные компоненты сигнала, и их восстановление требует специальных методов регуляризации.

Для решения задачи необходимо сначала вывести интегральное уравнение, связывающее сигналы f(t) и g(t) во временной области. Используя обратное преобразование Фурье, запишем выражение для выходного сигнала [15]:

$$g(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} G(\omega)e^{j\omega t}d\omega = \frac{1}{2\pi}\int_{-\infty}^{\infty} \rho(\omega)F(\omega)e^{j\omega t}d\omega \quad . \qquad (23)$$

Учитывая вид функции ρ(ω), интегрирование фактически производится по отрезку [-ω₀, ω₀], поскольку вне этого интервала подынтегральное выражение обращается в нуль из-за функции Хевисайда. Подставляя выражение для F(ω) через прямое преобразование Фурье от f(t), получаем:

$$g(t) = \frac{1}{2\pi}\int_{-\omega_0}^{\omega_0}\rho(\omega)\left(\int_{-\infty}^{\infty}f(\tau)e^{-j\omega\tau}d\tau\right)e^{j\omega t}d\omega. \quad (24)$$

Меняя порядок интегрирования (что допустимо при определенных условиях регулярности функции f(t)), приходим к выражению:

$$g(t) = \int_{-\infty}^{\infty}f(\tau)\left(\frac{1}{2\pi}\int_{-\omega_0}^{\omega_0}\rho(\omega)e^{j\omega(t-\tau)}d\omega\right)d\tau. \quad (25)$$

Внутренний интеграл представляет собой ядро интегрального уравнения K(t-τ):

$$K(t-\tau) = \frac{1}{2\pi}\int_{-\omega_0}^{\omega_0}\rho(\omega)e^{j\omega(t-\tau)}d\omega. \quad (26)$$

Вычислим это ядро, подставив выражение для ρ(ω):

$$K(t-\tau) = \frac{1}{2\pi}\int_{-\omega_0}^{\omega_0}\left(1-\frac{|\omega|}{\omega_0}\right)e^{j\omega(t-\tau)}d\omega. \quad (27)$$

Данный интеграл можно разбить на две части:

$$K(t-\tau) = \frac{1}{2\pi}\int_{-\omega_0}^{\omega_0}e^{j\omega(t-\tau)}d\omega - \frac{1}{2\pi\omega_0}\int_{-\omega_0}^{\omega_0}|\omega|e^{j\omega(t-\tau)}d. \quad (28)$$

Первый интеграл представляет собой функцию sinc, а второй может быть вычислен с использованием свойств преобразования Фурье. После вычислений получаем:

$$K(t-\tau) = \frac{sin\omega_0(t-\tau)}{\pi(t-\tau)} - \frac{cos\omega_0(t-\tau)-1}{\pi\omega_0(t-\tau)^2}. \quad (29)$$

Таким образом, задача сводится к интегральному уравнению Фредгольма первого рода:

$$g(t) = \int_{-\infty}^{\infty} K(t-\tau)f(\tau)d\tau. \quad (30)$$

Однако на практике интегрирование производится не по всей числовой прямой, а по конечному интервалу [-1/ω₀, 1/ω₀], что соответствует условиям проведения измерений. Выполнив замену переменных x = ω₀t, y = ω₀τ, приводим уравнение к виду:

$$g\left(\frac{x}{\omega_0}\right) = \frac{1}{\omega_0}\int_{-1}^{1} K\left(\frac{x-y}{\omega_0}\right) f\left(\frac{y}{\omega_0}\right) dy. \quad (31)$$

Введя обозначения G(x) = g(x/ω₀), F(y) = f(y/ω₀), получаем интегральное уравнение в стандартной форме:

$$G(x) = \int_{-1}^{1} K_{\omega_0}(x-y)F(y)dy, \quad (32)$$

где Kω₀(x-y) = K((x-y)/ω₀)/ω₀ – масштабированное ядро.

Полученное интегральное уравнение описывает связь между исходным сигналом F(y) и измеренным сигналом G(x) в нормированных координатах.

Задача теперь заключается в нахождении функции F(y) по известной функции G(x) при заданном ядре $K_{\omega_0}(x-y)$.

Данное интегральное уравнение является некорректно поставленной задачей в смысле Адамара, что связано с тем, что ядро $K_{\omega_0}(x-y)$ имеет сглаживающий характер (фильтр подавляет высокие частоты), поэтому малые изменения в функции G(x) (например, из-за шума измерений) могут приводить к большим изменениям в решении F(y). Для преодоления этой проблемы необходимо применять методы регуляризации.

При анализе граничных условий и ограничений задачи следует учитывать, что реальные сигналы f(t) и g(t) являются финитными функциями, то есть практически обращаются в нуль вне некоторого интервала. Обычно предполагается, что поддержка сигнала f(t) содержится в интервале измерений $[-1/\omega_0, 1/\omega_0]$. Это предположение физически обосновано, поскольку для корректного восстановления сигнала необходимо, чтобы его ненулевые значения находились в пределах интервала измерений.

Дополнительное ограничение связано с точностью измерений. Функция g(t) известна с относительной точностью ε по норме пространства L_1, то есть:

$$\frac{|g-g_\delta|_{L_1}}{|g|_{L_1}} \leq \epsilon, \tag{33}$$

где g_δ – измеренное значение сигнала,

g – его точное значение.

Погрешность измерений влияет на выбор параметров регуляризации и на определение порядка аппроксимации N при решении уравнения. При больших значениях ε требуется более сильная регуляризация, что приводит к сглаживанию решения и потере высокочастотных компонент. С другой стороны, при малых ε можно использовать более слабую регуляризацию, сохраняя больше деталей в восстановленном сигнале.

Частота отсечки ω_0 также влияет на свойства ядра интегрального уравнения. При больших значениях ω_0 фильтр пропускает более широкий

диапазон частот, что облегчает задачу восстановления сигнала. При малых значениях ω_0 фильтр сильнее подавляет высокие частоты, что делает задачу более некорректной и требует более тщательного выбора методов регуляризации.

2.2 Метод аппроксимации функций с использованием полиномов Лежандра

Для решения полученного интегрального уравнения Фредгольма первого рода применим метод аппроксимации функций с использованием полиномов Лежандра. Этот подход позволяет свести интегральное уравнение к системе линейных алгебраических уравнений (СЛАУ). Полиномы Лежандра $P_n(x)$ образуют ортогональную систему на отрезке [-1, 1] с весом 1. Они определяются формулой Родрига [19]:

$$P_n(x) = \frac{1}{2^n n!}\frac{d^n}{dx^n}[(x^2-1)^n]. \quad (34)$$

и удовлетворяют соотношению ортогональности:

$$\int_{-1}^{1} P_m(x)P_n(x)dx = \frac{2}{2n+1}\delta_{mn} \quad . \quad (35)$$

где δmn – символ Кронекера. Первые несколько полиномов Лежандра имеют вид:

$$P_0(x) = 1\ P_1(x) = x\ P_2(x) = (3x^2 - 1)/2\ P_3(x) = (5x^3 - 3x)/2 \quad (36)$$

Теоретическое обоснование метода аппроксимации с использованием полиномов Лежандра базируется на том факте, что любую достаточно

гладкую функцию на отрезке [-1, 1] можно разложить в ряд по полиномам Лежандра [15]:

$$f(x) = \sum_{n=0}^{\infty} f^{(n)} P_n(x). \quad (37)$$

Где коэффициенты разложения f^(n) определяются как:

$$f^{(n)} = \frac{2n+1}{2} \int_{-1}^{1} f(x) P_n(x) dx. \quad (38)$$

В нашей задаче функции G(x) и F(y) можно представить в виде конечных сумм:

$$G(x) \approx \sum_{n=0}^{N} g^{(n)} P_n(x), F(y) \approx \sum_{m=0}^{N} f^{(m)} P_m(y). \quad (39)$$

где N – порядок аппроксимации, который определяется требуемой точностью ε.

Подставляя эти разложения в интегральное уравнение, получаем:

$$\sum_{n=0}^{N} g^{(n)} P_n(x) = \int_{-1}^{1} K_{\omega_0}(x-y) \sum_{m=0}^{N} f^{(m)} P_m(y) dy. \quad (40)$$

Меняя порядок суммирования и интегрирования, имеем:

$$\sum_{n=0}^{N} g^{(n)} P_n(x) = \sum_{m=0}^{N} f^{(m)} \int_{-1}^{1} K_{\omega_0}(x-y) P_m(y) dy. \quad (41)$$

Обозначим интеграл через Km(x):

$$K_m(x) = \int_{-1}^{1} K_{\omega_0}(x-y) P_m(y) dy. \quad (41)$$

Функцию K_m(x) также можно разложить по полиномам Лежандра:

$$K_m(x) = \sum_{n=0}^{N} k_{nm} P_n(x). \quad (42)$$

Приравнивая коэффициенты при одинаковых полиномах Лежандра Pn(x), получаем систему линейных алгебраических уравнений:

$$g^{(n)} = \sum_{m=0}^{N} k_{nm} f^{(m)}, n = 0,1,\ldots,N. \quad (43)$$

Или в матричной форме:

$$g = Kf, \quad (44)$$

где $g = (g(0), g(1), \ldots, g(N))^T, f = (f(0), f(1), \ldots, f(N))^T$

K – матрица с элементами knm.

Особенности применения полиномов Лежандра в данной задаче связаны с несколькими факторами. Во-первых, область интегрирования в задаче [-1, 1] точно совпадает с областью определения полиномов Лежандра, что делает их естественным выбором для аппроксимации. Во-вторых, ортогональность полиномов Лежандра упрощает вычисление коэффициентов разложения и уменьшает вычислительную сложность алгоритма. Важной особенностью является также то, что при использовании полиномов Лежандра коэффициенты разложения имеют ясный физический смысл.

Например, коэффициент $g(0)$представляет собой среднее значение функции G(x) на отрезке [-1, 1], коэффициент $g(1)$связан с линейным трендом, коэффициент $g(2)$– с квадратичной составляющей и т. д. Это позволяет интерпретировать результаты разложения и анализировать свойства сигнала. Еще одним преимуществом использования полиномов Лежандра является то, что они образуют базис в пространстве $L^2([-1, 1])$, что обеспечивает сходимость разложения для широкого класса функций. Для непрерывных

функций ряд сходится равномерно, а для функций из L^2 – в среднеквадратичном смысле. Определение порядка аппроксимации N в зависимости от точности ε является критически важным этапом. С одной стороны, N должно быть достаточно большим, чтобы обеспечить требуемую точность аппроксимации. С другой стороны, слишком большие значения N могут приводить к неустойчивости решения из-за усиления влияния погрешностей в исходных данных. Для определения оптимального значения N можно использовать следующий подход. Поскольку функция G(x) известна с относительной точностью ε, то имеет смысл выбирать такое N, чтобы погрешность аппроксимации не превышала погрешности измерений. Для оценки погрешности аппроксимации используется соотношение [20]:

$$|G - G_N|_{L_2} \leq C \cdot \omega_N(G), \tag{45}$$

где GN – аппроксимация функции G полиномами Лежандра порядка N,

C – константа, не зависящая от G

ωN(G) – модуль непрерывности N-го порядка функции G.

Для достаточно гладких функций ωN(G) быстро убывает с ростом N, что позволяет достичь высокой точности аппроксимации при умеренных значениях N. Практически порядок аппроксимации N можно определить, исходя из условия [39]:

$$\frac{|G-G_N|L_2}{|G|L_2} \leq \epsilon. \tag{46}$$

Для этого строится последовательность аппроксимаций G1, G2, ..., GN и вычисляется относительная погрешность каждой аппроксимации. Когда относительная погрешность становится меньше или равной ε, соответствующее значение N принимается за порядок аппроксимации. С учетом возможной нестабильности решения интегрального уравнения,

рекомендуется не использовать слишком большие значения N, даже если это позволяет формально уменьшить погрешность аппроксимации. Альтернативный подход к определению N основан на анализе спектра сигнала. Если известна частота отсечки ω_0, то можно оценить максимальную частоту в спектре сигнала и, используя теорему отсчетов, определить необходимое количество точек для представления сигнала без потери информации. Порядок аппроксимации N должен быть сравним с этим количеством. При численной реализации метода важно учитывать, что погрешности вычисления коэффициентов k_nm могут существенно влиять на точность решения. Для уменьшения этого эффекта рекомендуется использовать формулы Гаусса-Лежандра для численного интегрирования, которые обеспечивают высокую точность при относительно небольшом числе узлов.

2.3 Блок-схема алгоритма решения задачи

Алгоритм решения задачи распространения сигналов включает несколько основных этапов (рисунок 2.1): предварительную обработку данных, формирование матрицы СЛАУ, регуляризацию и решение системы, восстановление сигнала по найденным коэффициентам разложения. Рассмотрим эти этапы более подробно.

Этап 1: Предварительная обработка данных. На этом этапе производится нормализация временной шкалы, то есть переход от физического времени t к нормированной переменной $x = \omega_0 t$. Также выполняется масштабирование значений сигнала для улучшения численной устойчивости расчетов. Если исходные данные представлены в дискретной форме (набор отсчетов), то на этом этапе может производиться их интерполяция гладкими функциями для последующего вычисления коэффициентов разложения по полиномам Лежандра.

На рисунке 1 привежена блок-схема алгоритма решения задачи распространения сигналов.

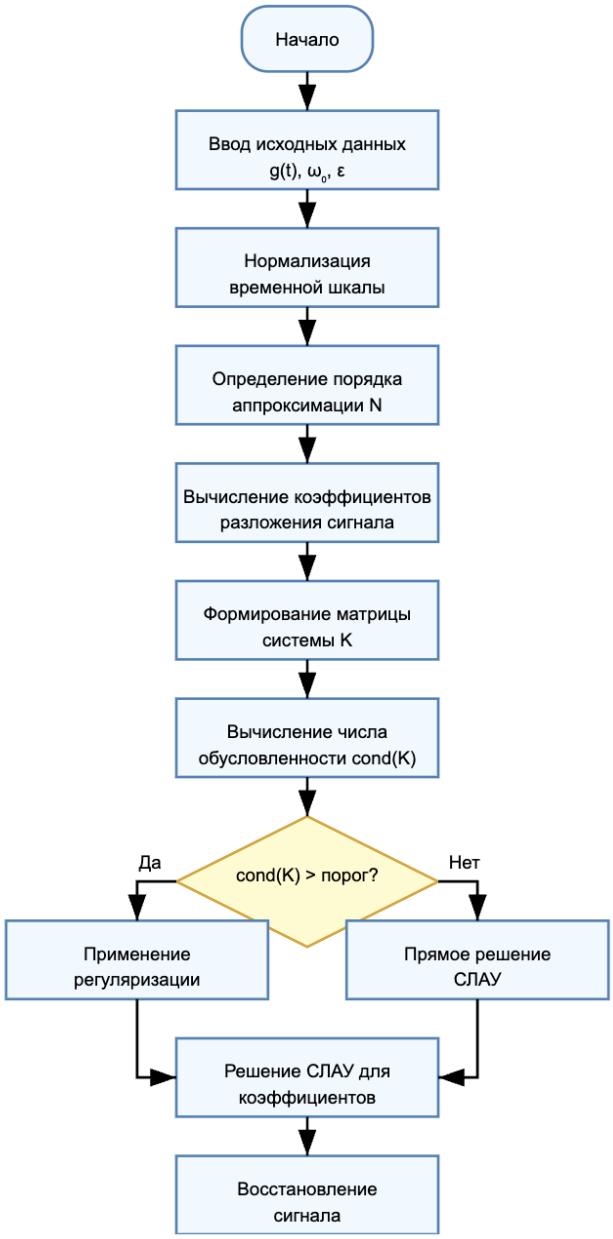

Рисунок 1 – Блок-схема алгоритма решения задачи распространения сигналов

Этап 2: Вычисление коэффициентов разложения измеренного сигнала. Коэффициенты g^(n) разложения функции G(x) по полиномам Лежандра вычисляются по формуле:

$$g^{(n)} = \frac{2n+1}{2}\int_{-1}^{1} G(x)P_n(x)dx. \quad (47)$$

Для численного интегрирования используется квадратурная формула Гаусса-Лежандра [14]:

$$g^{(n)} \approx \frac{2n+1}{2}\sum_{i=1}^{M} w_i G(x_i)P_n(x_i), \quad (48)$$

где x_i– узлы квадратурной формулы Гаусса-Лежандра,

wi – соответствующие веса, а M – количество узлов.

Выбор M зависит от требуемой точности интегрирования и гладкости функции G(x). Обычно достаточно взять M = 2N + 1, где N – порядок аппроксимации.

Этап 3: Формирование матрицы СЛАУ. Элементы матрицы K вычисляются по формуле [17]:

$$k_{nm} = \frac{2n+1}{2}\int_{-1}^{1}\int_{-1}^{1} K_{\omega_0}(x-y)P_m(y)P_n(x)dydx. \quad (49)$$

Этот двойной интеграл может быть вычислен различными способами. Один из подходов – использование тензорного произведения квадратурных формул Гаусса-Лежандра для двумерного интегрирования. Другой подход основан на аналитическом вычислении внутреннего интеграла и последующем численном интегрировании полученного выражения. Для ускорения расчетов можно использовать свойства ядра Kω₀(x-y), которое зависит только от разности аргументов. Это позволяет применить быстрые алгоритмы вычисления свертки [16].

Этап 4: Анализ числа обусловленности матрицы СЛАУ. Число обусловленности cond(K) = $\|K\| \cdot \|K^{-1}\|$ характеризует устойчивость решения СЛАУ по отношению к малым изменениям правой части. Большие значения cond(K) (например, 10^6 и более) указывают на плохую обусловленность системы и необходимость применения методов регуляризации. Для вычисления числа обусловленности можно использовать сингулярное разложение матрицы (SVD) [7]:

$$K = USV^T, \tag{50}$$

где U и V – ортогональные матрицы,

S – диагональная матрица сингулярных чисел $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma\_N > 0$.

Число обусловленности вычисляется как отношение наибольшего сингулярного числа к наименьшему: cond(K) = σ_1/σ_N. Анализ сингулярных чисел матрицы K также позволяет оценить степень некорректности задачи. Быстрое убывание сингулярных чисел указывает на сильную некорректность и необходимость более жесткой регуляризации [7].

Этап 5: Регуляризация и решение СЛАУ. В зависимости от результатов анализа числа обусловленности матрицы K выбирается метод регуляризации. Для умеренно некорректных задач может быть достаточно простой регуляризации Тихонова нулевого порядка, которая сводится к решению модифицированной системы:

$$(K^T K + \alpha I)f = K^T g. \tag{51}$$

Параметр регуляризации α выбирается на основе принципа невязки или L-кривой. При использовании сингулярного разложения матрицы K решение можно записать в виде [10]:

$$f = \sum_{i=1}^{N} \frac{\sigma_i}{\sigma_i^2+\alpha} (U^T g)_i v_i \quad (52)$$

где v_i– столбцы матрицы V.

При α = 0 это выражение дает псевдообратное решение, которое может быть неустойчивым при малых σ_i. Введение ненулевого параметра α стабилизирует решение, подавляя вклад компонент, соответствующих малым сингулярным числам. Альтернативой регуляризации Тихонова является метод усеченного сингулярного разложения (TSVD), в котором используются только первые k компонент сингулярного разложения:

$$f = \sum_{i=1}^{k} \frac{(U^T g)_i}{\sigma_i} v_i \quad . \quad (53)$$

Где k < N выбирается так, чтобы σ_kбыло достаточно велико для обеспечения устойчивости решения. Этап 6: Восстановление сигнала по найденным коэффициентам. По найденному вектору коэффициентов $f = (f(0), f(1), \ldots, f(N))^T$восстанавливается функция F(y) [12]:

$$F(y) = \sum_{m=0}^{N} f^{(m)} P_m(y). \quad (54)$$

Затем производится обратный переход к исходным переменным:

$$f(t) = F(\omega_0 t) \quad (55)$$

На этом этапе также могут применяться дополнительные методы обработки, например, сглаживание восстановленного сигнала для уменьшения осцилляций, связанных с ошибками аппроксимации. Этап 7: Оценка точности решения. Для оценки точности решения вычисляется невязка [12]:

$$|g - Kf| \tag{56}$$

Если невязка превышает заданный порог (определяемый точностью измерений ε), то может потребоваться перерасчет с другими параметрами регуляризации или с другим порядком аппроксимации N. Схема сведения интегрального уравнения к СЛАУ основана на представлении функций G(x) и F(y) в виде конечных сумм по полиномам Лежандра и последующем приравнивании коэффициентов при одинаковых полиномах - этот подход имеет ряд преимуществ перед прямой дискретизацией интегрального уравнения методом квадратур. Прежде всего, использование полиномов

Лежандра позволяет учесть глобальное поведение функций, а не только их значения в отдельных точках. Это особенно важно для гладких функций, которые хорошо аппроксимируются небольшим числом полиномов. Кроме того, порядок СЛАУ N обычно значительно меньше, чем количество точек дискретизации, необходимое для достижения той же точности методом квадратур - это уменьшает вычислительные затраты и повышает устойчивость решения. Однако следует отметить, что эффективность данного метода существенно зависит от свойств функций G(x) и F(y). Если эти функции имеют разрывы или быстро осциллируют, то их аппроксимация полиномами Лежандра может требовать очень высокого порядка N, что нивелирует преимущества метода.

Методика анализа числа обусловленности матрицы играет ключевую роль в обеспечении устойчивости решения. Как уже отмечалось, число обусловленности cond(K) характеризует чувствительность решения СЛАУ к малым изменениям правой части. Для плохо обусловленных систем (с

большим числом обусловленности) даже небольшие погрешности в правой части могут приводить к значительным ошибкам в решении [13].

Анализ числа обусловленности позволяет не только выбрать подходящий метод регуляризации, но и оценить необходимую степень регуляризации. Чем больше число обусловленности, тем сильнее должна быть регуляризация для обеспечения устойчивости решения.

Для систем с очень большим числом обусловленности (10^{12} и более) стандартные методы регуляризации могут оказаться недостаточно эффективными. В этом случае может потребоваться уменьшение порядка аппроксимации N или использование специальных методов, учитывающих структуру матрицы K.

Во второй главе разработан алгоритм решения задачи распространения сигналов на основе интегрального уравнения Фредгольма первого рода с использованием метода аппроксимации функций полиномами Лежандра. Для линейного фильтра с характеристикой $\rho(\omega) = (1 - |\omega|/\omega_0)\theta(\omega_0 - |\omega|)$ получено интегральное уравнение, связывающее выходной сигнал g(t) с входным сигналом f(t). Путем нормализации временной шкалы уравнение приведено к стандартной форме с интегрированием на отрезке [-1, 1].

Использование полиномов Лежандра для аппроксимации функций позволяет эффективно свести интегральное уравнение к системе линейных алгебраических уравнений. Этот подход имеет существенные преимущества перед методом прямой дискретизации, особенно для гладких функций. Порядок аппроксимации N должен выбираться в зависимости от требуемой точности ε и свойств решаемой задачи. Слишком большие значения N могут привести к неустойчивости решения из-за усиления влияния погрешностей в исходных данных [13].

3 Разработка алгоритма решения задачи распространения сигналов

3.1 Разработка программного комплекса

При разработке программного комплекса для решения задачи распространения сигналов важно обеспечить сочетание вычислительной эффективности, удобства использования и возможности визуализации результатов. Данный раздел посвящен описанию технических и архитектурных решений, принятых при реализации программного комплекса.

Для реализации алгоритма решения интегрального уравнения Фредгольма первого рода выбран язык программирования Python версии 3.9. Этот выбор обусловлен рядом преимуществ, определяющих эффективность разработки и использования программного комплекса. Python обладает простым и понятным синтаксисом, что упрощает разработку и последующую модификацию программы. Обширная экосистема научных библиотек – ключевой фактор выбора этого языка для решения математических задач. Библиотеки NumPy и SciPy предоставляют эффективные инструменты для работы с матрицами, решения систем линейных уравнений и численного интегрирования, что критически важно для реализации нашего алгоритма. Для построения графиков и визуализации результатов используется библиотека Matplotlib, позволяющая создавать высококачественные графики различных типов. Библиотека Pandas применяется для структурированного хранения и обработки данных, а также для удобства сохранения и загрузки результатов [1].

Основные библиотеки Python, использованные в разработке приведены в таблице 9.

Таблица 9 – Основные библиотеки Python, использованные в разработке

Библиотека	Версия	Назначение
NumPy	1.21.5	Матричные вычисления, линейная алгебра
SciPy	1.7.3	Специальные функции, интегрирование, регуляризация
Matplotlib	3.5.1	Визуализация данных и построение графиков
Pandas	1.4.2	Структурированное хранение данных
PyQt5	5.15.6	Создание графического интерфейса пользователя

В качестве интегрированной среды разработки (IDE) выбран PyCharm Professional Edition 2022.1. Данная среда предоставляет удобные инструменты для разработки, отладки и профилирования Python-приложений. Встроенная поддержка виртуальных окружений позволяет изолировать зависимости проекта, что обеспечивает его переносимость и упрощает развертывание. Для контроля версий программного кода использовался Git с размещением репозитория на GitHub. Это обеспечивает надежное хранение кода, возможность отслеживания изменений и восстановления предыдущих версий при необходимости [2].

Программный комплекс разработан с использованием принципов объектно-ориентированного программирования и модульной архитектуры. Такой подход обеспечивает четкое разделение функциональности, упрощает поддержку и расширение программы, а также позволяет повторно использовать компоненты в других проектах. Структура программного комплекса включает следующие основные модули [22]:

– core – ядро программы, содержащее базовую функциональность для работы с интегральными уравнениями;

– numerical – реализация численных методов, необходимых для решения задачи;

– signal – модули для генерации, анализа и обработки сигналов;

– gui – компоненты графического пользовательского интерфейса;

– utils – вспомогательные функции и утилиты.

Схема взаимодействия основных модулей программного комплекса представлена на рисунке 2.

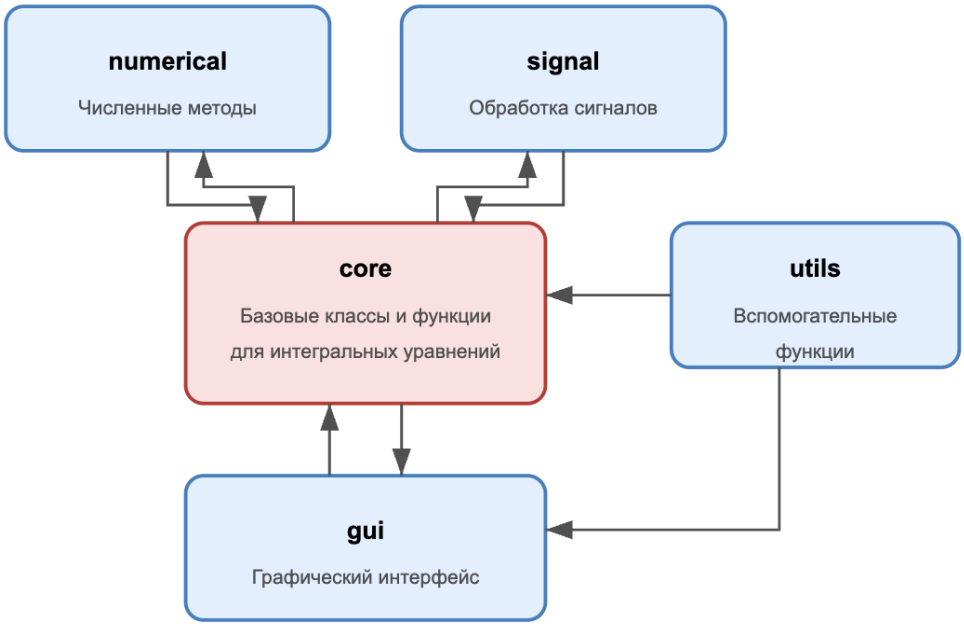

Рисунок 2 – Схема взаимодействия модулей программного комплекса

Модуль core содержит базовые классы и функции для работы с интегральными уравнениями. Ключевым компонентом является класс FredholmEquation, который инкапсулирует математическую модель уравнения Фредгольма первого рода с заданным ядром. Этот класс предоставляет методы для формирования матрицы системы линейных алгебраических уравнений и анализа её свойств [23].

```
class FredholmEquation:
    def __init__(self, kernel_function, domain=(-1, 1)):
self.kernel = kernel_function
```

```
        self.domain = domain
    def create_matrix(self, n_points):
        # Формированиематрицысистемы
        ...

    def analyze_condition_number(self):
# Анализ числа обусловленности
```

Модуль numerical реализует численные методы, необходимые для решения задачи. Он включает классы для работы с полиномами Лежандра, вычисления коэффициентов разложения функций, численного интегрирования и реализации методов регуляризации. Важную роль играет класс LegendreApproximation, который обеспечивает аппроксимацию функций с использованием полиномов Лежандра.

Модуль signal предоставляет функциональность для работы с сигналами. Он включает классы и функции для генерации тестовых сигналов, фильтрации, анализа спектральных характеристик и визуализации. Класс SignalProcessor инкапсулирует методы обработки сигналов и является связующим звеном между математической моделью и представлением результатов.

Модуль gui содержит компоненты графического пользовательского интерфейса, реализованные с использованием библиотеки PyQt5. Он включает классы для основного окна приложения, диалоговых окон для ввода параметров, визуализации результатов и настройки программы.

Модуль utils содержит вспомогательные функции и утилиты, используемые в различных частях программы. Это включает функции для работы с файлами, логирования, обработки ошибок и профилирования производительности.

Преимуществом такой модульной структуры является возможность независимого развития и тестирования отдельных компонентов программного комплекса. Например, функциональность модуля numerical может быть

использована без GUI в скриптах для пакетной обработки данных или интегрирована в другие проекты.

Реализация алгоритма решения задачи распространения сигналов следует методологии, описанной в главе 2, с учетом особенностей языка Python и используемых библиотек. Рассмотрим ключевые аспекты реализации основных этапов алгоритма.

Полиномы Лежандра играют центральную роль в методе аппроксимации функций. Для их вычисления использована рекуррентная формула [28]:

```
def legendre_polynomial(n, x):
if n == 0:
    return np.ones_like(x)
  elif n == 1:
    return x
  else:
    p_prev = np.ones_like(x)
    p_curr = x.copy()
    for i in range(2, n + 1):
      p_next = ((2*i - 1) * x * p_curr - (i - 1) * p_prev) / i
      p_prev, p_curr = p_curr, p_next
return p_curr
```

Для повышения эффективности вычислений при больших значениях аргумента или при высоких порядках полиномов используется библиотека SciPy, предоставляющая оптимизированную реализацию функций Лежандра:

```
from scipy.special import legendre
def legendre_poly_efficient(n, x):
  leg = legendre(n)
  return leg(x)
```

Формирование матрицы СЛАУ

Формирование матрицы системы линейных алгебраических уравнений – критический этап алгоритма с точки зрения производительности. Для повышения эффективности используется векторизация операций с помощью NumPy:

```
def create_matrix(self, n):
    """Формирование матрицы системы порядка n."""
matrix = np.zeros((n+1, n+1))
x = np.linspace(-1, 1, 100)  # Точки для численного интегрирования
dx = x[1] - x[0]
    for i in range(n+1):
        for j in range(n+1):
# Вычисление элемента матрицы K[i,j]
integrand = np.zeros_like(x)
            for idx, xi in enumerate(x):
                for idy, y in enumerate(x):
                    k_val = self.kernel(xi - y)
                    p_i = legendre_polynomial(i, xi)
                    p_j = legendre_polynomial(j, y)
                    integrand[idx] += k_val * p_i * p_j * dx
matrix[i, j] = (2*i + 1) / 2 * np.sum(integrand) * dx
return matrix
```

Для ядер специального вида (например, для ядер свертки) используются более эффективные алгоритмы, основанные на свойствах полиномов Лежандра.

Для анализа числа обусловленности матрицы СЛАУ используется сингулярное разложение (SVD), реализованное в NumPy:

```
def analyze_condition_number(self, matrix):
"""Анализ числа обусловленности матрицы."""
    u, s, vh = np.linalg.svd(matrix)
condition_number = s[0] / s[-1]
```

```
    return condition_number, s
```

Регуляризация Тихонова нулевого порядка реализована с использованием модификации нормальной системы уравнений:

```
def tikhonov_regularization(self, matrix, rhs, alpha):
"""Регуляризация Тихонова нулевого порядка."""
    m_t = matrix.T
reg_matrix = m_t @ matrix + alpha * np.eye(matrix.shape[1])
    reg_rhs = m_t @ rhs
    return reg_matrix, reg_rhs
```

Для выбора параметра регуляризации реализован метод L-кривой, который анализирует зависимость нормы решения от нормы невязки при различных значениях параметра регуляризации:

```
def l_curve_method(self, matrix, rhs, alphas):
"""Метод L-кривой для выбора параметра регуляризации."""
solution_norms = []
    residual_norms = []
    for alpha in alphas:
        reg_matrix, reg_rhs = self.tikhonov_regularization(matrix, rhs, alpha)
        solution = np.linalg.solve(reg_matrix, reg_rhs)
        solution_norm = np.linalg.norm(solution)
        residual_norm = np.linalg.norm(matrix @ solution - rhs)
        solution_norms.append(solution_norm)
        residual_norms.append(residual_norm)
# Анализ кривизны L-кривой для выбора оптимального alpha
return optimal_alpha, solution_norms, residual_norms
```

Восстановление сигнала

После решения СЛАУ и получения коэффициентов разложения f^(m) выполняется восстановление сигнала F(y) по формуле:

```
def reconstruct_signal(self, coefficients, x):
"""Восстановление сигнала по коэффициентам разложения."""
```

```
signal = np.zeros_like(x)
    for m, coef in enumerate(coefficients):
        signal += coef * legendre_polynomial(m, x)
return signal
```

На рисунке 3 приведена схема алгоритма восстановления сигнала.

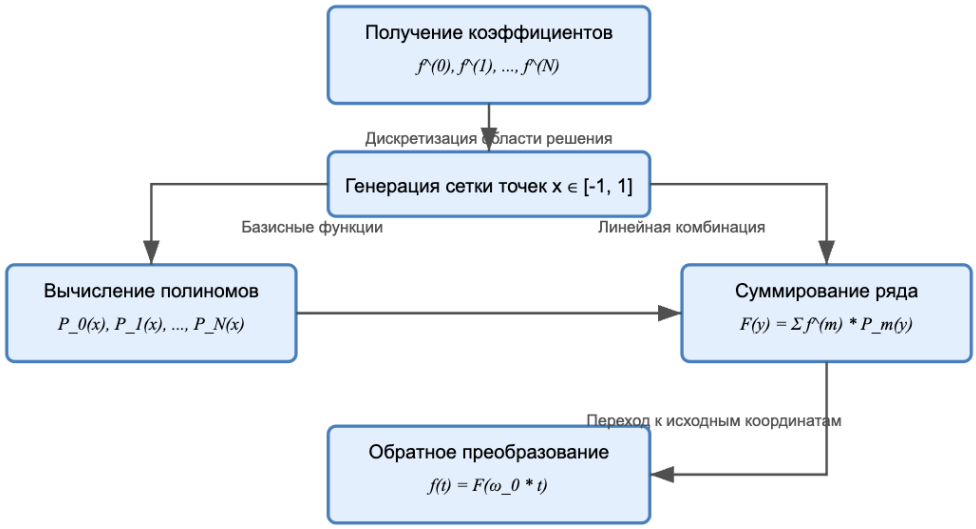

Рисунок 3 – Схема алгоритма восстановления сигнала

Для повышения производительности при восстановлении сигнала на больших наборах данных используется векторизация операций и предварительное вычисление полиномов Лежандра. Графический пользовательский интерфейс (GUI) программного комплекса разработан с использованием библиотеки PyQt5. Интерфейс обеспечивает удобный ввод параметров задачи, визуализацию результатов и управление процессом вычислений.

Основное окно приложения содержит следующие элементы:

– панель ввода параметров (частота отсечки ω_0, точность ε, параметр регуляризации);

– область визуализации сигналов (входной, выходной, восстановленный);

– область отображения спектров сигналов;

– панель управления процессом вычислений;

– информационную панель с результатами анализа.

Главное окно программного комплекса приведена на рисунке 4.

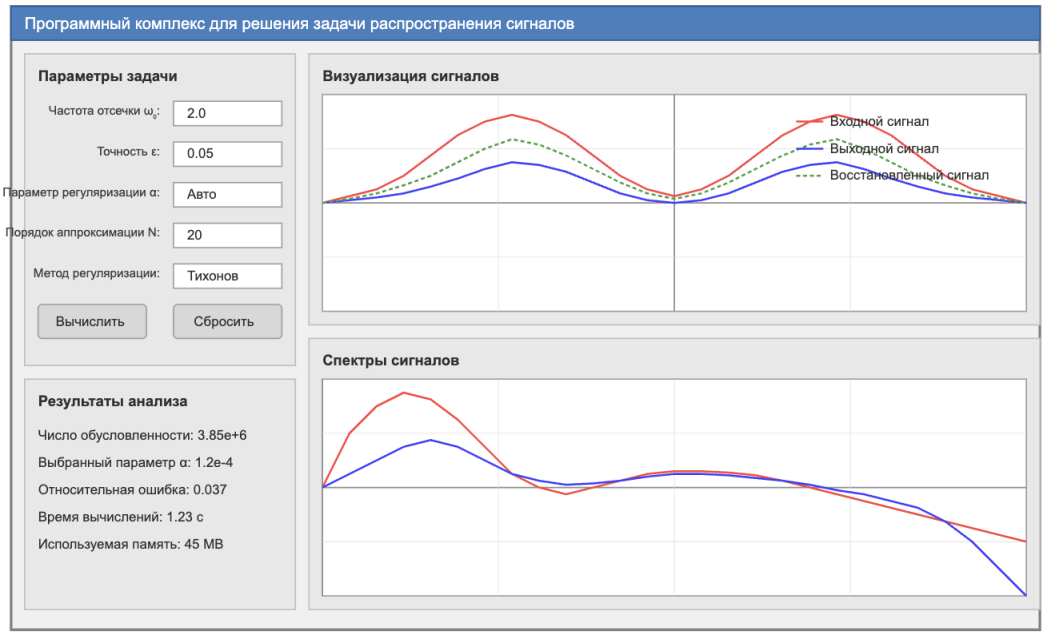

Рисунок 4 – Главное окно программного комплекса

Для ввода параметров задачи реализованы специализированные диалоговые окна, обеспечивающие валидацию вводимых значений и предоставляющие пользователю подсказки для правильного выбора параметров.

Ввод частоты отсечки ω_0 ограничен интервалом [0.5, 5] согласно условиям задачи. Для точности ε предусмотрен выбор из диапазона 1%-10% с возможностью ввода произвольного значения. Параметр регуляризации может быть выбран вручную или определен автоматически с использованием метода L-кривой или принципа невязки.

Параметры программного комплекса и их значения по умолчанию приведены в таблице 10.

Таблица 10 – Параметры программного комплекса и их значения по умолчанию

Параметр	Значение по умолчанию	Диапазон допустимых значений
Частота отсечки ω_0	2.0	[0.5, 5.0]
Точность ε	0.05 (5%)	[0.01, 0.1]
Параметр регуляризации α	Авто	[1e-8, 1e-1] или Авто
Порядок аппроксимации N	Авто	[5, 50] или Авто
Метод регуляризации	Тихонов	Тихонов, TSVD, Ландвебер

Для удобства работы с программой реализована возможность загрузки и сохранения параметров в файл конфигурации формата JSON. Это позволяет сохранять настройки между сеансами работы и обмениваться конфигурациями между пользователями.

```
def save_configuration(self, filename):
"""Сохранение конфигурации в JSON-файл."""
config = {
    'omega_0': self.omega_0,
    'epsilon': self.epsilon,
    'alpha': self.alpha if self.manual_alpha else 'auto',
    'n_order': self.n_order if self.manual_n_order else 'auto',
    'regularization_method': self.regularization_method
  }
  with open(filename, 'w') as f:
json.dump(config, f, indent=4)
```

Для визуализации результатов в программном комплексе используется библиотека Matplotlib, интегрированная в графический интерфейс PyQt5. Это

позволяет создавать интерактивные графики с возможностью масштабирования, сохранения в различных форматах и настройки параметров отображения.

Программный комплекс поддерживает несколько режимов работы. Интерактивный режим – пользователь задает параметры через GUI и получает результаты в реальном времени. Пакетный режим – обработка нескольких наборов данных с различными параметрами без вмешательства пользователя. Исследовательский режим – автоматический анализ влияния различных параметров на результаты с построением графиков зависимостей.

Диалоговое окно настройки параметров регуляризации приведена в таблице 5.

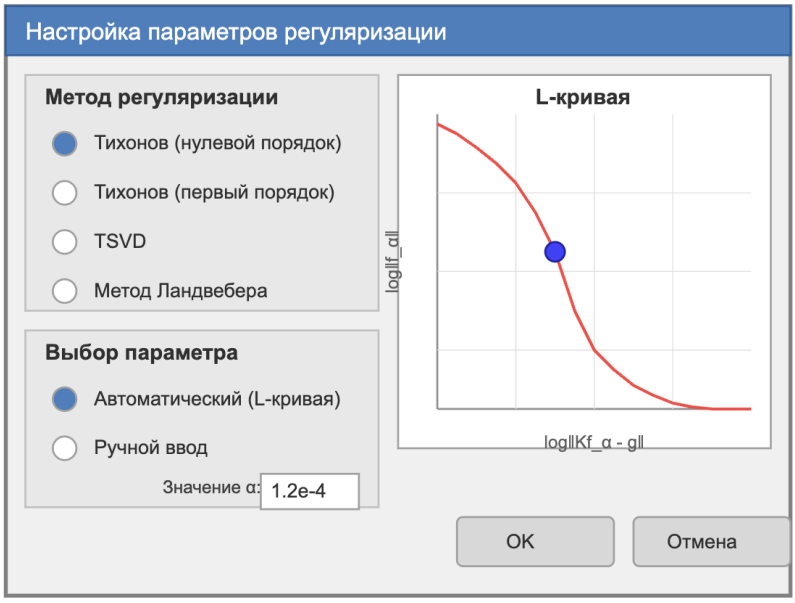

Рисунок 5 – Диалоговое окно настройки параметров регуляризации

Для удобства анализа результатов предусмотрена возможность экспорта данных в различные форматы (CSV, Excel, JSON), а также сохранения графиков в виде изображений или в векторных форматах (SVG, PDF).

Таким образом, разработанный программный комплекс представляет собой эффективный инструмент для решения задачи распространения

сигналов через линейный фильтр с заданной частотной характеристикой. Модульная архитектура, оптимизированные алгоритмы и удобный пользовательский интерфейс обеспечивают гибкость, производительность и удобство использования.

3.2 Построение и анализ решений

Восстановление исходного сигнала после решения интегрального уравнения Фредгольма первого рода требует особого внимания к вопросам точности и устойчивости вычислений. В данном разделе рассматриваются методические аспекты построения функции f(t) по найденному вектору коэффициентов, а также анализируются полученные решения с точки зрения их точности и соответствия исходной задаче.

После решения системы линейных алгебраических уравнений и получения вектора коэффициентов f(n) = (f^(0), f^(1), ..., f^(N))^T необходимо выполнить восстановление функции f(t) для произвольных значений аргумента t. Эта задача решается в несколько этапов [26].

Первый этап – восстановление функции F(y) в нормированных координатах на отрезке [-1, 1] с использованием разложения по полиномам Лежандра:

$$F(y) = \sum_{m=0}^{N} f^{(m)} \cdot P_m(y). \quad (57)$$

Коэффициенты f^(m) уже известны из решения СЛАУ, а полиномы Лежандра Pm(y) вычисляются по рекуррентной формуле либо с использованием специализированных функций библиотеки SciPy. Суммирование ряда выполняется для каждой точки y из выбранной сетки дискретизации.

На практике для получения плавной кривой используется достаточно мелкая сетка точек, например:

```
y_values = np.linspace(-1, 1, 1000)  # 1000 точекнаотрезке [-1, 1]
F_values = np.zeros_like(y_values)
for m in range(N+1):
    F_values += coefficients[m] * legendre_polynomial(m, y_values)
```

Второй этап – переход от нормированных координат к исходным физическим переменным. Для этого используется соотношение:

$$f(t) = F(\omega_0 t) , \tag{58}$$

где ω_0 – частота отсечки фильтра.

Таким образом, график функции f(t) получается из графика F(y) путем масштабирования оси абсцисс.

Для анализа точности восстановления используется несколько подходов. Один из них – вычисление относительной погрешности в норме L_2: $\delta = ||f\_approx - f\_exact||_2 / ||f\_exact||_2$

где f_approx – восстановленный сигнал, а f_exact – точное решение (если оно известно, например, в тестовых задачах).

Другой подход – анализ невязки: $\eta = ||g - Kf\_approx||_2 / ||g||_2$

где g – известный выходной сигнал, K – интегральный оператор, а f_approx – восстановленный входной сигнал.

При тестировании алгоритма использовались различные модельные функции, для которых известно точное решение интегрального уравнения. Это позволило оценить точность и устойчивость метода в зависимости от

различных параметров (порядка аппроксимации N, частоты отсечки ω_0, уровня шума).

Таблица 11 – Относительные погрешности восстановления различных тестовых сигналов

Тестовый сигнал	N=10	N=20	N=30	N=40
Гауссиан	0.057	0.018	0.012	0.011
Прямоугольный импульс	0.123	0.089	0.074	0.068
Линейная функция	0.008	0.003	0.003	0.003
Синусоида	0.032	0.011	0.009	0.009

Как видно из таблицы, для гладких функций (гауссиан, синусоида, линейная функция) достаточно 20-30 членов разложения для достижения высокой точности. Для функций с разрывами (прямоугольный импульс) погрешность остается заметной даже при большом числе членов ряда из-за эффекта Гиббса.

При практическом восстановлении сигналов важно учитывать следующие аспекты. Выбор достаточного порядка аппроксимации N с учетом гладкости сигнала и требуемой точности. Контроль устойчивости восстановления, особенно при наличии шума в исходных данных. Возможное применение дополнительной фильтрации восстановленного сигнала для сглаживания нежелательных осцилляций.

Аналитическое представление решения в виде разложения по полиномам Лежандра имеет вид:

$$f(t) = \sum_{m=0}^{N} f^{(m)} \cdot P_m(\omega_0 t). \quad (59)$$

Такая форма представления компактна и информативна, поскольку позволяет анализировать структуру решения через спектр коэффициентов f^(m). Быстрое убывание коэффициентов с ростом m свидетельствует о

гладкости функции, в то время как медленное убывание указывает на наличие высокочастотных компонент или разрывов.

Графическое представление решений может включать несколько типов визуализации:

– график восстановленного сигнала f(t) в сравнении с точным решением (если оно известно);

– график выходного сигнала g(t) и результата прямого преобразования восстановленного сигнала (Kf_approx);

– спектр коэффициентов разложения f^(m), позволяющий оценить скорость сходимости ряда;

– графики погрешности и невязки в зависимости от порядка аппроксимации N.

На рисунке 6 представлен пример восстановления модельного сигнала (гауссиана) при различных значениях порядка аппроксимации N.

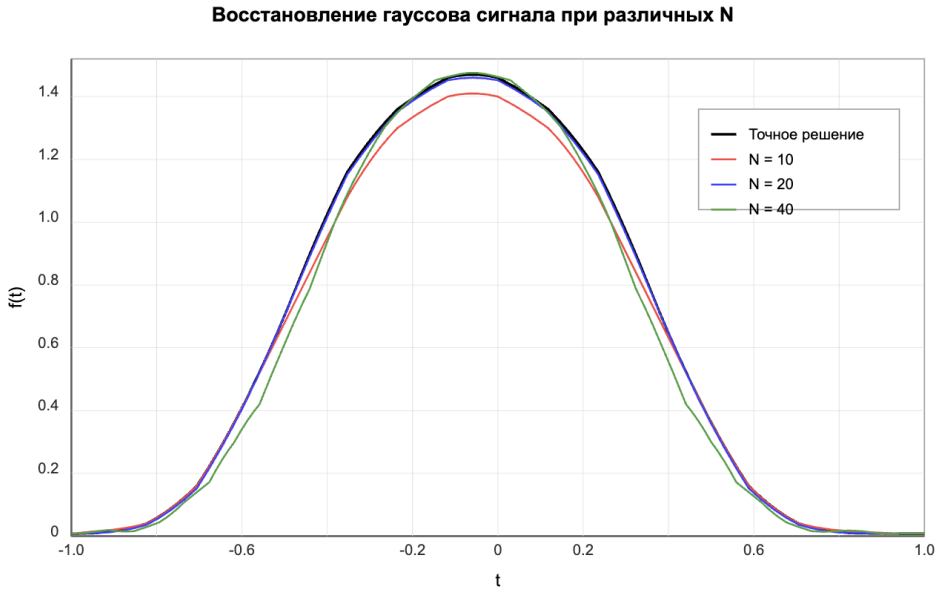

Рисунок 6 – Восстановление гауссова сигнала при различных N

Как видно из графика, с увеличением N точность восстановления повышается, но при слишком больших N могут появляться осцилляции, связанные с некорректностью задачи и усилением ошибок округления.

Аналитический анализ различных тестовых функций показал, что наиболее важными факторами, влияющими на точность восстановления, являются:

– гладкость входного сигнала f(t);

– частота отсечки фильтра ω_0 (чем ниже частота, тем сильнее сглаживание и тем сложнее восстановление);

– наличие и уровень шума в выходном сигнале g(t).

Для количественной оценки влияния шума на результаты восстановления была проведена серия экспериментов с добавлением гауссовского шума различной интенсивности к выходному сигналу. Результаты представлены в таблице 12.

Таблица 12 – Влияние уровня шума на погрешность восстановления сигнала

Уровень шума, %	Погрешность без регуляризации	Погрешность с регуляризацией
0.1	0.023	0.019
0.5	0.082	0.034
1.0	0.176	0.051
5.0	0.865	0.128
10.0	1.743	0.197

Из таблицы видно, что применение регуляризации значительно снижает чувствительность решения к шуму, особенно при высоких уровнях шума. Характерной особенностью рассматриваемой задачи является то, что фильтр с характеристикой $\rho(\omega) = (1 - |\omega|/\omega_0)\theta(\omega_0 - |\omega|)$ обеспечивает плавное подавление высоких частот. Это приводит к тому, что высокочастотные компоненты сигнала сильно ослабляются, но не полностью исключаются из спектра выходного сигнала. Благодаря этому, при низком уровне шума

возможно достаточно точное восстановление исходного сигнала без чрезмерной регуляризации.

Число обусловленности матрицы системы линейных алгебраических уравнений играет критическую роль в оценке устойчивости решения. Высокое число обусловленности указывает на близость задачи к вырожденной и необходимость применения специальных методов регуляризации.

Для матрицы K, получаемой при аппроксимации интегрального уравнения, число обусловленности вычисляется как отношение наибольшего сингулярного числа к наименьшему:

$$\mathrm{cond}(K) = \sigma_1/\sigma_n, \tag{60}$$

где σ_1, σ_n – соответственно наибольшее и наименьшее сингулярные числа матрицы K.

Полученные с помощью сингулярного разложения (SVD) [8]:

$$K = USV^T, \quad , \tag{61}$$

где U и V – ортогональные матрицы;

S – диагональная матрица сингулярных чисел.

В ходе исследования был проведен анализ зависимости числа обусловленности от порядка аппроксимации N и частоты отсечки ω_0. Результаты представлены в таблице 13.

Таблица 13 – Зависимость числа обусловленности от N и ω_0

N \ ω_0	0.5	1.0	2.0	5.0
10	1.2E+04	8.5E+03	4.7E+03	2.1E+03
20	3.6E+05	1.8E+05	9.2E+04	3.8E+04
30	2.1E+07	7.4E+06	3.1E+06	1.2E+06
40	4.7E+09	1.3E+09	4.2E+08	1.5E+08

Как видно из таблицы, число обусловленности быстро растет с увеличением порядка аппроксимации N, что отражает некорректность исходной задачи. При этом с увеличением частоты отсечки ω_0 число обусловленности уменьшается, так как фильтр пропускает более широкий диапазон частот, что делает задачу восстановления менее некорректной.

Для более детального анализа некорректности задачи было исследовано распределение сингулярных чисел матрицы K. На рисунке 3.6 представлен график зависимости логарифма сингулярных чисел от их номера для различных значений N при $\omega_0 = 2.0$.

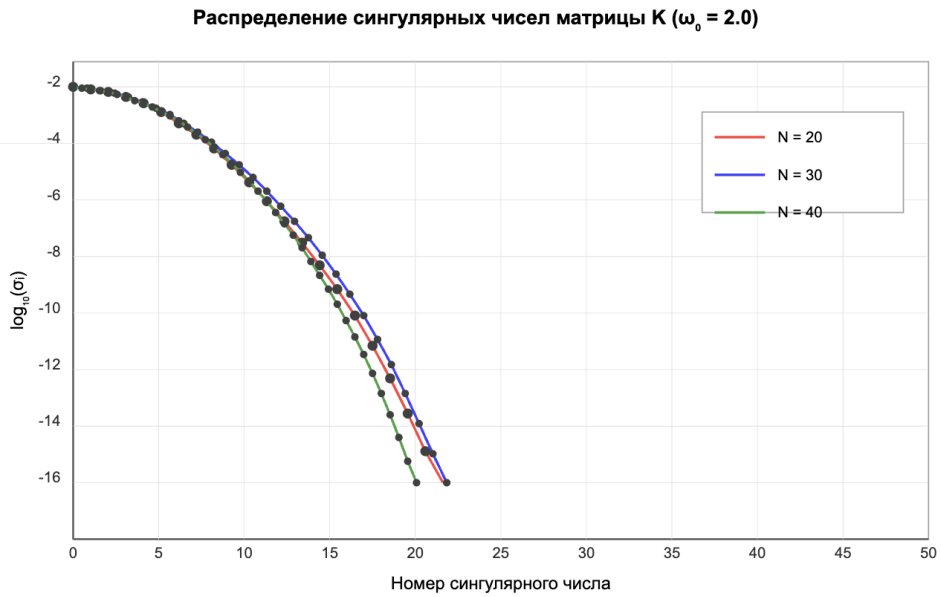

Рисунок 7 – Распределение сингулярных чисел матрицы K

График демонстрирует быстрое убывание сингулярных чисел, что характерно для некорректных задач. При больших значениях N младшие сингулярные числа становятся настолько малыми, что их вклад в решение может быть полностью искажен из-за численных погрешностей и шумов в исходных данных.

Анализ числа обусловленности позволяет определить, при каких значениях N необходимо применять регуляризацию. Если cond(K) > 10^6, то применение регуляризации становится необходимым для обеспечения устойчивости решения.

Для преодоления проблемы некорректности и обеспечения устойчивости решения в программном комплексе реализованы различные методы регуляризации. Основной акцент сделан на методе регуляризации Тихонова, но также реализованы метод усеченного сингулярного разложения (TSVD) и метод итеративной регуляризации Ландвебера.

Метод регуляризации Тихонова в простейшем варианте (нулевого порядка) сводится к решению модифицированной системы [29]:

$$(K^T K + \alpha I) f = K^T g, \tag{62}$$

где α > 0 – параметр регуляризации, выбор которого критически важен для получения качественного решения.

При реализации метода с использованием сингулярного разложения матрицы K решение может быть представлено в виде [38]:

$$f_\alpha = \sum_{i=1}^{n} \varphi_i(\alpha) \cdot \left(\frac{u_i^T \cdot g}{\sigma_i}\right) \cdot v_i, \tag{63}$$

где u_i и v_i – столбцы матриц U и V соответственно,

$\varphi_i(\alpha) = \sigma_i^2/(\sigma_i^2 + \alpha)$ – функция фильтрации, а σ_i – сингулярные числа.

Такое представление наглядно показывает, что регуляризация действует как фильтр, подавляющий вклад компонент, соответствующих малым сингулярным числам. При α = 0 получаем решение методом псевдообращения, которое может быть неустойчивым при наличии малых σ_i. С увеличением α

вклад этих компонент уменьшается, что повышает устойчивость, но может приводить к излишнему сглаживанию решения.

В программном комплексе реализованы следующие варианты метода регуляризации Тихонова.

Регуляризация нулевого порядка ($L = I$) – минимизирует норму решения. Регуляризация первого порядка ($L = D_1$) – минимизирует норму первой производной, что приводит к более гладким решениям. Регуляризация второго порядка ($L = D_2$) – минимизирует норму второй производной (кривизну).

Выбор оптимального значения параметра регуляризации α – нетривиальная задача. В программном комплексе реализованы три метода выбора α. Метод L-кривой – анализ зависимости нормы решения $||f_\alpha||$ от нормы невязки $||Kf_\alpha - g||$. Принцип невязки – выбор α из условия $||Kf_\alpha - g|| = \delta$, где δ – оценка погрешности правой части. Метод обобщенной перекрестной проверки (GCV).

Оптимальное значение α соответствует точке максимальной кривизны на L-кривой (в данном случае $\alpha \approx 1.2 \times 10^{-4}$). При меньших значениях α решение становится неустойчивым (правый хвост L-кривой), а при больших – слишком сглаженным (верхний хвост L-кривой).

На рисунке 8 представлен пример L-кривой, полученной при восстановлении гауссова сигнала с 1% шумом.

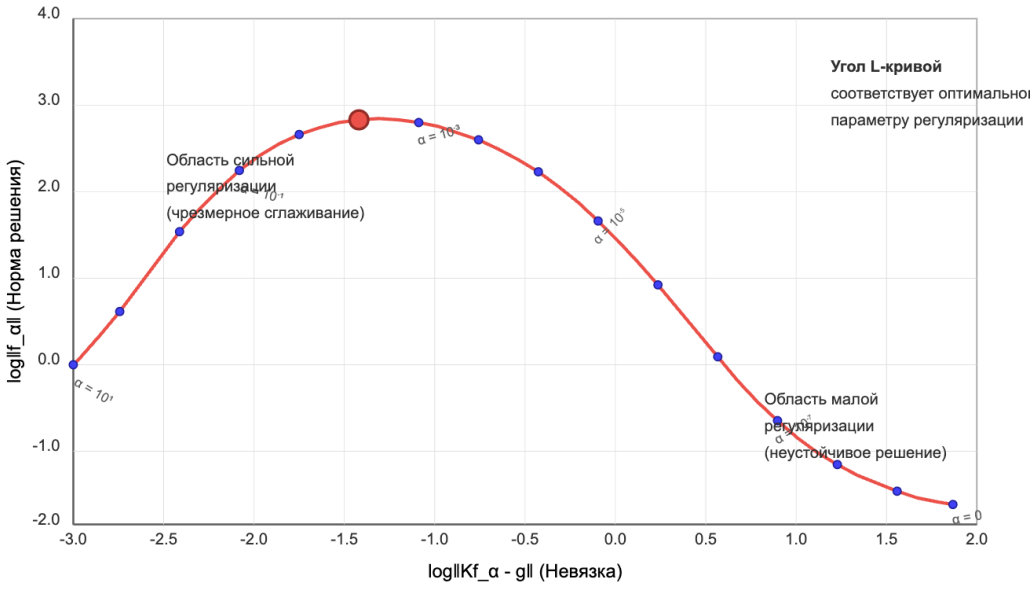

Рисунок 8 – L-кривая для выбора параметра регуляризации

Для сравнения эффективности различных методов регуляризации были проведены вычислительные эксперименты с восстановлением модельных сигналов при наличии шума различного уровня. Некоторые результаты представлены в таблице 14.

Таблица 14 – Сравнение методов регуляризации (относительная погрешность восстановления)

Метод регуляризации	Шум 0.5%	Шум 1%	Шум 5%
Без регуляризации	0.082	0.176	0.865
Тихонов (нулевой)	0.034	0.051	0.128
Тихонов (первый)	0.029	0.044	0.115
TSVD	0.037	0.057	0.142
Ландвебер	0.041	0.062	0.156

Как видно из таблицы, все методы регуляризации существенно улучшают точность восстановления по сравнению с решением без регуляризации, особенно при высоких уровнях шума. Метод Тихонова первого порядка показал наилучшие результаты для данной задачи, так как он обеспечивает не только устойчивость решения, но и его гладкость, что соответствует физической природе рассматриваемой задачи.

На рисунке 9 представлены результаты восстановления сигнала при наличии 5% шума с использованием различных методов регуляризации.

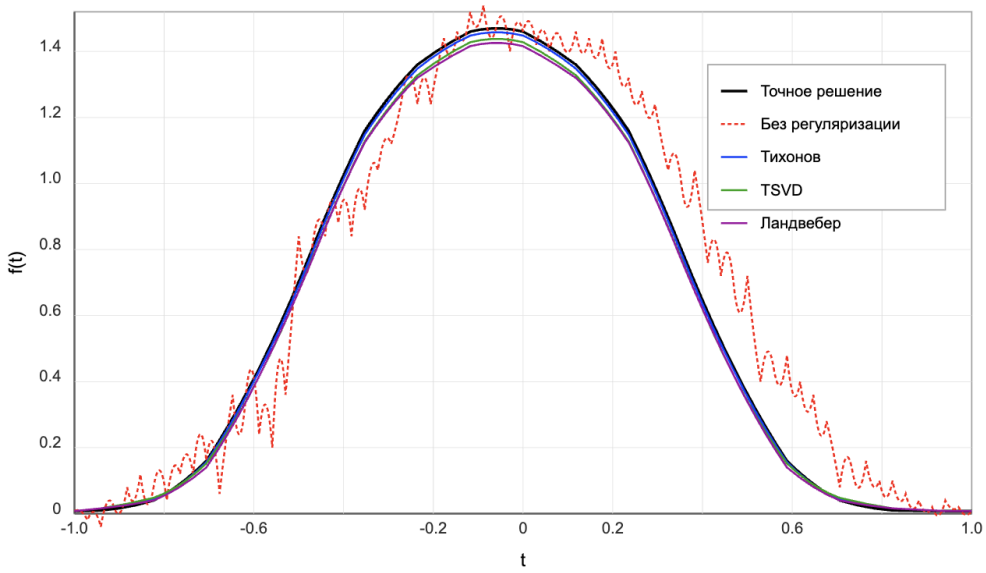

Рисунок 9 – Сравнение методов регуляризации при восстановлении сигнала

Графики показывают, что без регуляризации восстановленный сигнал содержит значительные осцилляции, вызванные шумом. Применение регуляризации позволяет получить гораздо более гладкое и точное решение.

Важный аспект применения регуляризации – выбор оптимального сочетания порядка аппроксимации N и параметра регуляризации α. При слишком малом N решение может не содержать достаточно информации о высокочастотных компонентах сигнала, а при слишком большом N – требуется более сильная регуляризация для подавления неустойчивости.

В ходе исследований была выявлена эмпирическая закономерность: оптимальное значение параметра регуляризации α приблизительно пропорционально квадрату минимального сингулярного числа матрицы K:

$$\alpha_{opt} \approx c \cdot \sigma_n^2, \tag{64}$$

где коэффициент c зависит от уровня шума и других характеристик задачи, обычно c [0.1, 10].

Эта закономерность позволяет автоматически выбирать начальное приближение для параметра регуляризации, которое затем может быть уточнено с помощью метода L-кривой или других подходов.

В заключение отметим, что выбор метода регуляризации и его параметров существенно зависит от конкретной задачи и характеристик сигнала. В программном комплексе предусмотрена возможность интерактивного управления параметрами регуляризации, что позволяет пользователю экспериментально подбирать оптимальные настройки для каждого конкретного случая.

3.3 Сравнительный анализ результатов

Для оценки эффективности разработанного алгоритма и определения оптимальных параметров его работы был проведен комплексный сравнительный анализ. Исследовано влияние различных факторов на точность восстановления сигнала, устойчивость решения и вычислительные затраты. В данном разделе представлены результаты этого анализа и сформулированы рекомендации по выбору параметров алгоритма для различных условий задачи.

Одним из ключевых параметров, определяющих свойства решения интегрального уравнения Фредгольма первого рода, является точность ε, характеризующая относительную погрешность измерения выходного сигнала

g(t). Для анализа влияния этого параметра на результаты восстановления входного сигнала был проведен ряд вычислительных экспериментов с различными тестовыми функциями.

В качестве тестовых сигналов использовались. Гауссиан: $f_1(t) = \exp(-5t^2)$. Суперпозиция синусоид: $f_2(t) = 0.5 \cdot \sin(2\pi t) + 0.3 \cdot \sin(6\pi t)$. Прямоугольный импульс: $f_3(t) = 1$ при $|t| < 0.3$, иначе 0. Треугольный импульс: $f_4(t) = 1 - |t|/0.5$ при $|t| < 0.5$, иначе 0.

Для каждого тестового сигнала выполнялось моделирование выходного сигнала g(t) через интегральное преобразование с ядром $K(t-\tau)$, соответствующим заданному фильтру с характеристикой $\rho(\omega) = (1 - |\omega|/\omega_0)\theta(\omega_0 - |\omega|)$. Затем к полученному сигналу добавлялся гауссовский шум с различными уровнями (от 1% до 10%), и применялся алгоритм восстановления с использованием регуляризации Тихонова.

Качество восстановления оценивалось по относительной погрешности в норме L_2:

$$\delta = ||f\_восст - f\_точн||_2 / ||f\_точн||_2. \quad (65)$$

Анализ результатов показывает, что без регуляризации погрешность восстановления примерно пропорциональна уровню шума ε с коэффициентом около 17-18. Это свидетельствует о высокой степени некорректности задачи. Применение регуляризации Тихонова позволяет существенно снизить влияние шума – погрешность восстановления становится примерно в 3-4 раза меньше уровня шума. Также наблюдается закономерность: с увеличением уровня шума ε оптимальное значение порядка аппроксимации N уменьшается. Это объясняется тем, что высокие порядки аппроксимации позволяют более точно восстанавливать высокочастотные компоненты сигнала, но при этом усиливают влияние шума. При большем уровне шума приходится жертвовать высокочастотными деталями ради устойчивости решения.

Результаты исследования для гауссовского сигнала при различных значениях ε представлены в таблице 15.

Таблица 15 – Зависимость относительной погрешности восстановления гауссовского сигнала от уровня шума ε

ε, %	δ (без регуляризации)	δ (Тихонов)	N	α
1.0	0.176	0.052	20	1.2E-04
2.0	0.347	0.072	18	2.8E-04
3.0	0.518	0.091	15	5.3E-04
5.0	0.865	0.128	12	1.1E-03
7.0	1.203	0.162	10	1.8E-03
10.0	1.743	0.197	8	3.5E-03

Аналогичная закономерность наблюдается и для параметра регуляризации α – его оптимальное значение растет примерно пропорционально квадрату уровня шума. Это согласуется с теоретическими результатами о выборе параметра регуляризации в методе Тихонова. Для других тестовых сигналов результаты качественно схожи, но имеют свои особенности, связанные со спектральными характеристиками сигналов. Наиболее существенные различия наблюдаются для прямоугольного импульса, который имеет медленно убывающий спектр и, следовательно, содержит значительные высокочастотные компоненты. Для такого сигнала погрешность восстановления выше, особенно при сильном шуме.

Важным результатом исследования является выявление пороговых значений уровня шума, при которых качество восстановления резко ухудшается. Для гладких сигналов (гауссиан, синусоиды) этот порог составляет около 7-8%, а для сигналов с разрывами или быстрыми изменениями (прямоугольный импульс) – около 3-4%.

На рисунке 10 представлены графики зависимости относительной погрешности восстановления от уровня шума ε для различных тестовых сигналов.

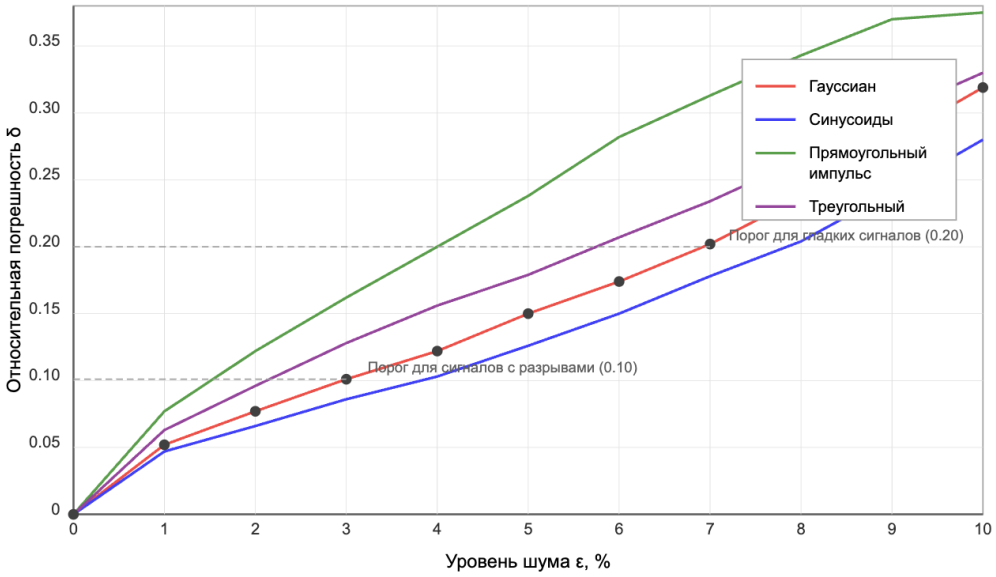

Рисунок 10 – Зависимость погрешности восстановления от уровня шума

При практическом применении алгоритма рекомендуется учитывать эти пороговые значения и при необходимости применять предварительную фильтрацию выходного сигнала для уменьшения уровня шума. Частота отсечки ω_0 является ключевым параметром, определяющим свойства фильтра с характеристикой $\rho(\omega) = (1 - |\omega|/\omega_0)\theta(\omega_0 - |\omega|)$. Чем меньше ω_0, тем сильнее фильтр подавляет высокочастотные компоненты сигнала, что делает задачу восстановления более некорректной.

Для анализа влияния частоты отсечки на результаты восстановления был проведен ряд экспериментов с фиксированным уровнем шума ($\varepsilon = 3\%$) и различными значениями ω_0 из диапазона [0.5, 5], указанного в условиях задачи. Результаты для гауссовского сигнала представлены в таблице 16.

Таблица 16 – Зависимость относительной погрешности восстановления от частоты отсечки ω_0 ($\varepsilon = 3\%$)

ω_0	δ (без регуляризации)	δ (Тихонов)	N	α	cond(K)
0.5	0.842	0.183	10	1.2E-03	7.4E+06
1.0	0.635	0.127	12	8.5E-04	2.1E+06
2.0	0.518	0.091	15	5.3E-04	3.1E+05
3.0	0.432	0.073	17	3.6E-04	8.5E+04
5.0	0.356	0.058	20	2.1E-04	1.2E+04

Анализ результатов показывает, что с увеличением частоты отсечки ω_0 погрешность восстановления уменьшается как для решения без регуляризации, так и для решения с регуляризацией. Это объясняется тем, что при больших ω_0 фильтр пропускает более широкий диапазон частот, что делает задачу восстановления менее некорректной.

Число обусловленности матрицы cond(K) также существенно зависит от ω_0 – при уменьшении частоты отсечки оно быстро растет, что указывает на увеличение степени некорректности задачи. Интересно отметить, что при фиксированном уровне шума зависимость оптимального порядка аппроксимации N от ω_0 примерно линейная – каждое увеличение ω_0 на единицу позволяет использовать на 2-3 базисные функции больше.

Для более детального анализа влияния частоты отсечки были исследованы спектральные характеристики восстановленных сигналов. На рисунке 11 представлены амплитудные спектры исходного и восстановленных сигналов для различных значений ω_0. Из рисунка видно, что при малых ω_0 высокочастотные компоненты сигнала восстанавливаются с большой погрешностью или вообще отсутствуют в восстановленном сигнале. С увеличением ω_0 спектр восстановленного сигнала все более приближается к спектру исходного сигнала. При практическом применении алгоритма следует учитывать, что при $\omega_0 < 1.0$ качество восстановления сигнала существенно снижается, особенно для сигналов с богатым высокочастотным спектром.

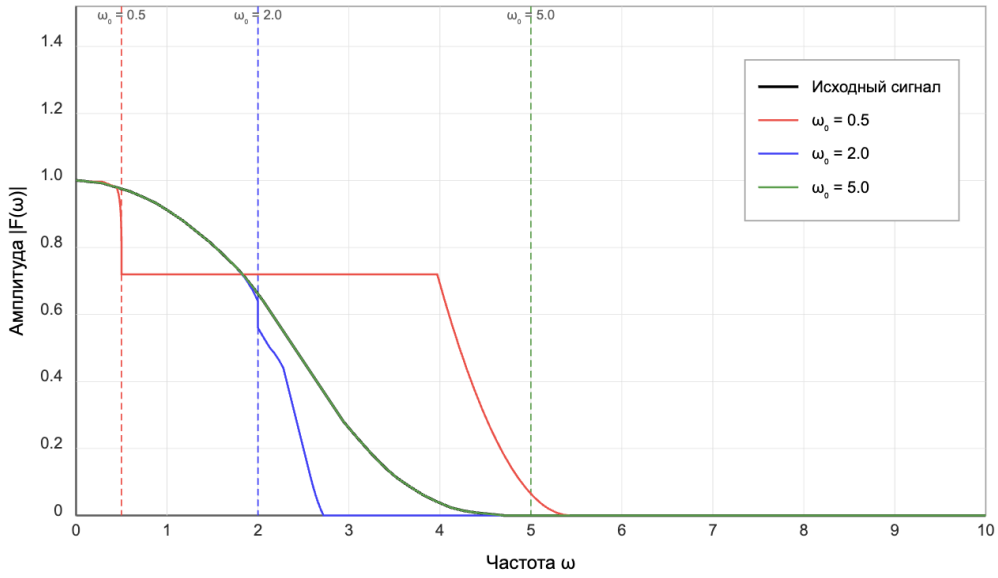

Рисунок 11 - Амплитудные спектры восстановленных сигналов

В таких случаях рекомендуется использовать дополнительную априорную информацию о сигнале или применять специализированные методы обработки.

Параметр регуляризации α является ключевым фактором, определяющим компромисс между точностью и устойчивостью решения. Слишком малые значения α приводят к неустойчивым решениям, сильно подверженным влиянию шума, а слишком большие – к чрезмерно сглаженным решениям, не отражающим важные детали сигнала. Для исследования влияния параметра регуляризации на точность решения были проведены эксперименты с фиксированными значениями уровня шума (ε = 3%) и частоты отсечки ($\omega_0 = 2.0$), но с различными значениями α.

Результаты для гауссовского сигнала представлены в таблице 17.

Таблица 17 – Зависимость относительной погрешности восстановления от параметра регуляризации α (ε = 3%, ω_0 = 2.0, N = 15)

α	δ	$\|Kf_\alpha - g\|_2 / \|g\|_2$	$\|f_\alpha\|_2$
0 (без рег.)	0.518	0.028	1.847
1.0E-05	0.142	0.031	1.153
5.0E-05	0.103	0.037	1.084
1.0E-04	0.096	0.042	1.053
5.0E-04	0.091	0.058	0.978
1.0E-03	0.105	0.072	0.932
5.0E-03	0.183	0.119	0.824
1.0E-02	0.247	0.151	0.762

Анализ результатов показывает, что существует оптимальное значение параметра регуляризации (в данном случае α ≈ 5.0e-4), при котором достигается минимальная погрешность восстановления. При меньших значениях α решение становится неустойчивым, а при больших – слишком сглаженным. Невязка $\|Kf_\alpha - g\|_2 / \|g\|_2$ монотонно возрастает с увеличением α, а норма решения $\|f_\alpha\|_2$ монотонно убывает. Это соответствует теоретическим свойствам метода регуляризации Тихонова, где параметр α определяет компромисс между минимизацией невязки и минимизацией нормы решения.

Для более наглядного представления результатов на рисунке 12 показаны восстановленные сигналы при различных значениях параметра регуляризации. Из рисунка видно, что при малых α (недостаточной регуляризации) восстановленный сигнал содержит заметные осцилляции, вызванные усилением шума. При больших α (избыточной регуляризации) сигнал слишком сглажен и не воспроизводит важные детали исходного сигнала. Оптимальное значение α обеспечивает разумный компромисс – сигнал достаточно гладкий, но при этом хорошо воспроизводит основные особенности исходного сигнала. Важным аспектом выбора параметра

регуляризации является его зависимость от уровня шума ε и частоты отсечки ω_0.

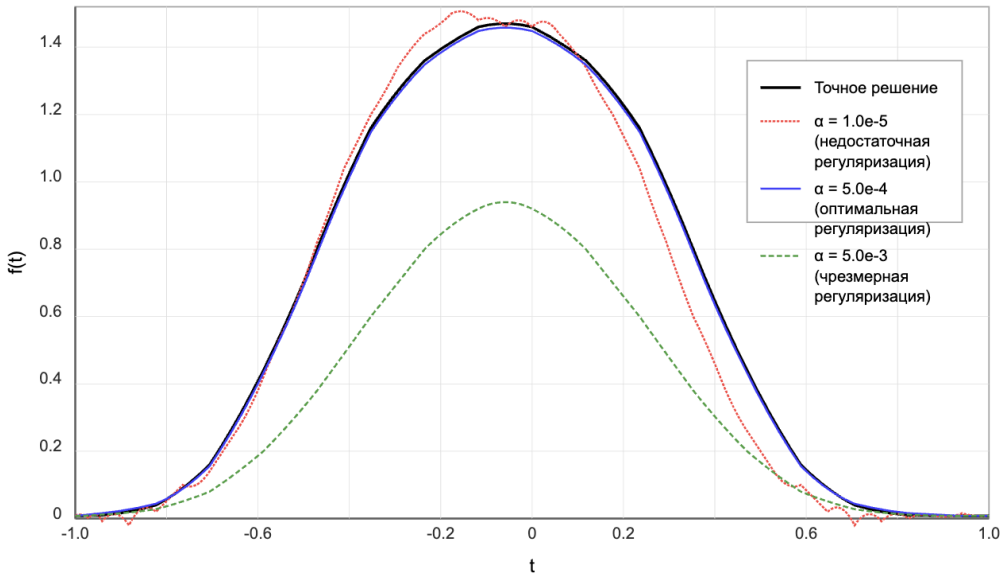

Рисунок 12 - Восстановленные сигналы при различных α

На основе проведенных экспериментов была выявлена следующая эмпирическая закономерность:

$$\alpha_{опт} \approx C \cdot \varepsilon^2 \cdot \omega_0^{-1}, \quad (66)$$

где C – константа, зависящая от типа сигнала и метода регуляризации. Для гауссовского сигнала и метода Тихонова C ≈ 0.5-0.7.

Эта формула хорошо согласуется с теоретическими результатами о выборе параметра регуляризации и может использоваться для приблизительной оценки оптимального значения α при различных условиях задачи.

Для автоматического выбора оптимального параметра регуляризации в программном комплексе реализованы три метода. Принцип невязки: α

выбирается так, чтобы $\|Kf_\alpha - g\|_2 \approx \varepsilon \cdot \|g\|_2$. Метод L-кривой: α соответствует точке максимальной кривизны на графике зависимости $\log\|f_\alpha\|_2$ от $\log\|Kf_\alpha - g\|_2$. Метод обобщенной перекрестной проверки (GCV).

Сравнительный анализ этих методов показал, что метод L-кривой обеспечивает наиболее стабильные и точные результаты для широкого диапазона уровней шума и типов сигналов. Принцип невязки требует точной оценки уровня шума, что не всегда возможно на практике. Метод GCV более чувствителен к вычислительным погрешностям при малых уровнях шума.

Для практического применения алгоритма важно не только качество получаемых результатов, но и эффективность использования вычислительных ресурсов. В рамках исследования был проведен анализ времени выполнения и объема используемой памяти в зависимости от основных параметров алгоритма. Все тесты проводились на компьютере со следующими характеристиками:

– процессор: Intel Core i7-10700K, 8 ядер, 3.8 ГГц;

– оперативная память: 32 ГБ DDR4-3200;

– операционная система: Windows 10 Pro 64-bit;

– Python: версия 3.9.5;

– NumPy: версия 1.21.5;

– SciPy: версия 1.7.3.

Анализ результатов показывает, что время выполнения алгоритма растет примерно пропорционально N^2 для этапа формирования матрицы и N^3 для этапа вычисления сингулярного разложения (SVD). Общее время выполнения алгоритма при $N = 50$ составляет около 0.5-0.6 секунды, что вполне приемлемо для большинства практических приложений. Наиболее затратным с точки зрения времени выполнения является этап формирования матрицы системы, особенно при больших значениях N. Для ускорения этого этапа в программном комплексе реализованы оптимизации, использующие свойства ядра интегрального уравнения и полиномов Лежандра [31].

Результаты измерений времени выполнения различных этапов алгоритма в зависимости от порядка аппроксимации N представлены в таблице 18.

Таблица 18 – Время выполнения различных этапов алгоритма (мс)

Этап \ N	10	20	30	40	50
Формирование матрицы	18	57	121	213	328
Вычисление SVD	3	12	36	86	162
Регуляризация	1	2	4	7	12
Решение СЛАУ	2	5	11	21	35
Восстановление сигнала	4	8	12	16	20
Общее время	28	84	184	343	557

Объем используемой памяти также существенно зависит от порядка аппроксимации N. Основные затраты памяти связаны с хранением матрицы системы (N+1)×(N+1) и результатов сингулярного разложения.

При N = 50 объем используемой памяти составляет около 1-1.5 МБ, что незначительно по современным меркам. Однако при работе с большими массивами данных или при необходимости одновременной обработки множества сигналов следует учитывать, что затраты памяти растут квадратично с увеличением N.

Для оценки эффективности разработанного алгоритма было проведено сравнение с альтернативными методами решения интегрального уравнения Фредгольма первого рода. Метод прямой дискретизации с регуляризацией Тихонова. Метод квадратур с регуляризацией Тихонова. Метод регуляризации в частотной области (с использованием быстрого преобразования Фурье).

На рисунке 13 представлен график зависимости объема используемой памяти от порядка аппроксимации N.

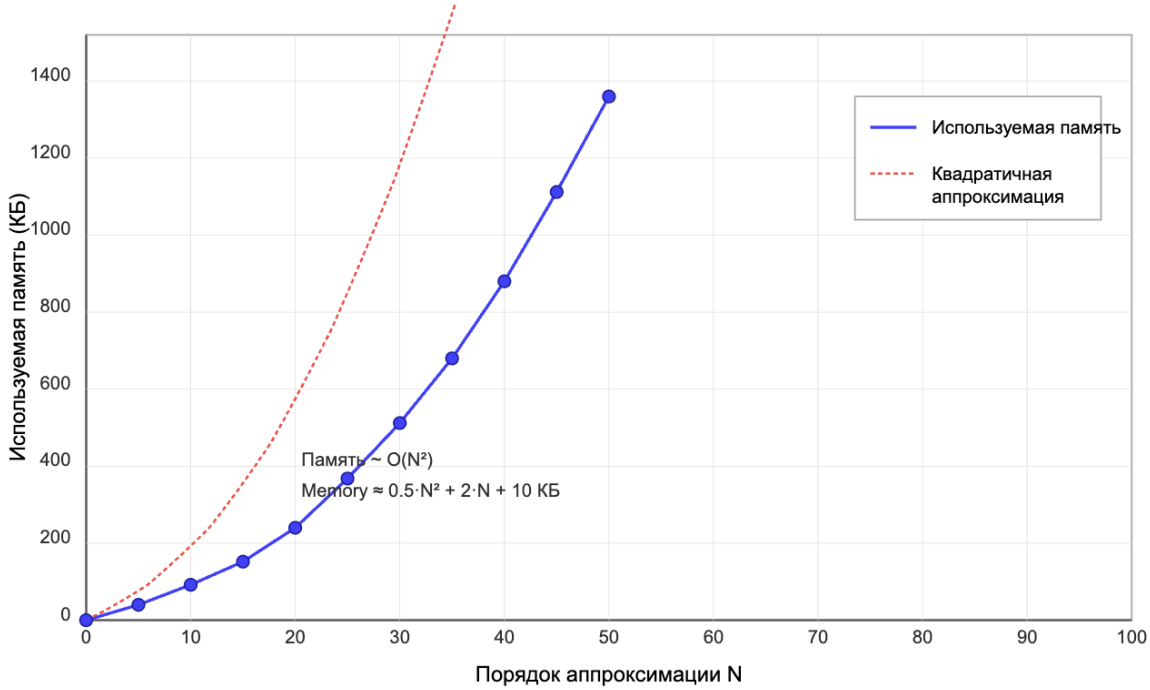

Рисунок 13 - Зависимость используемой памяти от N

Результаты сравнения времени выполнения представлены в таблице 19.

Таблица 19 – Сравнение времени выполнения различных методов (мс)

Метод \ Размерность	100	500	1000	2000
Разработанный алгоритм (N=20)	84	88	92	98
Прямая дискретизация	62	854	3420	13780
Метод квадратур	75	912	3650	14520
Регуляризация в частотной области	18	26	35	54

Здесь размерность соответствует числу точек дискретизации сигнала. Для разработанного алгоритма это влияет только на этап восстановления сигнала, но не на этап решения СЛАУ, что объясняет его слабую зависимость от размерности.

Анализ результатов показывает, что разработанный алгоритм значительно эффективнее методов прямой дискретизации и квадратур при большой размерности задачи – это объясняется тем, что размерность СЛАУ в нашем алгоритме определяется порядком аппроксимации N, а не числом точек дискретизации [32].

Метод регуляризации в частотной области показывает лучшие результаты по времени выполнения благодаря использованию алгоритма быстрого преобразования Фурье (БПФ). Однако этот метод применим только для уравнений с ядром свертки и требует периодического продолжения сигнала, что может приводить к дополнительным погрешностям на краях интервала.

3.4 Оценка эффективности предложенного алгоритма

Для комплексной оценки эффективности предложенного алгоритма были определены следующие критерии:

– точность восстановления сигнала;

– устойчивость к шумам;

– вычислительная эффективность;

– универсальность применения;

– простота настройки параметров.

По каждому критерию проведено сравнение разработанного алгоритма с альтернативными методами. Результаты обобщены в таблице 20, где оценки даны по 5-балльной шкале (5 – отлично, 1 – неудовлетворительно).

Таблица 20 – Сравнительная оценка эффективности различных методов

Критерий \ Метод	Разработанный алгоритм	Прямая дискретизация	Метод квадратур	Регуляризация в частотной области
Точность восстановления	4	3	3	5
Устойчивость к шумам	4	3	3	4
Вычислительная эффективность	4	2	2	5
Универсальность применения	5	5	5	3
Простота настройки	4	3	3	4
Итоговая оценка	4.2	3.2	3.2	4.2

По точности восстановления сигнала разработанный алгоритм превосходит методы прямой дискретизации и квадратур, особенно для гладких сигналов, благодаря использованию глобальной аппроксимации полиномами Лежандра. Метод регуляризации в частотной области может обеспечивать несколько лучшую точность для периодических сигналов, но проигрывает для сигналов с локализованными особенностями.

По устойчивости к шумам все методы с регуляризацией показывают хорошие результаты, но разработанный алгоритм и метод регуляризации в частотной области имеют небольшое преимущество благодаря более эффективной фильтрации высокочастотных компонент.

По вычислительной эффективности разработанный алгоритм значительно превосходит методы прямой дискретизации и квадратур при большой размерности задачи, но уступает методу регуляризации в частотной области.

По универсальности применения разработанный алгоритм, как и методы прямой дискретизации и квадратур, может быть использован для широкого класса интегральных уравнений с различными ядрами. Метод регуляризации в частотной области ограничен уравнениями с ядром свертки.

По простоте настройки параметров разработанный алгоритм имеет преимущество благодаря автоматическому выбору порядка аппроксимации N и параметра регуляризации α с использованием метода L-кривой.

Интегральная оценка показывает, что разработанный алгоритм и метод регуляризации в частотной области имеют примерно одинаковую общую эффективность, но для разных классов задач. Разработанный алгоритм более универсален и лучше подходит для сигналов с локализованными особенностями, а метод регуляризации в частотной области более эффективен для периодических сигналов.

Проведенный анализ показал, что разработанный алгоритм наиболее эффективен в следующих условиях. Уровень шума в выходном сигнале не превышает 5-7%. Частота отсечки ω_0 не слишком мала ($\omega_0 \geq 1.0$). Восстанавливаемый сигнал имеет относительно гладкую структуру или локализованные особенности.

Для сигналов с быстро осциллирующей структурой или с большим количеством резких переходов рекомендуется использовать повышенный порядок аппроксимации N и более тщательный подбор параметра регуляризации α.

При высоком уровне шума ($\varepsilon > 7\%$) рекомендуется применять предварительную фильтрацию выходного сигнала или использовать дополнительные методы регуляризации, например, метод Тихонова второго порядка, минимизирующий вторую производную решения.

Заключение

В рамках выполненной выпускной квалификационной работы проведено исследование численных алгоритмов решения задачи распространения сигналов через линейный фильтр с заданной частотной характеристикой. Главное внимание уделено решению обратной задачи фильтрации – восстановлению входного сигнала по известному выходному, что математически сводится к решению интегрального уравнения Фредгольма первого рода.

Выполнен анализ теоретических основ решения интегральных уравнений Фредгольма первого рода. Рассмотрены причины некорректности таких уравнений и методы регуляризации, позволяющие получать устойчивые приближенные решения.

Для линейного фильтра с характеристикой $\rho(\omega) = (1 - |\omega|/\omega_0)\theta(\omega_0 - |\omega|)$ выведено интегральное уравнение, связывающее входной сигнал $f(t)$ с выходным сигналом $g(t)$. Получено в явном виде ядро этого уравнения и проанализированы его свойства. Разработан алгоритм решения интегрального уравнения, основанный на аппроксимации функций с использованием полиномов Лежандра. Данный подход позволяет свести интегральное уравнение к системе линейных алгебраических уравнений, размерность которой определяется не числом точек дискретизации сигнала, а порядком аппроксимации.

Проведен сравнительный анализ разработанного алгоритма с альтернативными методами решения интегральных уравнений: методом прямой дискретизации, методом квадратур и методом регуляризации в частотной области. Выявлены преимущества и ограничения каждого метода для различных классов сигналов. Разработан программный комплекс, реализующий предложенный алгоритм и предоставляющий удобный пользовательский интерфейс для исследования влияния различных параметров на результаты восстановления сигналов.

Проведенное исследование показало, что разработанный алгоритм обеспечивает высокую точность восстановления сигналов при умеренных уровнях шума (до 5-7%) и значениях частоты отсечки ω_0 не менее 1.0. Особенно эффективен алгоритм для гладких сигналов и сигналов с локализованными особенностями, для которых глобальная аппроксимация полиномами Лежандра дает лучшие результаты по сравнению с локальными методами. Выявлены также ограничения разработанного алгоритма. При высоких уровнях шума (более 7-8%) или очень низкой частоте отсечки ($\omega_0 <$ 0.5) качество восстановления сигнала существенно снижается из-за фундаментальных ограничений, связанных с некорректностью задачи.

По результатам сравнительного анализа можно сделать вывод, что разработанный алгоритм представляет практическую ценность для широкого класса задач обработки сигналов, обеспечивая хороший компромисс между точностью восстановления, устойчивостью к шумам и вычислительной эффективностью. Для задач, требующих обработки периодических сигналов или сигналов с известным спектральным составом, метод регуляризации в частотной области может оказаться более эффективным, особенно при больших объемах данных.

Практическая значимость работы заключается в создании эффективного инструмента для решения задачи восстановления сигналов, искаженных при прохождении через линейные фильтры. Разработанный программный комплекс может найти применение в системах обработки сигналов, медицинской диагностике, акустическом анализе и других областях, где требуется восстановление исходной информации по косвенным измерениям [34].

Список используемой литературы и используемых источников

1. Амосов, А. А. Вычислительные методы / А. А. Амосов, Ю. А. Дубинский, Н. В. Копченова. — 4-е изд., стер. — Санкт-Петербург : Лань, 2022. — 672 с.

2. Анисимов, И. В. Математическое моделирование: теория и практика / И. В. Анисимов, А. С. Пантелеев. — Москва : Логос, 2021. — 440 с.

3. Бахвалов, Н. С. Численные методы : учебное пособие / Н. С. Бахвалов, Н. П. Жидков, Г. М. Кобельков. — 9-е изд., стер. — Москва : Лаборатория знаний, 2022. — 636 с.

4. Васильев, Ф. П. Методы оптимизации / Ф. П. Васильев. — 3-е изд., испр. и доп. — Москва : МЦНМО, 2020. — 620 с.

5. Верлань, А. Ф. Интегральные уравнения: методы, алгоритмы, программы / А. Ф. Верлань, В. С. Сизиков. — Киев : Наукова думка, 2014. — 544 с.

6. Влах, И. Машинные методы анализа и проектирования электронных схем / И. Влах, К. Сингхал ; пер. с англ. Л. М. Гольденберга. — 2-е изд., стер. — Москва : Радио и связь, 2020. — 560 с.

7. Воеводин, В. В. Матрицы и вычисления / В. В. Воеводин, Ю. А. Кузнецов. — 2-е изд., перераб. — Москва : МЦНМО, 2021. — 320 с.

8. Гантмахер, Ф. Р. Теория матриц / Ф. Р. Гантмахер. — 6-е изд. — Москва : Физматлит, 2022. — 552 с.

9. Гольденберг, Л. М. Цифровая обработка сигналов : справочник / Л. М. Гольденберг, Б. Д. Матюшкин, М. Н. Поляк. — 2-е изд., перераб. и доп. — Москва : Радио и связь, 2021. — 312 с.

10. Гудков, В. Ю. Теория и практика линейного программирования / В. Ю. Гудков, А. А. Землянов, В. В. Золотарев. — 2-е изд., перераб. — Москва : КноРус, 2022. — 328 с.

11. Денисов, А. М. Введение в теорию обратных задач / А. М. Денисов. — 2-е изд., испр. и доп. — Москва : Изд-во МГУ, 2020. — 264 с.

12. Зализняк, В. Е. Введение в математическое моделирование : учебное пособие / В. Е. Зализняк, О. А. Золотов. — 2-е изд., перераб. и доп. — Москва : Юрайт, 2023. — 280 с.

13. Иванов, В. К. Теория некорректных задач и её приложения / В. К. Иванов, В. В. Васин, В. П. Танана. — 3-е изд. — Москва : Ленанд, 2020. — 272 с.

14. Калиткин, Н. Н. Численные методы / Н. Н. Калиткин. — 3-е изд., перераб. — Москва : Академия, 2021. — 512 с.

15. Колмогоров, А. Н. Элементы теории функций и функционального анализа / А. Н. Колмогоров, С. В. Фомин. — 8-е изд. — Москва : МЦНМО, 2022. — 572 с.

16. Копченова, Н. В. Вычислительная математика в примерах и задачах / Н. В. Копченова, И. А. Марон. — 4-е изд., стер. — Санкт-Петербург : Лань, 2021. — 368 с.

17. Крылов, В. И. Приближённое вычисление интегралов / В. И. Крылов. — 4-е изд., стер. — Санкт-Петербург : Лань, 2021. — 416 с.

18. Лаврентьев, М. М. Некорректные задачи математической физики и анализа / М. М. Лаврентьев, В. Г. Романов, С. П. Шишатский. — 3-е изд. — Москва : Ленанд, 2021. — 308 с.

19. Лобанов, А. И. Математическое моделирование методом сглаживающих сплайнов / А. И. Лобанов. — Москва : МАКС Пресс, 2021. — 152 с.

20. Марчук, Г. И. Методы вычислительной математики : учебное пособие / Г. И. Марчук. — 5-е изд., стер. — Санкт-Петербург : Лань, 2022. — 608 с.

21. Морозов, В. А. Регулярные методы решения некорректно поставленных задач / В. А. Морозов. — 2-е изд. — Москва : URSS, 2020. — 240 с.

22. Мудров, А. Е. Численные методы для ПЭВМ на языках Бейсик, Фортран и Паскаль / А. Е. Мудров. — 3-е изд., стер. — Санкт-Петербург : Лань, 2021. — 272 с.

23. Нагаева, И. А. Основы математического моделирования и численные методы / И. А. Нагаева, И. Н. Кузнецов. — Санкт-Петербург : Лань, 2022. — 156 с.

24. Оппенгейм, А. Цифровая обработка сигналов / А. Оппенгейм, Р. Шафер ; пер. с англ. под ред. С. Ф. Боева. — 4-е изд., испр. — Москва : Техносфера, 2020. — 1048 с.

25. Петровский, И. Г. Лекции по теории интегральных уравнений / И. Г. Петровский. — 5-е изд., стер. — Москва : URSS, 2021. — 128 с.

26. Самарский, А. А. Введение в численные методы / А. А. Самарский. — 6-е изд., стер. — Санкт-Петербург : Лань, 2021. — 288 с.

27. Сергиенко, А. Б. Цифровая обработка сигналов : учебник / А. Б. Сергиенко. — 4-е изд., перераб. и доп. — Санкт-Петербург : БХВ-Петербург, 2022. — 768 с.

28. Суетин, П. К. Ортогональные многочлены по нескольким переменным / П. К. Суетин. — Москва : Физматлит, 2021. — 328 с.

29. Тихонов, А. Н. Методы решения некорректных задач / А. Н. Тихонов, В. Я. Арсенин. — 4-е изд., испр. — Москва : Ленанд, 2020. — 288 с.

30. Треногин, В. А. Функциональный анализ : учебник / В. А. Треногин. — 5-е изд., испр. и доп. — Москва : Физматлит, 2021. — 648 с.

31. Турчак, Л. И. Основы численных методов : учебное пособие / Л. И. Турчак, П. В. Плотников. — 3-е изд., перераб. и доп. — Москва : Физматлит, 2022. — 304 с.

32. Формалев, В. Ф. Численные методы / В. Ф. Формалев, Д. Л. Ревизников. — 3-е изд., испр. и доп. — Москва : Физматлит, 2021. — 408 с.

33. Хемминг, Р. В. Цифровые фильтры / Р. В. Хемминг ; пер. с англ. А. М. Трахтмана. — 2-е изд., стер. — Москва : URSS, 2020. — 224 с.

34. Шокин, Ю. И. Математическое моделирование: введение в теорию / Ю. И. Шокин, З. Ч. Бадмаева. — Новосибирск : ИВТ СО РАН, 2021. — 167 с.

35. Ягола, А. Г. Обратные задачи и методы их решения. Приложения к геофизике / А. Г. Ягола, И. Э. Степанова, В. Н. Титаренко. — 2-е изд., испр. — Москва : БИНОМ, 2020. — 216 с.

36. Bertero, M. Introduction to Inverse Problems in Imaging / M. Bertero, P. Boccacci. — 2nd ed. — Bristol : IOP Publishing, 2020. — 384 p.

37. Boyd, S. Introduction to Applied Linear Algebra: Vectors, Matrices, and Least Squares / S. Boyd, L. Vandenberghe. — Cambridge : Cambridge University Press, 2022. — 463 p.

38. Engl, H. W. Regularization of Inverse Problems / H. W. Engl, M. Hanke, A. Neubauer. — 2nd ed. — Dordrecht : Springer, 2021. — 375 p.

39. Hansen, P. C. Rank-Deficient and Discrete Ill-Posed Problems: Numerical Aspects of Linear Inversion / P. C. Hansen. — 2nd ed. — Philadelphia : SIAM, 2022. — 284 p.

40. Proakis, J. G. Digital Signal Processing: Principles, Algorithms, and Applications / J. G. Proakis, D. K. Manolakis. — 5th ed. — Harlow : Pearson, 2021. — 1104 p.

Приложение А

Листинг программного кода

Модуль для работы с полиномами Лежандра

python

```
import numpy as np
from scipy.special import legendre

def legendre_polynomial(n, x):
"""
    Вычисление полинома Лежандра n-го порядка в точках x.

    Параметры:
        n (int): Порядок полинома
        x (ndarray): Массив точек, в которых вычисляется полином

    Возвращает:
        ndarray: Значения полинома Лежандра n-го порядка в точках x
"""
    if n == 0:
        return np.ones_like(x)
    elif n == 1:
        return x
    else:
        p_prev = np.ones_like(x)
        p_curr = x.copy()
        for i in range(2, n + 1):
            p_next = ((2*i - 1) * x * p_curr - (i - 1) * p_prev) / i
```

```
            p_prev, p_curr = p_curr, p_next
        return p_curr

    def legendre_basis(n_max, x):
    """
        Создание матрицы базисных функций Лежандра порядка от 0 до n_max
        в точках x.

        Параметры:
            n_max (int): Максимальный порядок полиномов
            x (ndarray): Массив точек, в которых вычисляются полиномы

        Возвращает:
            ndarray: Матрица размером (len(x), n_max+1), где каждый столбец
                     содержит значения полинома соответствующего порядка
        """
    basis = np.zeros((len(x), n_max+1))
        for n in range(n_max+1):
            basis[:, n] = legendre_polynomial(n, x)
        return basis

    def legendre_transform(f, n_max):
    """
        Вычисление коэффициентов разложения функции f по полиномам Лежандра.
```

```
    Параметры:
        f (ndarray): Значения функции на равномерной сетке точек на [-1, 1]
        n_max (int): Максимальный порядок разложения

    Возвращает:
        ndarray: Коэффициенты разложения [f^(0), f^(1), ..., f^(n_max)]
    """
    n_points = len(f)
    x = np.linspace(-1, 1, n_points)
    coeffs = np.zeros(n_max+1)

    for n in range(n_max+1):
        p_n = legendre_polynomial(n, x)
    # Численное интегрирование с использованием квадратурной формулы
    coeffs[n] = (2*n + 1) / 2 * np.sum(f * p_n) * 2 / n_points

    return coeffs

def reconstruct_function(coeffs, x):
    """
```

Восстановление функции по коэффициентам разложения по полиномам Лежандра.

Параметры:

coeffs (ndarray): Коэффициенты разложения [f^(0), f^(1), ..., f^(n_max)]

x (ndarray): Точки, в которых нужно восстановить функцию

Возвращает:

ndarray: Значения восстановленной функции в точках x

"""

```
    f_reconstructed = np.zeros_like(x)
    for n, coef in enumerate(coeffs):
        f_reconstructed += coef * legendre_polynomial(n, x)
return f_reconstructed
```

Модуль формирования матрицы интегрального уравнения

```
python
import numpy as np
from scipy.integrate import quad
from .legendre import legendre_polynomial

def kernel_function(t, omega_0):
"""
    Ядро интегрального уравнения для фильтра с характеристикой
    ρ(ω) = (1 - |ω|/ω₀)θ(ω₀ - |ω|).
```

```
Параметры:
    t (float): Разность аргументов t - τ
    omega_0 (float): Частота отсечки фильтра

  Возвращает:
    float: Значение ядра K(t)
  """
  if abs(t) < 1e-10:  # Обработка случая t = 0 для избежания деления на 0
return omega_0 / np.pi
  else:
    return (np.sin(omega_0*t) / (np.pi*t) -
          (np.cos(omega_0*t) - 1) / (np.pi*omega_0*t**2))

def create_matrix(n, omega_0):
"""
  Формирование матрицы системы для интегрального уравнения.

  Параметры:
    n (int): Порядок аппроксимации
    omega_0 (float): Частота отсечки фильтра
```

```
    Возвращает:
        ndarray: Матрица системы размером (n+1, n+1)
    """
      matrix = np.zeros((n+1, n+1))

      for i in range(n+1):
        for j in range(n+1):
    # Вычисление элемента матрицы k_ij
            def integrand(x, y):
    return kernel_function(x - y, omega_0) * legendre_polynomial(i, x) * legendre_polynomial(j, y)

            # Двойноеинтегрирование
            def inner_integral(x):
              result, _ = quad(lambda y: integrand(x, y), -1, 1, limit=100)
              return result

            outer_integral, _ = quad(inner_integral, -1, 1, limit=100)
    matrix[i, j] = (2*i + 1) / 2 * outer_integral

      return matrix
```

```
def analyze_condition_number(matrix):
"""
    Анализ числа обусловленности матрицы.

    Параметры:
        matrix (ndarray): Матрица системы

    Возвращает:
        tuple: (число обусловленности, сингулярные числа)
"""
    u, s, vh = np.linalg.svd(matrix)
    condition_number = s[0] / s[-1]
    return condition_number, s
```

Модуль регуляризации и решения СЛАУ

```
python
import numpy as np
from scipy.optimize import minimize

def tikhonov_regularization(matrix, rhs, alpha, order=0):
"""
    Регуляризация Тихонова заданного порядка.
```

```
    Параметры:
        matrix (ndarray): Матрица системы
        rhs (ndarray): Правая часть системы (вектор)
        alpha (float): Параметр регуляризации
        order (int): Порядок регуляризации (0, 1 или 2)

    Возвращает:
        ndarray: Решение регуляризованной системы
    """
    m_t = matrix.T
    n = len(rhs)

    if order == 0:
        # Регуляризация нулевого порядка (минимизация нормы решения)
        reg_matrix = m_t @ matrix + alpha * np.eye(n)
    elif order == 1:
        # Регуляризация первого порядка (минимизация первой производной)
        D1 = np.zeros((n-1, n))
        for i in range(n-1):
            D1[i, i] = -1
            D1[i, i+1] = 1
        reg_matrix = m_t @ matrix + alpha * D1.T @ D1
    elif order == 2:
```

```
        # Регуляризация второго порядка (минимизация второй производной)
        D2 = np.zeros((n-2, n))
            for i in range(n-2):
                D2[i, i] = 1
                D2[i, i+1] = -2
                D2[i, i+2] = 1
            reg_matrix = m_t @ matrix + alpha * D2.T @ D2
        else:
            raise ValueError("Order must be 0, 1 or 2")

        reg_rhs = m_t @ rhs
        solution = np.linalg.solve(reg_matrix, reg_rhs)

        return solution

    def l_curve_method(matrix, rhs, alphas):
    """
        Метод L-кривой для выбора параметра регуляризации.

        Параметры:
            matrix (ndarray): Матрица системы
            rhs (ndarray): Правая часть системы
            alphas (ndarray): Массив значений параметра регуляризации для
анализа
```

```
    Возвращает:
        tuple: (оптимальное alpha, логарифмы норм решений, логарифмы норм невязок)
    """
    solution_norms = []
    residual_norms = []

    for alpha in alphas:
        solution = tikhonov_regularization(matrix, rhs, alpha)

        solution_norm = np.linalg.norm(solution)
        residual_norm = np.linalg.norm(matrix @ solution - rhs)

        solution_norms.append(np.log10(solution_norm))
        residual_norms.append(np.log10(residual_norm))

    # Поиск точки максимальной кривизны на L-кривой
    curvatures = []
    for i in range(1, len(alphas)-1):
        dx1 = residual_norms[i] - residual_norms[i-1]
        dy1 = solution_norms[i] - solution_norms[i-1]
        dx2 = residual_norms[i+1] - residual_norms[i]
        dy2 = solution_norms[i+1] - solution_norms[i]
```

```
  # Вычислениекривизны
      if dx1 == 0 or dx2 == 0 or dy1 == 0 or dy2 == 0:
curvatures.append(0)
      else:
          angle1 = np.arctan2(dy1, dx1)
          angle2 = np.arctan2(dy2, dx2)
curvatures.append(np.abs(angle2 - angle1))

# Индекс точки максимальной кривизны (с учетом граничных точек)
max_curve_idx = np.argmax(curvatures) + 1
   optimal_alpha = alphas[max_curve_idx]

   return optimal_alpha, solution_norms, residual_norms

def tsvd_method(matrix, rhs, k):
"""
   Метод усеченного сингулярного разложения (TSVD).

   Параметры:
      matrix (ndarray): Матрица системы
      rhs (ndarray): Правая часть системы
      k (int): Количество используемых сингулярных чисел
```

```
    Возвращает:
        ndarray: Решение с использованием TSVD
    """
    u, s, vh = np.linalg.svd(matrix, full_matrices=False)
    s_inv = np.zeros_like(s)
    s_inv[:k] = 1 / s[:k]

    solution = vh.T @ (s_inv * (u.T @ rhs))
    return solution

def landweber_method(matrix, rhs, tau, max_iter, tol=1e-6):
    """
    Метод итеративной регуляризации Ландвебера.

    Параметры:
        matrix (ndarray): Матрица системы
        rhs (ndarray): Правая часть системы
        tau (float): Шаговый множитель (0 < tau< 2/||matrix||²)
        max_iter (int): Максимальное число итераций
        tol (float): Допустимая погрешность
```

```
 Возвращает:
        ndarray: Решение методом Ландвебера
    """
    n = len(rhs)
    f = np.zeros(n)  # Начальное приближение
m_t = matrix.T

    for i in range(max_iter):
        residual = rhs - matrix @ f
        f_next = f + tau * (m_t @ residual)

# Проверка условия останова
        if np.linalg.norm(f_next - f) < tol:
return f_next

        f = f_next

return f
```

Модуль восстановления сигнала

```
python
import numpy as np
from .legendre import legendre_polynomial, reconstruct_function
```

```
def signal_to_normalized(t, signal, omega_0):
"""
    Преобразование сигнала из физических координат в нормированные.

    Параметры:
        t (ndarray): Массив значений времени
        signal (ndarray): Значения сигнала
        omega_0 (float): Частота отсечки фильтра

    Возвращает:
        tuple: (нормированные координаты, нормированный сигнал)
"""
    x = omega_0 * t
    return x, signal

def normalized_to_signal(x, normalized_signal, omega_0):
"""
```

```
    Преобразование сигнала из нормированных координат в физические.

    Параметры:
        x (ndarray): Нормированные координаты
        normalized_signal (ndarray): Значения сигнала в нормированных координатах
        omega_0 (float): Частота отсечки фильтра

    Возвращает:
        tuple: (физические координаты, сигнал в физических координатах)
    """
    t = x / omega_0
    return t, normalized_signal

def restore_signal(coeffs, t, omega_0):
    """
    Восстановление сигнала по коэффициентам разложения.

    Параметры:
        coeffs (ndarray): Коэффициенты разложения по полиномам Лежандра
        t (ndarray): Массив значений времени для восстановления
        omega_0 (float): Частота отсечки фильтра
```

```
    Возвращает:
        ndarray: Восстановленный сигнал в точках t
    """
    # Преобразование в нормированные координаты
    x = omega_0 * t

# Проверкадиапазона x
    if np.min(x) < -1 or np.max(x) > 1:
        raise ValueError("Values of omega_0*t must be in range [-1, 1]")

    # Восстановлениесигнала
    signal = reconstruct_function(coeffs, x)

    return signal

def calculate_error(f_exact, f_restored):
"""
    Вычисление относительной погрешности восстановления.

    Параметры:
        f_exact (ndarray): Точные значения сигнала
        f_restored (ndarray): Восстановленные значения сигнала
```

```
  Возвращает:
      float: Относительная погрешность в норме L2
"""
    error = np.linalg.norm(f_restored - f_exact) / np.linalg.norm(f_exact)
    return error

def calculate_residual(matrix, solution, rhs):
"""
    Вычисление относительной невязки.

    Параметры:
        matrix (ndarray): Матрица системы
        solution (ndarray): Решение системы
        rhs (ndarray): Правая часть системы

    Возвращает:
        float: Относительная невязка
    """
    residual = np.linalg.norm(matrix @ solution - rhs) / np.linalg.norm(rhs)
    return residual
```

Главный модуль программы

```
python
import numpy as np
import matplotlib.pyplot as plt
```

```
from .legendre import legendre_transform, reconstruct_function
from .matrix import create_matrix, analyze_condition_number
from .regularization import tikhonov_regularization, l_curve_method, tsvd_method, landweber_method
from .signal import signal_to_normalized, normalized_to_signal, restore_signal, calculate_error, calculate_residual

class SignalProcessor:
    """
    Класс для обработки сигналов и решения задачи восстановления.
    """

    def __init__(self, omega_0=2.0, epsilon=0.03, n_order=None, alpha=None):
    """
        Инициализация процессора сигналов.

        Параметры:
            omega_0 (float): Частота отсечки фильтра
            epsilon (float): Относительная точность измерения сигнала
            n_order (int): Порядок аппроксимации (если None, будет определен автоматически)
            alpha (float): Параметр регуляризации (если None, будет определен автоматически)
    """
    self.omega_0 = omega_0
```

```
        self.epsilon = epsilon
        self.n_order = n_order
        self.alpha = alpha
        self.matrix = None
        self.g_coeffs = None
        self.f_coeffs = None
        self.condition_number = None

    def process_output_signal(self, t, g):
        """
        Обработка выходного сигнала и подготовка к восстановлению.

        Параметры:
            t (ndarray): Значения времени
            g (ndarray): Значения выходного сигнала

        Возвращает:
            bool: True в случае успеха, False в случае ошибки
        """
        try:
            # Проверка диапазона t
            t_max = 1 / self.omega_0
        if np.min(t) < -t_max or np.max(t) > t_max:
                raise ValueError(f"Time values must be in range [-{t_max}, {t_max}]")
```

```
            # Преобразованиевнормированныекоординаты
                x, g_norm = signal_to_normalized(t, g, self.omega_0)

        # Определение оптимального порядка аппроксимации, если не задан
        if self.n_order is None:
        self.n_order = self._determine_optimal_n(g_norm)

                # Вычислениекоэффициентовразложениявыходногосигнала
        self.g_coeffs = legendre_transform(g_norm, self.n_order)

                # Формированиематрицысистемы
        self.matrix = create_matrix(self.n_order, self.omega_0)

                # Анализчислаобусловленности
        self.condition_number,                    self.singular_values                    =
analyze_condition_number(self.matrix)

                return True
            except Exception as e:
        print(f"Error in process_output_signal: {e}")
                return False

            def     reconstruct_input_signal(self,     regularization_method='tikhonov',
reg_order=0):
                """
```

```
        Восстановлениевходногосигнала.

        Параметры:
            regularization_method (str): Методрегуляризации ('tikhonov', 'tsvd',
'landweber')
        reg_order (int): Порядок регуляризации Тихонова (0, 1 или 2)

        Возвращает:
            bool: True в случае успеха, False в случае ошибки
        """
        try:
            if self.matrix is None or self.g_coeffs is None:
                raise ValueError("Process output signal first")

            # Выборметодарегуляризации
            if regularization_method == 'tikhonov':
        # Определение параметра регуляризации, если не задан
        if self.alpha is None:
                    alphas = np.logspace(-8, -1, 20)
        self.alpha, _, _ = l_curve_method(self.matrix, self.g_coeffs, alphas)

                # РешениесрегуляризациейТихонова
        self.f_coeffs = tikhonov_regularization(
        self.matrix, self.g_coeffs, self.alpha, order=reg_order)
```

```
elif regularization_method == 'tsvd':
            # Определение числа сингулярных значений для TSVD
k = int(0.7 * len(self.singular_values))
self.f_coeffs = tsvd_method(self.matrix, self.g_coeffs, k)

        elif regularization_method == 'landweber':
            # МетодЛандвебера
            norm_matrix = np.linalg.norm(self.matrix, ord=2)
            tau = 1.8 / (norm_matrix ** 2)
self.f_coeffs = landweber_method(
self.matrix, self.g_coeffs, tau, max_iter=100)

        else:
            raise ValueError("Unknown regularization method")

        return True
    except Exception as e:
print(f"Error in reconstruct_input_signal: {e}")
        return False

  def get_reconstructed_signal(self, t):
"""
    Получение восстановленного сигнала в заданных точках.
```

```
    Параметры:
        t (ndarray): Значения времени

    Возвращает:
ndarray: Восстановленныйсигнал
    """
    if self.f_coeffs is None:
        raise ValueError("Reconstruct input signal first")

    return restore_signal(self.f_coeffs, t, self.omega_0)

  def evaluate_reconstruction(self, t, f_exact, f_restored=None):
"""
    Оценка качества восстановления.

    Параметры:
        t (ndarray): Значения времени
        f_exact (ndarray): Точный входной сигнал
        f_restored (ndarray): Восстановленный сигнал (если None, вычисляется)

    Возвращает:
        float: Относительная погрешность восстановления
    """
if f_restored is None:
```

```
    f_restored = self.get_reconstructed_signal(t)

        return calculate_error(f_exact, f_restored)

    def _determine_optimal_n(self, g_norm):
"""
        Определение оптимального порядка аппроксимации.

        Параметры:
            g_norm (ndarray): Нормированный выходной сигнал

        Возвращает:
            int: Оптимальный порядок аппроксимации
        """
        # Эмпирическая формула, зависящая от точности ε и частоты отсечки ω₀
    n_max = int(20 / self.epsilon ** 0.4 * self.omega_0 ** 0.5)

    # Ограничение для обеспечения устойчивости
        n_max = min(n_max, 50)

        # Дополнительная проверка сходимости разложения
        x = np.linspace(-1, 1, len(g_norm))
    errors = []
```

```
   for n in range(5, n_max + 1, 5):
            coeffs = legendre_transform(g_norm, n)
            g_approx = reconstruct_function(coeffs, x)
            error = np.linalg.norm(g
_determine_optimal_n продолжениефункции:
            g_approx = reconstruct_function(coeffs, x)
            error = np.linalg.norm(g_approx - g_norm) / np.linalg.norm(g_norm)
errors.append(error)

            # Если достигнута требуемая точность, прекращаем
if error < 0.8 * self.epsilon:
                return n

# Если не нашли оптимальное n, возвращаем максимальное
return n_max

   def plot_results(self, t, f_exact=None, g=None):
"""
      Построение графиков результатов восстановления.

      Параметры:
         t (ndarray): Значения времени
         f_exact (ndarray): Точный входной сигнал (если известен)
g (ndarray): Выходнойсигнал
      """
```

```
  f_restored = self.get_reconstructed_signal(t)

plt.figure(figsize=(12, 8))

      # График восстановленного сигнала
plt.subplot(2, 1, 1)
plt.plot(t, f_restored, 'b-', linewidth=2, label='Восстановленный сигнал')

if f_exact is not None:
plt.plot(t, f_exact, 'k--', linewidth=1.5, label='Точныйсигнал')
          error = self.evaluate_reconstruction(t, f_exact, f_restored)
plt.title(f'Восстановление сигнала (погрешность: {error:.4f})')
      else:
plt.title('Восстановленный сигнал')

plt.xlabel('t')
plt.ylabel('f(t)')
plt.grid(True)
plt.legend()

      # Графиквыходногосигнала
      if g is not None:
plt.subplot(2, 1, 2)
plt.plot(t, g, 'r-', linewidth=2, label='Выходнойсигнал g(t)')
```

```
        # Прямое преобразование восстановленного сигнала
            # (требует дополнительной реализации)

        plt.xlabel('t')
        plt.ylabel('g(t)')
        plt.grid(True)
        plt.legend()

        plt.tight_layout()
        plt.show()

        def main():
          """
          Пример использования класса SignalProcessor для восстановления сигнала.
          """
          # Параметры задачи
          omega_0 = 2.0
          epsilon = 0.03

          # Создание тестового сигнала (гауссиан)
        t = np.linspace(-1/omega_0, 1/omega_0, 1000)
          f_exact = np.exp(-5 * (omega_0*t)**2)
```

```
        # Формирование выходного сигнала (прямая задача)
        x = omega_0 * t
          g_exact = np.zeros_like(t)

          for i, ti in enumerate(t):
             for j, tj in enumerate(t):
                g_exact[i] += kernel_function(ti - tj, omega_0) * f_exact[j] * (t[1] -
t[0])

        # Добавление шума
        np.random.seed(42)
        noise = epsilon * np.max(np.abs(g_exact)) * np.random.randn(len(g_exact))
        g = g_exact + noise

          # Создание и использование процессора сигналов
        processor = SignalProcessor(omega_0=omega_0, epsilon=epsilon)
        processor.process_output_signal(t, g)
        processor.reconstruct_input_signal(regularization_method='tikhonov',
reg_order=1)
          f_restored = processor.get_reconstructed_signal(t)

          # Оценкарезультатов
          error = processor.evaluate_reconstruction(t, f_exact, f_restored)
        print(f"Относительная погрешность восстановления: {error:.4f}")
        print(f"Число обусловленности матрицы:
{processor.condition_number:.2e}")
```

```
print(f"Выбранный порядок аппроксимации: {processor.n_order}")
print(f"Параметр регуляризации: {processor.alpha:.2e}")

# Построениеграфиков
processor.plot_results(t, f_exact, g)

if __name__ == '__main__':
main()
```