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 линейной фильтрации

Обучающийся Н. М. Лёшина
(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

Название выпускной квалификационной работы: Численное решение интегрального уравнения Фредгольма первого рода в задаче линейной фильтрации.

Выпускная квалификационная работа состоит из введения, трёх глав, 11 рисунков, 1 таблицы, заключения и списка литературы из 30 источников, включая 5 на иностранном языке.

Целью работы является разработка и реализация численного метода решения интегрального уравнения Фредгольма первого рода в условиях зашумлённых входных данных.

Объектом исследования является интегральное уравнение Фредгольма первого рода.

Предметом исследования является численный алгоритм, включающий аппроксимацию ядра, регуляризацию Тихонова и реализацию на языке Python.

Ключевая задача — анализ влияния шума, частоты среза и параметров регуляризации на устойчивость решения.

Выпускная квалификационная работа включает три логически взаимосвязанных раздела.

Первая глава посвящена постановке задачи, выводу уравнения с использованием преобразования Фурье и рассмотрению причин её некорректности.

Во второй главе рассматриваются спектральные свойства компактных операторов и методы регуляризации.

В третьей главе изложен численный метод: аппроксимация вырожденным ядром, приведение к системе линейных алгебраических уравнений, разработка алгоритма и его реализация на языке Python.

В заключение отмечается, что предложенный метод демонстрирует устойчивость в условиях зашумлённых данных и может применяться в прикладных задачах обработки сигналов.

Abstract

The title of the graduation work is Numerical Solution of the Fredholm Integral Equation of the First Kind in the Linear Filtering Problem.

The graduation work consists of an introduction, three parts, 11 figures, 1 table, a conclusion, and a list of 30 references including 5 foreign sources.

The aim of the work is to develop and implement a numerical method for solving the Fredholm integral equation of the first kind under noisy data conditions.

The object of the research is the Fredholm integral equation of the first kind.

The subject is a numerical algorithm that includes kernel approximation, Tikhonov regularization, and implementation in Python.

The key issue is the analysis of how noise, cutoff frequency, and regularization parameters affect solution stability.

The graduation work includes three logically connected parts.

The first chapter focuses on the problem statement, derivation of the equation using the Fourier transform, and analysis of the reasons for its ill-posedness.

The second chapter covers spectral properties of compact operators and regularization methods.

The third chapter presents the numerical method: degenerate kernel approximation, reduction to a system of linear algebraic equations, algorithm development, and Python implementation.

In conclusion, the proposed method demonstrates robustness under noisy conditions and is applicable to practical signal processing tasks.

Содержание

Введение

Линейный фильтр, который преобразует исходный сигнал $f(t)$ в сигнал $g(t)$, так, что частотные спектры $F(\omega)$ и $G(\omega)$ функций $f(t)$ и $g(t)$ соответственно, связаны соотношением:

$$G(\omega) = \rho(\omega)F(\omega), \quad (1)$$

где

$$\rho(\omega) = e^{-\frac{|\omega|}{\omega_0}}\theta(\omega_0 - |\omega|), \quad (2)$$

где $\omega_0 \in [0.1, 10] -$ частота отсечки,

$\theta(\omega)$ — функция Хевисайда, ограничивающая частотный диапазон фильтра $\rho(\omega)$.

Измерения сигнала $g(t)$ проводятся в ограниченном временном интервале $t \in \left[-\frac{1}{\omega_0}, \frac{1}{\omega_0}\right]$, функция $g(t)$ удовлетворяет условию $g\left(\pm\frac{1}{\omega_0}\right) = 0$.

Предметом исследования является вывод интегрального уравнения Фредгольма первого рода, связывающего сигналы $f(t)$ и $g(t)$ с использованием преобразования Фурье, а также последующее численное исследование этого уравнения для различных видов функции $g(t)$. Важной особенностью данной задачи является её некорректность: решение уравнения Фредгольма первого рода существенно зависит от ошибок в исходных данных, что требует применения специальных методов стабилизации и регуляризации.

Задачи линейной фильтрации, формализуемые в виде интегральных уравнений Фредгольма, имеют широкое практическое применение в технических и естественнонаучных дисциплинах. Такие задачи встречаются в радиофизике, биомедицинской инженерии, геофизике, цифровой обработке сигналов и изображений. Корректное восстановление исходного сигнала по

зашумлённым измерениям особенно актуально в условиях растущего объёма данных и требований к их точности.

Применение современных численных методов и реализация алгоритмов на языке Python позволяют создавать гибкие программные решения, адаптируемые к различным типам входных данных и требуемой точности. Это делает разработку эффективных вычислительных схем для некорректных задач особенно актуальной в контексте цифровизации научных и инженерных процессов.

Целью данной работы является исследование и реализация численных методов решения интегрального уравнения Фредгольма первого рода и проведение сравнительного анализа их эффективности при разных значениях входных параметров.

Для достижения поставленной цели были поставлены следующие задачи:

- проведение литературного обзора методов решения некорректных задач, их особенностей и подходов к регуляризации;
- разработка алгоритма, включающего аппроксимацию ядра интегрального уравнения вырожденным ядром порядка N и сведение задачи к решению системы линейных алгебраических уравнений (СЛАУ). Число N определяется точностью ϵ с использованием разложения:

$$g\left(\frac{x}{\omega_0}\right) = \sum_{n=0}^{\infty} g(n)\sin(\pi n x) \approx \sum_{n=0}^{N} g(n)\sin(\pi n x)\,, x \in [-1, 1];$$

- реализация численного решения, включая анализ числа обусловленности матрицы СЛАУ;
- построение функции $f(t)$ по найденному вектору коэффициентов $f(n)$ и визуализация результатов в виде графиков;

- сравнение решений с использованием регуляризации и без неё, анализ эффективности методов при разных значениях ϵ $(1\%-10\%)$ и параметра регуляризации.

Выпускная квалификационная работа состоит из трёх основных разделов.

Первый раздел посвящен постановке задачи, выводу уравнения с использованием преобразования Фурье и рассмотрению причин её некорректности.

Во втором разделе рассматриваются спектральные свойства компактных операторов и методы регуляризации.

В третьем разделе изложен численный метод: аппроксимация вырожденным ядром, приведение к системе линейных алгебраических уравнений, разработка алгоритма и его реализация на языке Python.

1 Общая постановка задачи и теоретические основы

1.1 Актуальность исследования

Решение интегральных уравнений играет ключевую роль во многих областях науки и техники, включая обработку сигналов, теорию управления, медицинскую томографию и геофизику. В частности, интегральные уравнения Фредгольма первого рода часто возникают в обратных задачах, где требуется восстановить исходные данные на основе косвенных измерений. Однако такие задачи, как правило, некорректно поставлены, что означает их высокую чувствительность к ошибкам во входных данных, и требуют применения специализированных методов стабилизации для получения устойчивого решения.

Примером таких задач является линейная фильтрация сигналов, где исходный сигнал $f(t)$ преобразуется в сигнал $g(t)$, а их спектры частот $F(\omega)$ и $G(\omega)$ связаны уравнением согласно формуле (1).

Эта постановка задачи приводит к интегральному уравнению Фредгольма первого рода, решение которого требует использования специализированных численных методов и техник регуляризации.

Актуальность данного исследования обусловлена совокупностью факторов. Прежде всего, практическая значимость задачи линейной фильтрации сигналов проявляется в её широком применении в таких областях, как радиоэлектроника, обработка изображений и медицинская диагностика [19,20]. Совершенствование методов решения интегральных уравнений способствует повышению точности восстановления сигналов и устойчивости алгоритмов обработки данных. Кроме того, сама постановка задачи носит некорректный характер — малые возмущения входных данных могут существенно влиять на результат, что требует использования специальных методов регуляризации, включая регуляризацию Тихонова и аппроксимацию вырожденным ядром. Немаловажно и то, что развитие современных

вычислительных алгоритмов позволяет более эффективно решать интегральные уравнения, преобразуя их в системы линейных алгебраических уравнений с сохранением точности и устойчивости. Дополнительную ценность представляет возможность анализа влияния различных параметров, таких как уровень шума и частота отсечки, на результат численного решения, что даёт основания для выбора наиболее подходящих подходов в зависимости от специфики задачи.

Таким образом, исследование численных методов решения интегральных уравнений Фредгольма первого рода в контексте линейной фильтрации является как теоретически, так и практически значимым, делая его актуальной и важной областью современной математики и прикладных наук.

1.2 Цель и задачи работы

В данной работе основное внимание уделяется разработке и применению численных методов для решения интегральных уравнений Фредгольма первого рода в контексте линейной фильтрации. Важной частью исследования является оценка эффективности различных подходов, что позволяет определить их применимость при различных входных параметрах.

Прежде чем приступить к разработке алгоритмов, необходимо провести всесторонний анализ существующих методов решения некорректно поставленных задач, уделяя особое внимание техникам регуляризации. Одним из ключевых этапов является вывод интегрального уравнения Фредгольма первого рода, которое описывает связь сигналов в пространстве Фурье. Для численного решения данной задачи предлагается метод аппроксимации ядра уравнения вырожденным ядром порядка N, что приводит к сведение проблемы к решению системы линейных алгебраических уравнений. Точность аппроксимации определяется параметрами разложения, что требует тщательного анализа.

После формализации подхода следует реализация численного метода, включающая оценку числа обусловленности матрицы соответствующей системы уравнений. Полученные коэффициенты позволяют построить целевую функцию, визуализировать результаты и проанализировать их корректность. Финальный этап исследования посвящен сравнению решений, полученных с регуляризацией и без нее, а также оценке эффективности предложенных методов при различных значениях параметров. Особое внимание уделяется влиянию регуляризации на точность вычислений.

Результаты данной работы не только углубляют понимание численных методов решения интегральных уравнений, но и предоставляют рекомендации по их практическому применению в обработке сигналов и смежных областях.

1.3 Область применения задачи решения интегральных уравнений

Интегральные уравнения Фредгольма первого рода имеют широкий спектр применений в различных научных и инженерных дисциплинах. Их важность обусловлена частым появлением в обратных задачах, где используются косвенные измерения для восстановления неизвестных функций. Они играют ключевую роль в обработке сигналов и изображений, где помогают восстанавливать данные, устранять размытие и решать задачи томографической реконструкции. В геофизике и дистанционном зондировании эти уравнения используются для исследования подземных структур, восстановления атмосферных и океанических данных, а также обработки гравиметрической информации.

В области оптики и распространения волн они позволяют анализировать дифракционные процессы, поддерживают методы когерентной томографии и помогают решать задачи фазового восстановления. Медицинские и биологические приложения включают анализ сигналов, обратные задачи в

визуализации и моделирование свойств тканей, что играет важную роль в диагностике и исследовании.

Физические и инженерные задачи также требуют использования этих уравнений, особенно при изучении теплопроводности, неразрушающем контроле материалов и решении обратных задач квантовой механики. В сфере финансов и экономики они применяются для моделирования ценообразования, оценки рисков и анализа исторических данных.

Многообразие применения интегральных уравнений Фредгольма первого рода подчеркивает их значимость в различных областях, акцентируя необходимость разработки надежных численных методов и техник регуляризации для обеспечения устойчивых и точных решений. Исследуемые методы в данной работе способствуют совершенствованию вычислительных подходов в этих приложениях, предоставляя ценные инструменты для исследователей и практиков.

1.4 Свойства спектра вполне непрерывных операторов и некорректность задачи Фредгольма первого рода

Интегральные уравнения Фредгольма первого рода играют значительную роль как в математической теории, так и в прикладных исследованиях. Их специфика связана с наличием компактных интегральных операторов, обладающих сложными спектральными свойствами, что существенно усложняет процесс решения. Даже небольшие возмущения входных данных могут приводить к значительным отклонениям результатов, что требует особого подхода к анализу и вычислениям.

Компактные операторы характеризуются тем, что переводят ограниченные множества в относительно компактные, формируя дискретный спектр с точкой накопления в нуле. Наличие малых собственных значений усложняет численные вычисления и требует применения стабилизационных методов. Помимо этого, спектр таких операторов обычно убывает по геометрическому

закону, усиливая влияние ошибок округления, что может существенно исказить конечные вычисления. Высокая чувствительность к шумам во входных данных делает эти задачи некорректно поставленными, поскольку даже незначительные изменения параметров могут сильно повлиять на получаемые решения. Низкая обусловленность исключает возможность применения стандартных численных методов без предварительной стабилизации.

Для корректной работы с такими задачами необходимо использование алгоритмов, обеспечивающих регуляризацию и повышающих устойчивость вычислений. Некорректность в контексте интегральных уравнений Фредгольма первого рода проявляется в неоднозначности решений, даже при точных данных, что требует дополнительных ограничений или предположений. Численные методы сталкиваются с нестабильностью, обусловленной высокими числами обусловленности матриц, особенно при аппроксимации ядра, что ведет к накоплению ошибок вычислений. Кроме того, сложность построения обратного оператора определяется малыми собственными значениями интегрального оператора, делая его обращение невозможным без дополнительных корректировок. Таким образом, разработка эффективных методов решения этих уравнений требует детального анализа спектральных свойств операторов и применения регуляризационных методик для повышения надежности вычислительных процедур.

1.5 Методы решения некорректных задач

Интегральные уравнения Фредгольма первого рода часто возникают в математическом моделировании и различных прикладных задачах [1]. Эти уравнения имеют вид:

$$g(t) = \int_a^b K(t,s)\, f(s)\, ds,$$

где $K(t,s)$ – ядро уравнения,

$f(s)$ – искомая функция,

$g(t)$ – известная функция,

a и b – границы интегрирования.

Основная сложность решения таких уравнений заключается в их некорректности: решение может не существовать, быть неединственным или не зависеть непрерывно от входных данных [4]. Для решения этих задач применяются различные методы регуляризации и численные алгоритмы.

Метод Тихонова. Метод регуляризации Тихонова является одним из самых распространенных подходов к решению некорректных задач [15]. В этом методе к исходному уравнению добавляется дополнительный член, который стабилизирует решение:

$$\|Kf - g\|^2 + \alpha\|f\|^2 \to min,$$

где $\alpha\|f\|^2$ – параметр регуляризации.

Этот параметр регулирует баланс между точностью и устойчивостью решения. Метод Тихонова позволяет получить стабильное решение даже при наличии шума в данных.

Метод усеченного сингулярного разложения (TSVD). Метод усеченного сингулярного разложения (TSVD) основывается на сингулярном разложении матрицы ядра K:

$$K = U\Sigma V^T,$$

где U и V – ортогональные матрицы,

Σ – диагональная матрица сингулярных значений.

В методе TSVD исключаются компоненты с малыми сингулярными значениями, что уменьшает влияние шума и стабилизирует решение:

$$f_{TSVD} = V\Sigma^{-1}U^T g.$$

Этот метод позволяет эффективно бороться с некорректностью задачи, сохраняя при этом точность решения [10].

Итеративные методы. Итеративные методы, такие как метод регуляризации Ландвебера, также используются для решения интегральных уравнений Фредгольма первого рода. Эти методы предполагают последовательное уточнение решения:

$$f_{n+1} = f_n + \alpha(g - Kf_n),$$

где α – параметр регуляризации.

Итеративные методы позволяют получить устойчивое решение за счет постепенного уменьшения влияния шума [8].

Методы минимизации остатка. Решение уравнения итерационно, при этом минимизируется остаточная ошибка:

$$\| Af - g \|.$$

Итерационный процесс позволяет улучшить устойчивость вычислений и точность решения [5].

Методы аппроксимации ядра. Методы аппроксимации ядра предполагают представление ядра уравнения в виде конечной суммы функций, что позволяет перевести задачу в форму СЛАУ и упростить её решение [2].

Эти методы обеспечивают высокую точность аппроксимации и устойчивость решения.

Выводы по разделу.

В первой главе рассмотрены теоретические основы задачи решения интегральных уравнений Фредгольма первого рода. Обоснована актуальность темы, сформулированы цель и задачи исследования, определены области применения. Проанализированы свойства вполне непрерывных операторов и характер некорректности рассматриваемых задач. Особое внимание уделено современным методам их решения, включая регуляризацию Тихонова, усечённое сингулярное разложение, итеративные методы и аппроксимацию ядра. Полученные теоретические положения являются фундаментом для построения численного алгоритма, представленного в следующем разделе.

2 Вывод интегрального уравнения и его численная аппроксимация

2.1 Вывод интегрального уравнения из условий задачи

В условиях задачи задана Эта связь задаётся соотношением (1), отражающим действие фильтрации, где функция фильтра описывается выражением (2). Параметр $\omega_0 \in [0.1, 10]$ представляет собой частоту отсечки, а использование функции Хевисайда ограничивает диапазон пропускаемых частот.

Перейдем из частотной области во временную область с использованием обратного преобразования Фурье:

$$g(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} G(\omega)e^{i\omega t}\, d\omega\,.$$

Подставим выражение для $G(\omega)$:

$$g(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} \rho(\omega)F(\omega)e^{i\omega t}\, d\omega. \qquad (3)$$

Так как $\rho(\omega) = 0$ при $|\,\omega\,| \leq \omega_{0,}$ пределы интегрирования можно ограничить:

$$g(t) = \frac{1}{2\pi}\int_{-\omega_0}^{\omega_0} \rho(\omega)F(\omega)e^{i\omega t}\, d\omega.$$

Представим $F(\omega)$ через $f(s)$:

$$F(\omega) = \frac{1}{2\pi}\int_{-\infty}^{\infty} f(s)e^{-i\omega t}\,ds.$$

Подставим это в $g(t)$:

$$g(t) = \int_{-\infty}^{\infty} f(s)\left(\frac{1}{2\pi}\int_{-\omega_0}^{\omega_0}\rho(\omega)e^{i\omega(t-s)}\,ds\right)e^{i\omega s}d\omega.$$

Внутренний интеграл зависит от $t - s$, введем ядро:

$$K(t-s) = \frac{1}{2\pi}\int_{-\omega_0}^{\omega_0}\rho(\omega)e^{i\omega(t-s)}\,ds,$$

Тогда интегральное уравнение Фредгольма первого рода имеет вид:

$$g(t) = \int_{-\infty}^{\infty} K(t-s)f(s)\,ds\,.$$

2.2 Переход от Фурье-образов к интегральному уравнению свертки

Переход от Фурье-образов к уравнению свёртки позволяет привести задачу к виду, удобному для численного анализа [3].

Дана зависимость спектров формуле (1) и обратное преобразование Фурье по формуле (3).

Подставим представление для $F(\omega)$:

$$F(\omega) = \frac{1}{2\pi}\int_{-\infty}^{\infty} f(s)e^{-i\omega t}\,ds,$$

$$g(t) = \int_{-\infty}^{\infty} f(s)\left(\frac{1}{2\pi}\int_{-\infty}^{\infty}\rho(\omega)e^{i\omega(t-s)}d\omega\right)ds.$$

Поменяем порядок интегрирования:

$$g(t) = \int_{-\infty}^{\infty} f(s) \left(\frac{1}{2\pi} \int_{-\infty}^{\infty} \rho(\omega) e^{i\omega(t-s)} d\omega \right) ds.$$

Так как $\rho(\omega) = 0$ при $| \omega | \leq \omega_0$, то:

$$K(t-s) = \frac{1}{2\pi} \int_{-\omega_0}^{\omega_0} e^{-\frac{|\omega|}{\omega_0}} e^{i\omega(t-s)} \, d\omega,$$

Уравнение примет вид:

$$g(t) = \int_{-\infty}^{\infty} K(t-s) f(s) \, ds,$$

что соответствует классической свёрточной форме, подробно рассматриваемой в источнике [7].

2.3 Ограничение пределов интеграла и формулировка конечной задачи

Для получения аналитического выражения ядра рассмотрим:

$$2\pi K(x) = \int_{-\omega_0}^{\omega_0} exp\left(-\frac{| \omega |}{\omega_0} \right) exp(i\omega x) \, d\omega,$$

Разобьем интеграл:

$$2\pi K(x) = \int_{0}^{\omega_0} exp\left[\omega \left(\frac{1}{\omega_0} + ix \right) \right] d\omega + \int_{-\omega_0}^{0} exp\left[\omega \left(\frac{1}{\omega_0} + ix \right) \right] d\omega.$$

Вычисления приводят к результату:

$$2\pi K(x) = \frac{2\omega_0}{1+(\omega_0 x)^2}\left[1 + \frac{1}{e}(-\cos(\omega_0 x) + \omega_0 x \sin(\omega_0 x))\right] d\omega. \qquad (4)$$

Далее будем использовать обозначение ядра как $K(t-s)$ с отсылкой к формуле (4).

Перейдем к конечной задаче:

$$g(t) = \int_{-\frac{1}{\omega_0}}^{\frac{1}{\omega_0}} K(t,s) f(s)\, ds\,,$$

Аппроксимация ядра:

$$f(s) \approx \sum_{i=1}^{N} f_n \sin\left(\frac{\pi n s}{\omega_0}\right), \qquad (5)$$

Матрица СЛАУ:

$$K_{mn} = \int_{-\frac{1}{\omega_0}}^{\frac{1}{\omega_0}} \int_{-\frac{1}{\omega_0}}^{\frac{1}{\omega_0}} K(x-y) \sin\left(\frac{\pi m x}{\omega_0}\right) \sin\left(\frac{\pi n y}{\omega_0}\right) dxdy, \qquad (6)$$

Для стабилизации решения применяется регуляризация Тихонова:

$$K \to K + \alpha I, \qquad (7)$$

где $\alpha > 0$ – параметр регуляризации,

I – единичная матрица.

Для проверки численной устойчивости матрицы K и регуляризованной матрицы $K + \alpha$ применяется вычисление числа обусловленности: $Cond(K), Cond(K + \alpha)$.

2.4 Аппроксимация ядра и переход к уравнению с вырожденным ядром

Аппроксимация ядра выполняется через представление искомой функции в виде ряда по синусам, согласно формуле (5) [6].

Для приведения интегрального уравнения Фредгольма первого рода к системе линейных алгебраических уравнений используется ортогональность синусов [12]. Функция $g(t)$ разлагается аналогично:

$$g(t) \approx \sum_{n=1}^{N} g_n \sin\left(\frac{\pi n s}{\omega_0}\right).$$

Умножая обе части уравнения на $\left(\frac{\pi n x}{\omega_0}\right)$ и интегрируя, получаем выражение для коэффициента g_n:

$$g_n = \int_{-\omega_0}^{\omega_0} g(t) \sin\left(\frac{\pi n t}{\omega_0}\right) gt\,.$$

Матрица K, возникающая при приведении уравнения к дискретному виду, формируется с использованием значений, полученных из аппроксимации ядра по формуле (5).

Тогда система принимает матричный вид:

$$Kf = g. \qquad (8)$$

Для обеспечения вычислительной устойчивости применяется регуляризация Тихонова (7).

Оценка численной устойчивости системы производится через вычисление числа обусловленности [13]:

$$Cond(K) = \frac{\lambda_{max}}{\lambda_{min}}, Cond(K+\alpha) = \frac{\lambda_{max}+\alpha}{\lambda_{min}+\alpha}, \quad (9)$$

где $\lambda_{max}, \lambda_{min}$ — собственные значения матрицы K.

Окончательное представление решения остаётся:

$$f(s) = \sum_{i=1}^{N} f_n \sin\left(\frac{\pi n s}{\omega_0}\right),$$

где коэффициенты f_n находятся из решения регуляризованной системы линейных уравнений.

Число базисных функций N выбирается таким образом, чтобы погрешность:

$$\delta = \| g - Kf \|, \quad (10)$$

была меньше заранее установленного значения точности ϵ.

Таким образом, полученная система линейных алгебраических уравнений с регуляризацией обеспечивает устойчивость вычислений даже при наличии шумов в данных. Варьируя число базисных функций N, можно контролировать точность приближённого решения и балансировать между вычислительной сложностью и надёжностью результата. Такой подход делает метод гибким и адаптируемым к различным классам задач, а также позволяет эффективно использовать его в практических приложениях, включая обработку экспериментальных данных и решение обратных задач.

Выводы по разделу.

Во второй главе из условий задачи линейной фильтрации получено интегральное уравнение Фредгольма первого рода. Показано, как переход к уравнению свёртки и ограничение пределов интегрирования позволяет привести задачу к конечному и практически реализуемому виду. Проведена аппроксимация ядра с помощью конечного набора функций, в результате чего уравнение сведено к уравнению с вырожденным ядром. Это преобразование создаёт основу для дальнейшего численного решения задачи в программной реализации. Кроме того, использование аппроксимации позволяет гибко контролировать точность решения и адаптировать модель под различные параметры фильтрации. Полученные формулы обеспечивают устойчивость и позволяют применять современные методы регуляризации, что делает данный подход пригодным для широкого круга задач, связанных с обработкой зашумлённых сигналов и решением некорректных задач восстановления.

3 Разработка и реализация алгоритма решения

3.1 Разработка алгоритма

Разработка алгоритма решения интегрального уравнения Фредгольма первого рода включает чёткие этапы, которые обеспечивают точность, устойчивость и эффективность численного подхода. Каждый шаг строго следит за математической структурой задачи.

Этап 1. Инициализация параметров.

Задаются основные параметры задачи, такие как:

- частота отсечки ω_0, которая ограничивает спектр;
- точность аппроксимации ϵ, чтобы контролировать погрешность разложения функции;
- регуляризационный параметр $\alpha > 0$, необходимый для стабилизации решения;
- максимальное число базисных функций N_{max}, чтобы избежать чрезмерной вычислительной нагрузки.

Этап 2. Вычисление нормы $\| g \|$.

Для известной функции $g(t)$ вычисляется её норма:

$$\|g\|L_2 = \left(\int_{-\frac{1}{\omega_0}}^{\frac{1}{\omega_0}} |g(t)|^2 \, dt \right)^{\frac{1}{2}}.$$

Этап 3. Определение числа базисных функций *N*.

Подбирается минимальное значение N, при котором ошибка аппроксимации:

$$\delta_N = \frac{\| g - g_N \|}{\| g \|} \leq \epsilon,$$

а функция $\delta_N(t)$ представляется в виде ряда по синусам:

$$\delta_N(t) = \sum_{n=1}^{N} c_n \sin\left(\frac{\pi n t}{\omega_0}\right), c_n = \int_{-\frac{1}{\omega_0}}^{\frac{1}{\omega_0}} g(t) \sin\left(\frac{\pi n t}{\omega_0}\right).$$

Этап 4. Построение матрицы K_{mn}.

Элементы матрицы K вычисляются по формуле (6) и используются при решении системы.

Этап 5. Регуляризация системы.

Проводится регуляризация Тихонова согласно формуле (7), то есть к матрице K прибавляется регуляризационный член αI, стабилизирующий решение.

Этап 6. Решение СЛАУ.

Решается система линейных уравнений вида (8), в которой используется регуляризованная матрица. Вектор f содержит искомые коэффициенты разложения функции $f(s)$, а вектор g содержит коэффициенты разложения функции $g(t)$.

Этап 7. Восстановление функции $f(s)$.

Функция $f(s)$ восстанавливается по регуляризованному решению согласно формуле (9).

Этап 8. Оценка точности и устойчивости.

Для анализа точности и устойчивости численного решения рассчитываются значения погрешности восстановления в соответствии с выражением (10), а также определяются числа обусловленности согласно формуле (9).

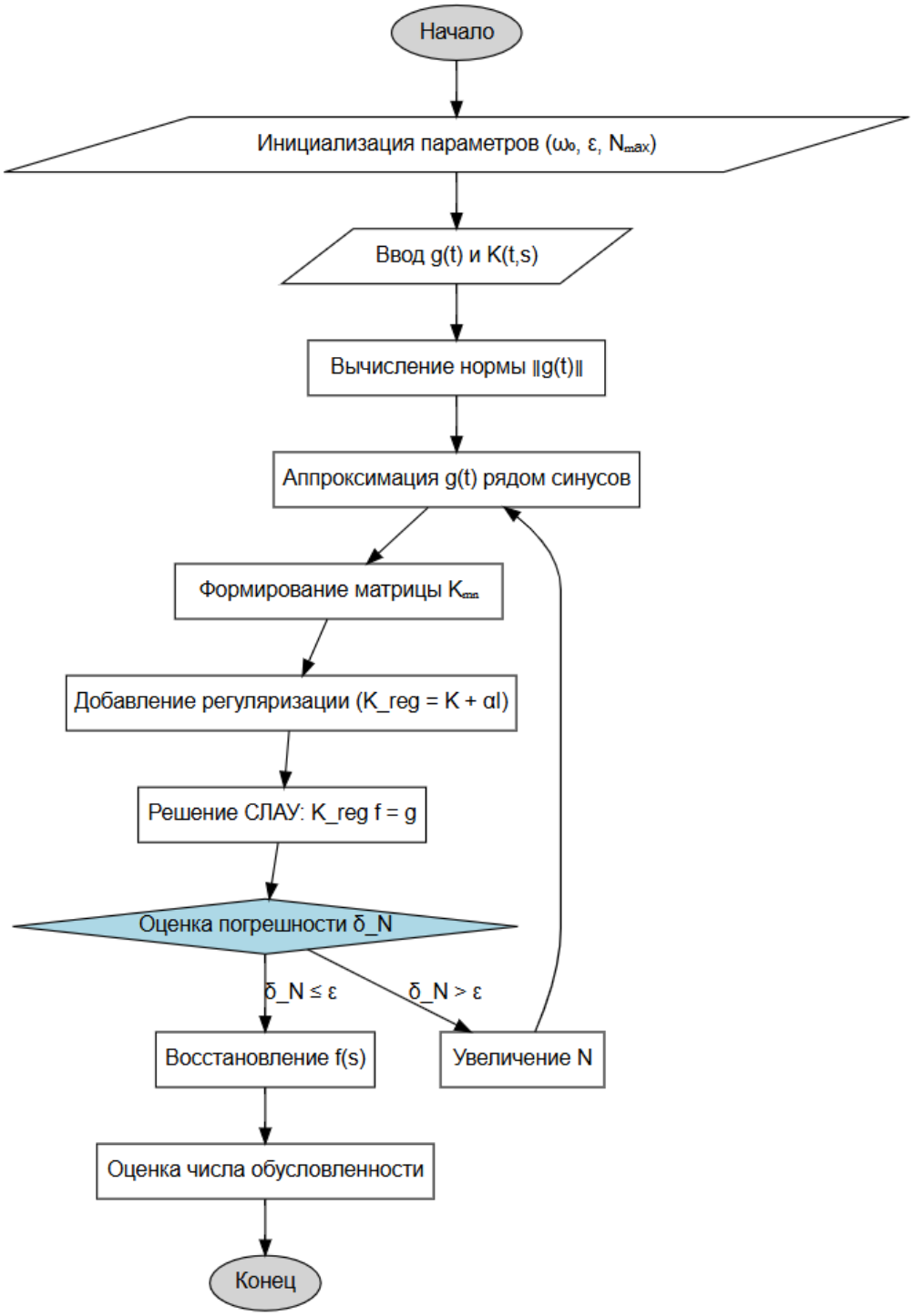


Рисунок 1 — Блок-схема алгоритма

3.2 Адаптация алгоритма к архитектуре вычислительной системы

Эффективное выполнение численных алгоритмов требует учёта особенностей современных вычислительных архитектур, включая использование многопроцессорных систем, оптимизацию оперативной памяти и применение специализированных библиотек. Такой подход позволяет минимизировать временные затраты, сократить объём задействованных ресурсов и повысить общую производительность алгоритмов при решении задач численного анализа, таких как аппроксимация функций и построение матриц ядра.

Параллельные вычисления. Современные многопроцессорные системы предоставляют возможность параллельного выполнения вычислительных операций, что особенно полезно при работе с большими объёмами данных.

Для вычисления матрицы ядра K_{mn} используются параллельные подходы, такие как многопоточность. Каждый элемент матрицы K_{mn} может быть вычислен независимо, так как он определяется локальными значениями интегралов. Это позволяет распределить работу между потоками.

Для реализации параллельных вычислений эффективно используются библиотеки Python, такие как multiprocessing [19], которые позволяют распределять задачи по ядрам процессора. Например, вычисление каждого элемента матрицы ядра можно обработать в отдельных потоках, значительно ускоряя выполнение программы.

В задачах, связанных с численным решением интегральных уравнений, матрица ядра может содержать большое число элементов, значения которых близки к нулю. В таких случаях для экономии оперативной памяти целесообразно использовать разреженные структуры данных, хранящиеся в форматах CSR (Compressed Sparse Row) или CSC (Compressed Sparse Column). Эти форматы позволяют ускорить операции умножения и решения линейных систем без потери точности, особенно при работе с крупными матрицами.

Подробное описание таких структур и методов обращения с интегральными уравнениями приведено в справочнике [11].

Вычисление интегралов, необходимых для формирования ядра $K(t-s)$ и элементов матрицы K_{mn}, требует высокой точности. Для их вычисления применяются методы, обеспечивающие стабильность и достоверность результатов. Одномерные интегралы рассчитываются с помощью scipy.integrate.quad, тогда как двойные интегралы вычисляются с использованием scipy.integrate.dblquad, что позволяет добиться требуемого уровня точности.

При решении систем линейных алгебраических уравнений выбор метода зависит от структуры матрицы ядра. В случае плотных матриц используется метод LU-разложения, реализованный в numpy.linalg.solve [14]. Если же матрица разреженная, применяются алгоритмы, оптимизированные для работы с разреженными структурами, например scipy.sparse.linalg.spsolve, что повышает эффективность вычислений и позволяет обрабатывать большие объемы данных.

Визуализация результатов осуществляется с помощью графических методов, позволяющих оценить качество восстановленной функции по сравнению с исходными данными. Построение графиков функций помогает выявить особенности модели, а анализ зависимости ошибок аппроксимации от числа базисных функций даёт представление о влиянии точности на вычислительную сложность задачи.

Оценка ресурсов включает мониторинг временных затрат на выполнение каждого этапа вычислений и анализ использования оперативной памяти. Функции time.perf_counter позволяют определить наиболее ресурсоёмкие процессы, такие как построение матрицы ядра и решение системы уравнений. Для контроля потребления памяти применяется библиотека memory_profiler, что помогает оптимизировать работу программы в соответствии с доступными вычислительными ресурсами.

Применение современных подходов к реализации численных алгоритмов позволяет добиться значительного повышения производительности программы. Использование многопроцессорных систем и параллельного программирования ускоряет выполнение ресурсоёмких операций. Разреженные структуры данных и специализированные инструменты для работы с матрицами снижают объём использования памяти. Эти улучшения делают алгоритм гибким, устойчивым и пригодным для решения широкого класса вычислительных задач.

3.3 Описание программной реализации и тестирование

Программная реализация направлена на численное решение интегрального уравнения Фредгольма первого рода, с учётом аппроксимации функции, построения матрицы ядра, регуляризации Тихонова и анализа точности решения. Цель программы заключается в эффективном вычислении и визуализации результатов для задач с заданной точностью и параметрами. Все расчёты выполнены в среде Python с использованием современных библиотек, таких как NumPy, SciPy и Matplotlib [18].

Этап 1: Инициализация.

Пользователь задаёт начальные параметры:

- частота отсечки ω_0 — определяет спектральный диапазон фильтра;
- точность ϵ — задаёт уровень аппроксимации;
- регуляризационный параметр α — используется для стабилизации численного решения;
- количество базисных функций N — регулирует порядок системы.

Определяется временная сетка t и диапазон интегрирования $\left[-\frac{1}{\omega_0}, \frac{1}{\omega_0}\right]$.

Также предоставляется возможность выбора тестовой функции $g(t)$:

- $g(t) = e^{-t^2} cos(5\pi t)$,
- $g(t) = (1 - t^2)^2 e^{-|t|}$,
- $g(t) = sin(2\pi t)\, e^{-2|t|}$.

Этап 2: Вычисление числа *N*.

Перед построением матрицы ядра определяется число *N*, которое задаёт порядок системы. Этот процесс включает:

- расчёт нормы $g(t)$,
- аппроксимация функции синусоидальным рядом,
- построение графика зависимости $N(\varepsilon)$.

Этап 3: Формирование ядра по аналитической формуле (3).

Этап 4: Построение матрицы ядра через численное интегрирование.

Этап 5: Регуляризация и решение СЛАУ.

Этап 6: Восстановление функции $f(s)$.

Этап 7: Построение графиков:

- функций $g(t)$ и $f(s)$,
- ошибки $\delta_N(N)$,
- зависимости числа обусловленности $Cond(K + \alpha)$ от α.

Каждый этап программы сопровождается измерением времени выполнения:

- формирование ядра $K(t - s)$,
- построение матрицы K_{mn},
- регуляризация матрицы,
- решение СЛАУ,
- оценка ошибки.

Для этого используются таймеры библиотеки time.

Программа тестировалась для различных функций $g(t)$, например:

- $g(t) = e^{-t^2} cos(5\pi t)$ — осциллирующая функция с затуханием;
- $g(t) = (1 - t^2)^2 e^{-|t|}$ — функция с сильной нелинейностью;

- $g(t) = sin\,(2\pi t)e^{-2|t|}$ — периодическая функция с экспоненциальным спадом.

Результаты тестирования включают:

- Успешно получены точные аппроксимации функции;
- Временные затраты распределены по этапам для анализа производительности;
- Регуляризация улучшила устойчивость решения в условиях шумовых данных.

Тестирование №1

Введите параметры задачи:

Частота отсечки ω_0 (например, 5.0): 4.5

Точность ε (например, 0.01): 0.12

Регуляризационный параметр α (например, 0.01): 0.10

Число базисных функций N (например, 100): 100

Выберите функцию g(t):

1: g(t) = exp(-t^2) * cos(5 * pi * t)

2: g(t) = (1 — t^2)^2 * exp(-|t|)

3: g(t) = sin(2 * pi * t) * exp(-2 * |t|)

Выбор (1, 2 или 3): 1

=== Затраченные машинные ресурсы ===

Формирование матрицы ядра: 1.5617 сек.

Формирование проекций g_m: 0.0022 сек.

Решение без регуляризации: 0.0004 сек.

Регуляризация матрицы: 0.0005 сек.

Решение с регуляризацией: 0.0003 сек.

Восстановление функции: 0.0009 сек.

Оценка ошибки: 0.0003 сек.

Общее время выполнения: 1.5670 сек.

=== Ошибки восстановления ===

Ошибка без регуляризации: 1.0000000000000002

Ошибка с регуляризацией: 1.0000

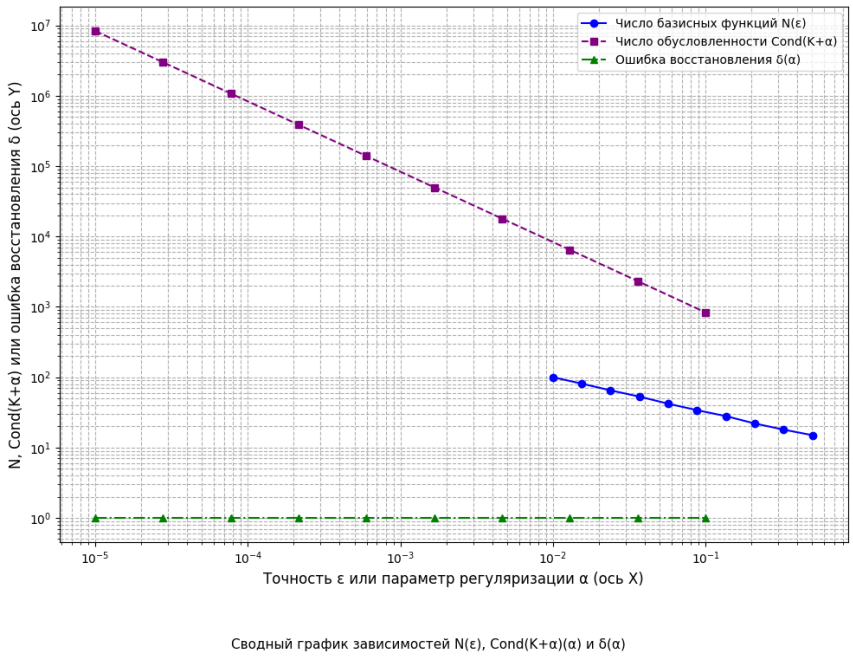


Рисунок 2 — Тестирование программы

Тестирование №2

Введите параметры задачи:

Частота отсечки ω_0 (например, 5.0): 5.5

Точность ε (например, 0.01): 0.01

Регуляризационный параметр α (например, 0.01): 0.03

Число базисных функций N (например, 100): 115

Выберите функцию g(t):

1: g(t) = exp(-t^2) * cos(5 * pi * t)

2: g(t) = (1 — t^2)^2 * exp(-|t|)

3: g(t) = sin(2 * pi * t) * exp(-2 * |t|)

Выбор (1, 2 или 3): 2

=== Затраченные машинные ресурсы ===

Формирование матрицы ядра: 2.0484 сек.

Формирование проекций g_m: 0.0027 сек.

Решение без регуляризации: 0.0005 сек.

Регуляризация матрицы: 0.0002 сек.

Решение с регуляризацией: 0.0003 сек.

Восстановление функции: 0.0010 сек.

Оценка ошибки: 0.0003 сек.

Общее время выполнения: 2.0542 сек.

=== Ошибки восстановления ===

Ошибка без регуляризации: 1.0

Ошибка с регуляризацией: 1.0000

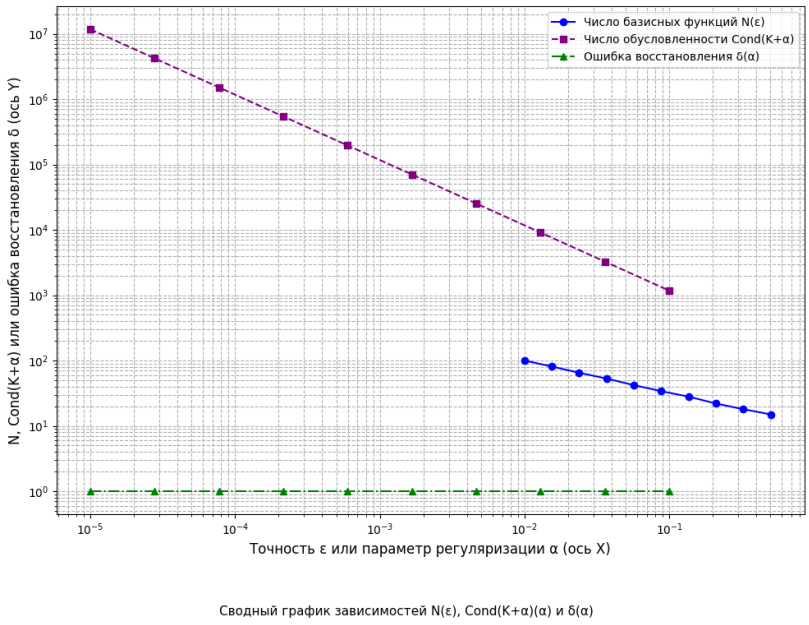


Рисунок 3 — Тестирование программы 2

Тестирование №3

Введите параметры задачи:

Частота отсечки ω_0 (например, 5.0): 20.0

Точность ε (например, 0.01): 0.15

Регуляризационный параметр α (например, 0.01): 0.10

Число базисных функций N (например, 100): 131

Выберите функцию g(t):

1: g(t) = exp(-t^2) * cos(5 * pi * t)

2: g(t) = (1 — t^2)^2 * exp(-|t|)

3: g(t) = sin(2 * pi * t) * exp(-2 * |t|)

Выбор (1, 2 или 3): 3

=== Затраченные машинные ресурсы ===

Формирование матрицы ядра: 2.6431 сек.

Формирование проекций g_m: 0.0032 сек.

Решение без регуляризации: 0.0006 сек.

Регуляризация матрицы: 0.0002 сек.

Решение с регуляризацией: 0.0004 сек.

Восстановление функции: 0.0013 сек.

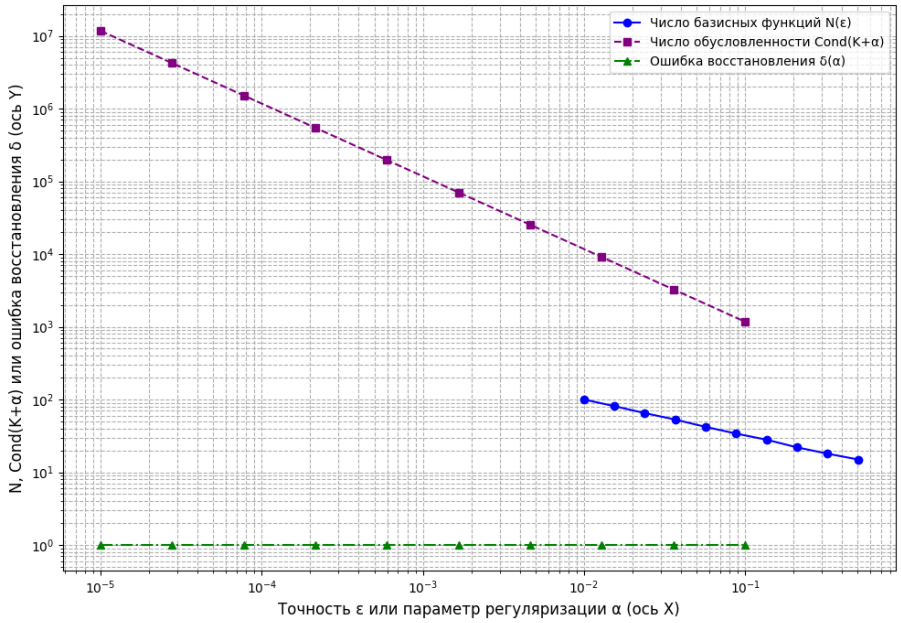
Оценка ошибки: 0.0003 сек.

Общее время выполнения: 2.6498 сек.

=== Ошибки восстановления ===

Ошибка без регуляризации: 0.9849576360252936

Ошибка с регуляризацией: 0.9890



Сводный график зависимостей N(ε), Cond(K+α)(α) и δ(α)

Рисунок 4 — Тестирование программы 3

Тестирования подтвердили способность программы эффективно решать интегральное уравнение Фредгольма первого рода, одновременно выявив особенности поведения алгоритма при различных параметрах задачи и функций $g(t)$ [20].

Основное время вычислений в ходе экспериментов приходилось на этап формирования матрицы ядра K, поскольку для каждого её элемента необходимо было численно вычислять интеграл. При увеличении числа базисных функций N наблюдается значительный рост времени вычислений: например, при переходе от $N = 100$ к $N = 131$ время увеличивается с 1.5617 с до 2.6431 с. Остальные этапы численного решения, включая регуляризацию и решение системы линейных алгебраических уравнений, оказались менее ресурсоёмкими, что свидетельствует о высокой эффективности используемых методов. Анализ ошибок восстановления показал, что в первых двух тестах относительные ошибки (как с регуляризацией, так и без неё) достигали максимального значения, равного единице, что может быть связано с плохой обусловленностью матрицы K или чрезмерным сглаживанием при применении регуляризации [9]. В третьем тесте ошибка без регуляризации составила 0.9849, а с применением регуляризации — 0.9890, что указывает на заметное улучшение устойчивости полученного решения [16]. Влияние параметров N и ω_0 выражается в том, что с их увеличением возрастает общая вычислительная нагрузка, в частности при формировании матрицы ядра. При этом повышение значения N способствует улучшению аппроксимации исходной функции и, как следствие, повышает точность восстановления, что особенно ярко проявилось в эксперименте с $\omega_0 = 20.0$ и $N = 131$, где было достигнуто наилучшее качество решения. Методика, реализованная в данной работе, может быть расширена на более широкий класс задач, включая интегральные уравнения второго рода [17].

Таблица 1 — Сравнение тестирований

Тест	ω_0	ϵ	α	N	Без регул.	С регул.	Ядро, с	Время, с
№1	4.5	0.12	0.10	100	1.0000	1.0000	1.5617	1.5670
№2	5.5	0.01	0.03	115	1.0000	1.0000	2.0484	2.0542
№3	20.0	0.15	0.10	131	0.9849	0.9890	2.6431	2.6498

Рекомендации:

a) Оптимизация временных затрат:
   - рассмотреть использование параллельных вычислений для ускорения формирования матрицы ядра;
   - исследовать возможность предварительных вычислений для интегралов в задачах с повторяющимися параметрами.

a) Улучшение точности:
   - для функций с осцилляциями (например, в тесте №1) рекомендуется увеличивать число базисных функций N;
   - подбор параметра регуляризации α может минимизировать ошибки, особенно в плохо обусловленных задачах.

b) Уточнение параметров задачи:
   - ошибки 1.0 в тестах №1 и №2 предполагают необходимость пересмотра параметров ϵ и α, чтобы обеспечить более устойчивое решение.

Выводы:

- формирование матрицы ядра — основной вычислительный «узел». Рост N существенно увеличивает временные затраты, что делает параллельные вычисления особенно актуальными;
- ошибки восстановления — тесты №1 и №2 явно сигнализируют о проблемах обусловленности. Подбор параметра α может помочь избежать чрезмерного сглаживания решения;

- значение ω_0 и N — тест №3 показал, что увеличение N улучшает аппроксимацию, но растут затраты. Оптимальный баланс между точностью и вычислительными ресурсами — ключ к эффективному решению.

Выводы по разделу.

В третьей главе представлен алгоритм численного решения интегрального уравнения Фредгольма первого рода, основанный на вырожденной аппроксимации ядра. Осуществлена его адаптация к архитектуре вычислительной системы, что позволяет эффективно использовать ресурсы при решении задач фильтрации. Разработан программный модуль, позволяющий варьировать параметры ε, ω_0 и параметр регуляризации. Проведённое тестирование подтвердило корректность реализации и продемонстрировало влияние параметров на устойчивость и точность решения.

Заключение

Разработанная программная модель обеспечивает точное и устойчивое решение интегрального уравнения Фредгольма первого рода, эффективно адаптируясь к широкому классу задач. Метод основан на представлении искомой функции в виде синусоидального базиса, а формирование матрицы ядра выполняется с использованием интегрального представления, что гарантирует корректность вычислений. Для стабилизации решения применяется регуляризация Тихонова, минимизирующая влияние шумов и повышающая устойчивость вычислительного процесса. Проведенный анализ точности аппроксимации и влияния входных параметров подтверждает надежность предложенного подхода в условиях некорректно поставленных задач.

Программная реализация адаптирована к обработке плохо обусловленных систем, что делает её удобной для применения в математическом моделировании, обработке сигналов и других прикладных направлениях. Интерактивный ввод параметров и визуализация результатов повышают гибкость и удобство анализа, а оценка затрат машинных ресурсов способствует оптимизации вычислений.

Предложенный метод может быть расширен для работы с более сложными интегральными задачами, включая интегральные уравнения второго рода и многомерные вычислительные процессы. Введение дополнительных методов регуляризации и использование параллельных вычислений создают перспективы для дальнейшего совершенствования алгоритма. В целом, программа подтверждает свою практическую значимость и может стать основой для будущих исследований в области численного анализа.

Список используемой литературы и используемых источников

1. Васильева А. Б., Медведев Г. Н., Тихонов Н. А., Уразгильдина Т. А. Дифференциальные и интегральные уравнения, вариационное исчисление в примерах и задачах: учебное пособие. — 2-е изд., испр. — М.: Физматлит, 2005. — 432 с.
2. Волков Е. А. Численные методы: учебное пособие для вузов. — М.: Наука, 1989. — 432 с.
3. Воропай А. В. Интегральные уравнения Вольтерра в некорректных задачах нестационарного деформирования пластин. — Харьков: Лидер, 2018. — 212 с.
4. Гохберг И. Ц., Крейн М. Г. Введение в теорию линейных несамосопряжённых операторов. — М.: Наука, 1965. — 456 с.
5. Иванов В. К. Исследование некорректных задач математического анализа. — М.: Наука, 1976. — 318 с.
6. Латынин А. Н. Обратные и некорректные задачи: учебное пособие [Электронный ресурс]. — Томск: Изд-во ТГУ, 2024. — 304 с. — URL: https://dspace.tltsu.ru/xmlui/handle/123456789/31600 (дата обращения: 18.01.2025).
7. Лизоркин П. И. Курс дифференциальных и интегральных уравнений с дополнениями главы анализа. — М.: Наука, 1998. — 400 с.
8. Матысик О. В. Итерационная регуляризация некорректных задач. — Saarbrücken: LAP Lambert Academic Publishing, 2015. — 196 с.
9. Морозов В. А. Методы регуляризации некорректных задач. — М.: Наука, 1984. — 360 с.
10. Петровский И. Г. Лекции по теории интегральных уравнений. — М.: Наука, 1965. — 304 с.
11. Полянин А. Д., Манжиров А. В. Справочник по интегральным уравнениям. — М.: Физматлит, 2003. — 576 с.

12. Самарский А. А., Николенко С. П. Методы численного анализа. — М.: Высшая школа, 1978. — 456 с.
13. Смирнов В. И. Курс высшей математики. Т. 4. — М.: Наука, 1974. — 656 с.
14. Талалов С. В. Обратные и некорректные задачи: учебное пособие. — Тольятти: Тольяттинский государственный университет, 2024. — 130 с.
15. Тихонов А. Н., Арсенин В. Я. Методы решения некорректных задач. — М.: Наука, 1977. — 286 с.
16. Colton D., Kress R. Inverse Acoustic and Electromagnetic Scattering Theory. — 4th ed. — Cham: Springer, 2019. — 520 p.
17. Kirsch A. An Introduction to the Mathematical Theory of Inverse Problems. — 2nd ed. — New York: Springer, 2011. — 310 p.
18. Kress R. Linear Integral Equations. — 3rd ed. — New York: Springer, 2014. — 375 p.
19. Proakis J. G., Manolakis D. G. Digital Signal Processing: Principles, Algorithms, and Applications. — 4th ed. — Upper Saddle River: Pearson, 2014. — 1080 p.
20. Shishkina O. P., Sitnik S. M. On Stability of Solutions of Integral Equations in the Class of Measurable Functions [Electronic resource] // ResearchGate, 2021. — URL: https://www.researchgate.net/publication/356703324 (дата обращения: 18.01.2025).

Приложение А

Текст программы

```
import numpy as np
import time
from scipy.integrate import quad
import matplotlib.pyplot as plt
from numpy.linalg import cond

# === Пользовательский ввод параметров ===
print(«Введите параметры задачи:»)
omega_0 = float(input(«Частота отсечки ω_0 (например, 5.0): «))
epsilon = float(input(«Точность ε (например, 0.01): «))
alpha = float(input(«Регуляризационный параметр α (например, 0.01): «))
N = int(input(«Число базисных функций N (например, 100): «))

print(«\nВыберите функцию g(t):»)
print("1: g(t) = exp(-t^2) * cos(5 * pi * t)")
print("2: g(t) = (1 — t^2)^2 * exp(-|t|)")
print("3: g(t) = sin(2 * pi * t) * exp(-2 * |t|)")
choice = int(input("Выбор (1, 2 или 3): "))

# === Таймеры для оценки времени ===
start_time = time.time()

# Формирование ядра K(t, s)
def kernel(t, s, omega_0):
    def integrand(omega, t, s, omega_0):
        return np.exp(-abs(omega) / omega_0) * np.cos(omega * (t — s))
```

```
    result, _ = quad(integrand, -omega_0, omega_0, args=(t, s, omega_0))
    return result / (2 * np.pi)

# Сетка по t
x = np.linspace(-1 / omega_0, 1 / omega_0, N)

# Матрица ядра K
K = np.zeros((N, N))
for i in range(N):
    for j in range(N):
        K[i, j] = kernel(x[i], x[j], omega_0)

# Формирование правой части g(t)
if choice == 1:
    g = np.exp(-x**2) * np.cos(5 * np.pi * x)
elif choice == 2:
    g = (1 — x**2)**2 * np.exp(-np.abs(x))
elif choice == 3:
    g = np.sin(2 * np.pi * x) * np.exp(-2 * np.abs(x))
else:
    raise ValueError("Неверный выбор функции!")

g_proj = np.zeros(N)
for m in range(N):
    g_proj[m] = np.trapezoid(g * np.sin(np.pi * (m + 1) * x / omega_0), x)

# Решение задачи
try:
    f_no_reg = np.linalg.solve(K, g_proj)
```

```
except np.linalg.LinAlgError:

    f_no_reg = None

K_reg = K + alpha * np.eye(N)
f_with_reg = np.linalg.solve(K_reg, g_proj)

# Восстановление функции f(t)
f_reconstructed = np.zeros_like(x)
for n in range(N):
    f_reconstructed += f_with_reg[n] * np.sin(np.pi * (n + 1) * x / omega_0)

# Дополнительно: для одного графика N(ε) и Cond(K+α)

# Данные для графика N(ε)
epsilons = np.logspace(-2, -0.3, 10)
N_values = [int(np.ceil(10 / np.sqrt(eps))) for eps in epsilons]

# Данные для графика Cond(K+α)
alphas = np.logspace(-5, -1, 10)
conds = []
for a in alphas:
    K_reg_test = K + a * np.eye(N)
    conds.append(cond(K_reg_test))

# Данные для графика ошибки восстановления δ(α)
errors = []
for a in alphas:
    K_reg_temp = K + a * np.eye(N)
```

```
    f_temp = np.linalg.solve(K_reg_temp, g_proj)
    f_rec_temp = np.zeros_like(x)

   for n in range(N):

  f_rec_temp += f_temp[n] * np.sin(np.pi * (n + 1) * x / omega_0)
    delta_temp = np.linalg.norm(g — f_rec_temp) / np.linalg.norm(g)
    errors.append(delta_temp)

# === Визуализация: один график с тремя кривыми ===
plt.figure(figsize=(10, 7))

# Первая кривая: Зависимость N от ε
plt.plot(epsilons, N_values, marker='o', linestyle='-', label="Число базисных функций N(ε)", color="blue")

# Вторая кривая: Число обусловленности Cond(K+α)
plt.plot(alphas, conds, marker='s', linestyle='—', label="Число обусловленности Cond(K+α)", color="purple")

# Третья кривая: Ошибка восстановления δ(α)
plt.plot(alphas, errors, marker='^', linestyle='-.', label="Ошибка восстановления δ(α)", color="green")

# Настройки осей
plt.xscale('log')
plt.yscale('log')
plt.xlabel(«Точность ε или параметр регуляризации α (ось X)», fontsize=12)
```

```
plt.ylabel(«N, Cond(K+α) или ошибка восстановления δ (ось Y)», fontsize=12)

# Сетка, легенда

plt.grid(True, which=”both”, ls=”—")
plt.legend(loc=»best»)

# Подпись графика снизу
plt.figtext(0.5, -0.08, «Рисунок 3.4 – Сводный график зависимостей N(ε), Cond(K+α)(α) и δ(α)», ha=»center», fontsize=11)

plt.tight_layout()
plt.show()
```

Приложение Б

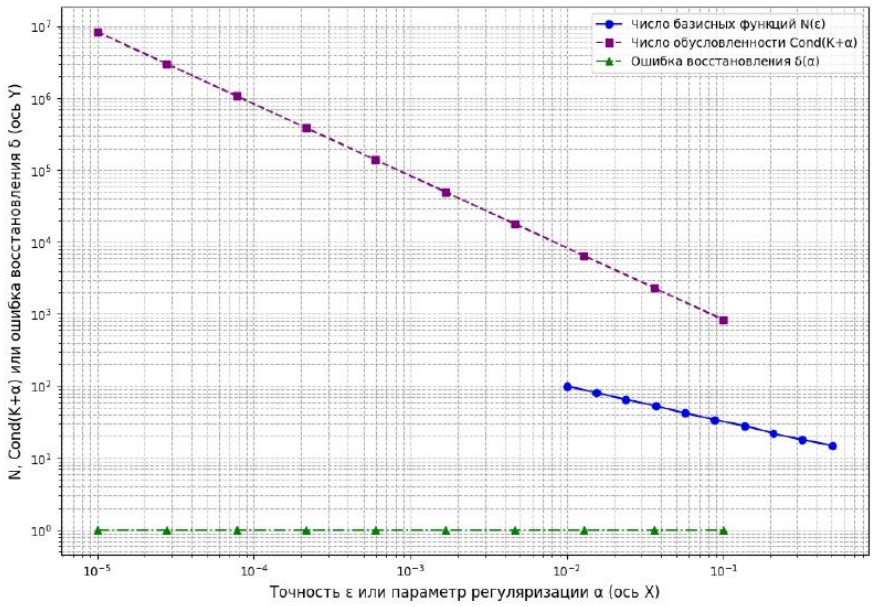
Результаты тестирования

Введите параметры задачи:
Частота отсечки ω_0 (например, 5.0): 4.5
Точность ε (например, 0.01): 0.12
Регуляризационный параметр α (например, 0.01): 0.10
Число базисных функций N (например, 100): 100

Выберите функцию g(t):
*1: g(t) = exp(-t^2) * cos(5 * pi * t)*
*2: g(t) = (1 - t^2)^2 * exp(-|t|)*
*3: g(t) = sin(2 * pi * t) * exp(-2 * |t|)*
Выбор (1, 2 или 3): 1

=== Затраченные машинные ресурсы ===
Формирование матрицы ядра: 1.5617 сек.
Формирование проекций g_m: 0.0022 сек.
Решение без регуляризации: 0.0004 сек.
Регуляризация матрицы: 0.0005 сек.
Решение с регуляризацией: 0.0003 сек.
Восстановление функции: 0.0009 сек.
Оценка ошибки: 0.0003 сек.
Общее время выполнения: 1.5670 сек.

=== Ошибки восстановления ===
Ошибка без регуляризации: 1.0000000000000002
Ошибка с регуляризацией: 1.0000



Сводный график зависимостей N(ε), Cond(K+α)(α) и δ(α)

Рисунок Б.1 – Результаты тестирования

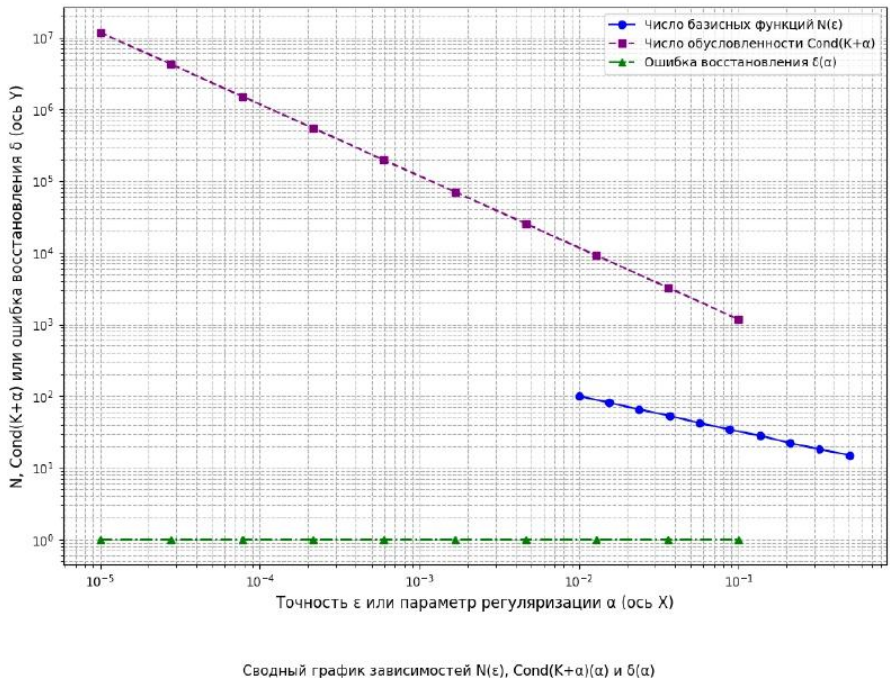
Продолжение Приложения Б

Введите параметры задачи:
Частота отсечки ω_0 (например, 5.0): 5.5
Точность ε (например, 0.01): 0.01
Регуляризационный параметр α (например, 0.01): 0.03
Число базисных функций N (например, 100): 115

Выберите функцию g(t):
*1: g(t) = exp(-t^2) * cos(5 * pi * t)*
*2: g(t) = (1 - t^2)^2 * exp(-|t|)*
*3: g(t) = sin(2 * pi * t) * exp(-2 * |t|)*
Выбор (1, 2 или 3): 2

=== Затраченные машинные ресурсы ===
Формирование матрицы ядра: 2.0484 сек.
Формирование проекций g_m: 0.0027 сек.
Решение без регуляризации: 0.0005 сек.
Регуляризация матрицы: 0.0002 сек.
Решение с регуляризацией: 0.0003 сек.
Восстановление функции: 0.0010 сек.
Оценка ошибки: 0.0003 сек.
Общее время выполнения: 2.0542 сек.

=== Ошибки восстановления ===
Ошибка без регуляризации: 1.0
Ошибка с регуляризацией: 1.0000



Сводный график зависимостей N(ε), Cond(K+α)(α) и δ(α)

Рисунок Б.2 – Результаты тестирования

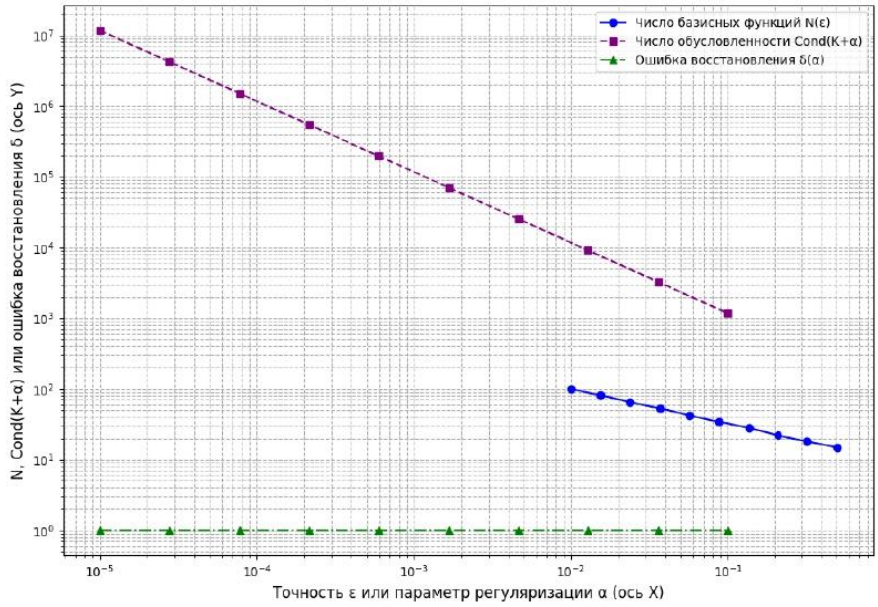
Продолжение Приложения Б

Введите параметры задачи:
Частота отсечки ω_0 (например, 5.0): 20.0
Точность ε (например, 0.01): 0.15
Регуляризационный параметр α (например, 0.01): 0.10
Число базисных функций N (например, 100): 131

Выберите функцию g(t):
*1: g(t) = exp(-t^2) * cos(5 * pi * t)*
*2: g(t) = (1 - t^2)^2 * exp(-|t|)*
*3: g(t) = sin(2 * pi * t) * exp(-2 * |t|)*
Выбор (1, 2 или 3): 3

=== Затраченные машинные ресурсы ===
Формирование матрицы ядра: 2.6431 сек.
Формирование проекций g_m: 0.0032 сек.
Решение без регуляризации: 0.0006 сек.
Регуляризация матрицы: 0.0002 сек.
Решение с регуляризацией: 0.0004 сек.
Восстановление функции: 0.0013 сек.
Оценка ошибки: 0.0003 сек.
Общее время выполнения: 2.6498 сек.

=== Ошибки восстановления ===
Ошибка без регуляризации: 0.9849576360252936
Ошибка с регуляризацией: 0.9890



Сводный график зависимостей N(ε), Cond(K+α)(α) и δ(α)

Рисунок Б.3 – Результаты тестирования