МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**
Кафедра дискретной математики и информационных технологий

# РАЗРАБОТКА МОДУЛЯ ВЫЧИСЛЕНИЯ ГРАДИЕНТОВ И ЛАПЛАСИАНОВ ПРИЛОЖЕНИЯ «GEOSTATSGU»

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 421 группы
направления 09.03.01 — Информатика и вычислительная техника
факультета КНиИТ
Кладиева Павла Денисовича

Научный руководитель
старший преподаватель _____ М. В. Белоконь

Заведующий кафедрой
к. ф.-м. н., доцент _____ Л. Б. Тяпаев

Саратов 2025

# ВВЕДЕНИЕ

Гидрометеорология, как наука, занимается исследованием процессов и явлений, связанных с атмосферными и гидрологическими процессами. Современные технологии позволяют собирать огромные объемы гидрометеорологических данных, что открывает новые возможности для их анализа и получения ценной информации. Однако обработка и анализ таких больших объемов данных могут быть сложной задачей, требующей применения численных методов.

На протяжении всей истории развития цивилизации климат оказывал и продолжает оказывать значительное влияние на жизнь и деятельность человека. Наука о климате, известная как климатология, возникла на основе практических потребностей человека и является одной из древнейших наук. В современную эпоху научно-технического прогресса зависимость человека от капризов погоды и климата не уменьшилась, хотя у человечества теперь значительно больше возможностей для преодоления последствий климатических бедствий. Тем не менее, последствия различных климатических катаклизмов становятся все более ощутимыми для экономики и трагичными для населения.

В связи с этим актуальными становятся проблемы, связанные с диагностикой современных климатических изменений, оценкой вклада естественных и антропогенных факторов в наблюдаемую климатическую изменчивость, а также разработкой сценариев будущего состояния климата.

Климатические данные весьма разнообразны из-за большого набора погодных (метеорологических) характеристик и специфики временных интервалов обобщений. Климатические данные могут быть сформированы из значений метеорологических параметров в отдельные сроки наблюдений, а также из значений различных масштабов усреднения: средних значений за ночь и день, средних суточных, пентадных, декадных, месячных, сезонных, годовых значений или экстремальных значений за отдельные периоды. Климатические ряды могут формироваться из самих значений метеорологических параметров, а также из их амплитуд и аномалий, числа дней с определенными значениями и т.п.

Для решения многих научных и прикладных задач возникает необходимость анализа климатических характеристик на больших территориях. Для

этого применяется картирование климатических данных.

Картирование климатических характеристик позволяет получать поля их распределений по площади за определенные временные промежутки. Сравнение климатических карт за отдельные временные интервалы позволяет диагностировать происходящие климатические изменения на отдельных территориях и выявлять на каждой конкретной территории особенности проявления климатических изменений. Таким образом решается важнейшая задача современной климатологии – дальнейшее более глубокое и более полное исследование изменений и изменчивости регионального и глобального климата, а также выявление закономерностей таких процессов.

Целью бакалаврской работы является разработка приложения для статистического анализа гидрометеорологических процессов и полей. Разработанное приложение впоследствии будет использоваться студентами магистратуры на географическом факультете.

Основным языком программирования для реализации приложения выбран C#, а для создания графического интерфейса используется библиотека Avalonia UI.

Для достижения поставленной цели в работе решаются следующие задачи:

— изучение метода конечных разностей;
— изучение принципа работы с библиотекой Avalonia UI;
— реализация модуля преобразования сеточных данных для решения конкретных задач;
— реализация модуля для вычисления градиентов.
— реализация модуля для вычисления лапласианов.

Структура работы включает пять глав:

1. Постановка задачи и описание инструментария.
2. Изменение технологического стека разработки.
3. Архитектура приложения.
4. Используемые элементы графического интерфейса.
5. Программная реализация приложения.

## КРАТКОЕ СОДЕРЖАНИЕ РАБОТЫ

В первом разделе выпускной квалификационной работы изложена постановка задачи и описание используемого математического инструментария. Он служит теоретической основой для последующей реализации приложения и включает в себя два подраздела: общие сведения о градиентах и лапласианах, а также методы их численного вычисления.

В первом подразделе представлены математические определения двух дифференциальных операторов, используемых в анализе пространственных данных:

— Градиент – векторная величина, характеризующая направление и скорость наибольшего возрастания скалярной функции. В n-мерном пространстве градиент скалярной функции представляет собой вектор, компоненты которого равны частным производным этой функции по соответствующим координатам.

— Лапласиан – оператор второго порядка, определяемый как дивергенция градиента. В случае декартовых координат он выражается как сумма вторых частных производных по каждой из координат.

Во втором подразделе подробно описан метод конечных разностей, применяемый для приближённого численного вычисления производных по дискретной сетке, что используется в климатологии и гидрометеорологии, где поля значений (например, давления, температуры, геопотенциала) представлены в виде таблиц значений, привязанных к определённым географическим координатам.

Приведены стандартные схемы для оценки первых и вторых производных по значениям в окрестностях заданной точки на сетке. Для расчётов используется прямоугольная равномерная сетка, шаг которой может варьироваться в зависимости от масштаба.

Во втором разделе бакалаврской работы рассматривается изменение технологического стека разработки приложения. Изначально приложение было реализовано на языке Python с использованием библиотеки PySide для создания графического интерфейса. Такой выбор был обусловлен популярностью Python в научной среде, наличием большого количества библиотек для анализа данных и относительной простотой разработки. Для работы с

табличными данными использовалась библиотека Pandas, предоставляющая удобные функции для работы с данными и структуру DataFrame, которая хорошо подходит для обработки двумерных массивов чисел.

Однако в процессе практического применения такой реализации выявились существенные недостатки. Одной из основных проблем стала сложность упаковки приложения в исполняемый файл. Используемый инструмент PyInstaller не обеспечивал стабильной сборки: при запуске .exe-файла открывалось фоновое консольное окно даже при применении специального флага, что отрицательно сказывалось на пользовательском опыте. Кроме того, запуск приложения занимал длительное время (5–8 секунд), что также являлось большим недостатком.

Учитывая эти проблемы, было принято решение отказаться от технологического стека Python и PySide в пользу более производительного и удобного стека на основе языка C# и библиотеки Avalonia UI.

Avalonia UI – это современный кроссплатформенный фреймворк, поддерживающий архитектуру MVVM и XAML-разметку.

Переход к новому стеку сопровождался следующими преимуществами:

— Высокая производительность, так как, в отличие от Python, C# является компилируемым языком программирования.
— Отсутствие консольного окна, так как приложения на C# можно собрать в виде полноценного GUI-приложения.
— Доступность плагина для визуального редактирования интерфейсов, использующих язык разметки XAML, для интегрированных сред разработки (IDE) Microsoft Visual Studio и JetBrains Rider.
— Простота упаковки и распространения, так как платформа .NET предоставляет встроенные средства для сборки кроссплатформенных исполняемых файлов.

В третьем разделе описывается архитектура разработанного приложения и описанию подхода, использованного при его проектировании.

В рамках реализации приложения была выбрана архитектурная модель MVVM (Model–View–ViewModel) — один из наиболее популярных и удобных шаблонов проектирования. Шаблон MVVM представляет собой развитие идеи разделения обязанностей, заложенной в более старом шаблоне MVC (Model–View–Controller). Шаблона MVVM обеспечивает чёткое разделение

логики приложения, пользовательского интерфейса и данных. Это позволяет повысить читаемость, расширяемость и тестируемость кода.

Архитектура приложения, реализующая шаблон MVVM, строится на трёх слоях:

1. Model (Модель) – содержит бизнес-логику и работает напрямую с данными. В данной реализации это слой, отвечающий за загрузку и обработку данных, реализацию численных методов (градиенты и лапласианы), а также сохранение результатов.
2. View (Представление) – графический интерфейс пользователя. Описан в формате XAML с использованием Avalonia UI. View отображает данные и предоставляет элементы управления (кнопки, флаги, поля ввода), но не содержит логики обработки.
3. ViewModel (Модель представления) — промежуточный слой, который связывает View и Model. Он реализует команды, свойства и обработчики событий, позволяя View взаимодействовать с логикой приложения.

Файловая структура проекта так же организована в соответствии с шаблоном MVVM:

— Models – содержит класс `ProcessData`, реализующий всю логику работы с файлами, а также логику расчётов градиентов и лапласианов.
— ViewModels – включает класс `MainWindowViewModel`, где находятся свойства для привязки, команды, реализация журнала событий и обработка пользовательских действий.
— Views – сожержит XAML-разметку и код главного окна.
— Services – отвечает за работу с диалогами открытия и сохранения файлов.
— Converters – содержит класс-конвертер для преобразования значений перечислений в логические, нужные для переключателей в интерфейсе.
— ViewLocator – обеспечивает связывание слоёв View с соответствующими слоями ViewModel.

Также в разделе представлена диаграмма вариантов использования. Участником системы является пользователь, который взаимодействует с приложением через графический интерфейс. Диаграмма отражает следующие сценарии взаимодействия пользователя с приложением:

— Импорт данных из CSV-файла.
— Выбор шага выборки (вручную или «по градусам»).
— Выбор режима вычислений (градиенты или лапласианы).
— Запуск расчёта.
— Просмотр результатов в Excel.
— Сохранение результатов.

В четвёртом разделе работы описываются элементы графического интерфейса пользователя (GUI), используемые в приложении, которые реализуются с использованием кроссплатформенной библиотеки Avalonia UI.

Графический интерфейс обеспечивает удобное взаимодействие между пользователем и вычислительной логикой приложения. Используемая в приложении библиотека Avalonia UI предоставляет широкий выбор визуальных компонентов для построения кроссплатформенных интерфейсов.

В приложении применены следующие элементы управления:
— Button – кнопка для запуска операций (открытие файла, запуск расчёта, сохранение и просмотр результатов).
— RadioButton – переключатели, позволяющие пользователю выбрать режим вычислений (градиенты или лапласианы).
— CheckBox – флаг «Расчёт по градусам», при активации которого шаг выборки по широте и долготе автоматически устанавливается равным четырём.
— TextBox – поля ввода, в которых пользователь может задать шаг выборки вручную.
— TextBlock – элемент для отображения статических или изменяемых текстовых сообщений. Используется для журнала событий.
— ScrollViewer – контейнер с полосами прокрутки, оборачивающий журнал событий и позволяющий просматривать длинные тексты.
— Grid – основной контейнер для построения макета окна, позволяющий разместить элементы в таблице с заданной геометрией.

В пятом разделе описывается программная реализация приложения. В нём подробно рассмотрены все этапы обработки данных, архитектура вычислительных модулей и механизм взаимодействия между компонентами системы.

Приложение ориентировано на студентов географического факультета, работающих с метеорологическими данными. Источник данных – файлы CSV, получаемые из файлов формата NetCDF через приложение Panoply. Объём данных – 360 строк и 1440 столбцов. Данные имеют шаг 0.25 градусов по широте и долготе.

Данные считываются в массив с помощью метода `ReadCsvFile`. Затем происходит проверка на наличие некорректных значний (NaN-значений), которые могут возникать в исходных файлах. При обнаружении таких значений они заменяются на `double.NaN`, а затем корректируются с помощью метода `FixNaNs`, применяющего интерполяцию по соседним ячейкам (по строкам или столбцам).

Также реализована функция дискретизации данных с помощью метода `SelectEveryNthElement`. Он позволяет пользователю задать шаг выборки по широте и долготе, тем самым уменьшив объём обрабатываемой информации и ускорив вычисления.

Вычислительная логика сосредоточена в методах `Calculate` и `Compute`. Метод `Calculate` организует проход по всей сетке данных, исключая граничные строки, и вызывает `Compute` для каждой ячейки. Метод `Compute`, в свою очередь, реализует два режима работы: вычисление градиентов и вычисление лапласианов.

После завершения вычислений результат округляется до шести знаков после запятой.

В методе `SaveToXlsx` реализована возможность сохранения результата в формате Excel (.xlsx). Для этого используется .NET-библиотека ClosedXML, которая обеспечивает удобный API для работы с Excel-документами.

Результат автоматически экспортируется в создаваемый приложением временный файл (для предварительного просмотра в Excel). Затем пользователь может экспортировать результат в выбранную им директорию с нужным именем файла.

Интерфейс реализован в XAML-файле `MainWindow.axaml` и соответствующем классе в файле `MainWindow.axaml.cs`. Основной контейнер интерфейса — Grid, разбитый на логические секции:

— Слева – панель управления параметрами (переключатели режимов, поля ввода, кнопки).

— Справа – журнал событий, в котором отображаются сообщения о ходе выполнения операций.

Логика взаимодействия пользователя с интерфейсом реализована в классе `MainWindowViewModel`, который содержит следующие ключевые компоненты:

— Свойства: шаг выборки, путь к файлу, статус расчёта, коллекция сообщений журнала событий.
— Команды: команды открытия и сохранения файла, команда для запуска расчёта, команда для открытия временного файла. Каждая команда связана с кнопкой интерфейса и запускает соответствующую операцию.
— Журнал событий: реализован как `ObservableCollection<string>`, автоматически обновляющая интерфейс при добавлении новых сообщений.

# ЗАКЛЮЧЕНИЕ

В рамках данной бакалаврской работы было создано приложение для численного анализа процессов и полей, характерных для гидрометеорологии. Разработанное приложение позволяет удобно выполнять расчёты градиентов и лапласианов.

Актуальность разработки обусловлена потребностью в доступных и практичных средствах анализа гидрометеорологических данных. Приложение ориентировано в первую очередь на студентов географического факультета, облегчая им работу с пространственными данными. Вместе с тем, при дальнейшем развитии и расширении функциональности оно может быть полезно также специалистам и исследователям, работающим в области гидрометеорологии и смежных дисциплин.

В работе представлен обзор метода конечных разностей, а также подробно описана структура и возможности разработанного программного обеспечения. Полученные результаты могут найти применение в практической деятельности, связанной с анализом и интерпретацией гидрометеорологической информации.

Для достижения цели были решены следующие задачи:

— изучен метод конечных разностей;
— освоен принцип работы с библиотекой Avalonia UI;
— реализован модуль преобразования сеточных данных под конкретные задачи анализа;
— разработаны модули для расчёта градиентов и лапласианов.

В результате было разработано приложение, которое выполняет свою задачу по облегчению анализа процессов и полей в гидрометеорологии.

**Основные источники информации**:

1 Борисенков Е. П. Климат и деятельность человека - Москва : Академия наук СССР, 1982 г. – 133 с.
2 Лобанов В. А. Учебное пособие по региональной климатологии. Учебное издание - Санкт-Петербург : РГГМУ, 2020 г. – 170 с.
3 Зверев А. С. Синоптическая метеорология - Ленинград : Гидрометеоиздат., 1977 г. – 711 с.
4 Exploring Avalonia UI: A Modern Cross-Platform Framework for .NET

Developers: [Электронный ресурс] URL: https://www.linkedin.com/pulse/exploring-avalonia-ui-modern-cross-platform-framework-sharma-wg7zf (дата обращения: 29.04.2025) Загл. с экрана. Яз. Англ.

5 Model-View-ViewModel (MVVM): [Электронный ресурс] URL: https://learn.microsoft.com/ru-ru/dotnet/architecture/maui/mvvm (дата обращения: 05.05.2025) Загл. с экрана. Яз. Англ.