**«Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В.И.Ульянова (Ленина)»**

**(СПбГЭТУ «ЛЭТИ»)**

|  |  |  |
| --- | --- | --- |
| **Направление** | 09.03.01 – Информатика и вычислительная техника | |
| **Профиль** | Системы автоматизированного проектирования | |
| **Факультет** | Компьютерных технологий и информатики | |
| **Кафедра** | систем автоматизированного проектирования | |
| *К защите допустить* |  | |
| Зав. кафедрой, к.т.н., доцент |  | Бутусов Д.Н. |

выпускная квалификационная работа бакалавра

**Тема: Высокопроизводительные программные средства анализа динамических систем на основе технологии CUDA**

|  |  |  |
| --- | --- | --- |
| Студент |  | Ширнин К.В. |
| *подпись* |
| Руководитель к.т.н., доцент |  | Бутусов Д.Н. |
| *подпись* |
| Консультант |  | Рыбин В.Г. |
| *подпись* |
| Консультант к.э.н., доцент |  | Голигузова Г.В. |
| *подпись* |
| Консультант по нормоконтролю |  | Родионова Е.А. |
| *подпись* |

Санкт-Петербург

2023

задание

на выпускную квалификационную работу

|  |  |
| --- | --- |
|  | Утверждаю |
|  | Зав. кафедрой САПР, к.т.н., доцент |
|  | \_\_\_\_\_\_\_\_\_\_\_\_ Бутусов Д.Н. |
|  | «\_4\_»\_апреля\_\_\_\_\_\_\_2023 г. |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Студент | Ширнин Кирилл Витальевич | | | | Группа | 9302 |
| Тема работы: Высокопроизводительные программные средства анализа динамических систем на основе технологии CUDA | | | | | | |
| Место выполнения ВКР: кафедра САПР, СПбГЭТУ «ЛЭТИ» | | | | | | |
| Исходные данные (технические требования):  Результатом ВКР служит информационный продукт (программная библиотека), для высокопроизводительного распределенного анализа нелинейных динамических систем. Разработка позволяет ускорить время анализа на 2-3 порядка по сравнению с существующими косвенными аналогами. Для использования разработанного программного обеспечения требуется компьютер, который соответствует следующим минимальным требованиям:   1. Операционная система (OS): Windows Vista SP3, Windows 7 или Windows HPC Server 2008 2. Центральный процессор (CPU): Intel Pentium Dual-core CPU 3. Оперативная память (RAM): 2 GB 4. Видеокарта (GPU): GeForce GTX 745 в линейке GeForce или Quadro K620 в линейке Quadro или Tesla K20 в линейке Tesla 5. Поддержка Nvidia CUDA Toolkit 11 | | | | | | |
| Содержание ВКР: аннотация, введение, основные методы анализа и численные методы, программное обеспечение моделирования и анализа динамических систем, экономическое обоснование ВКР, заключение, список использованных источников (литературы). | | | | | | |
| Перечень отчетных материалов: пояснительная записка, иллюстративный материал, материалы, материалы к презентации ВКР, иные отчетные материалы. | | | | | | |
| Дополнительный раздел – экономическое обоснование ВКР | | | | | | |
| Дата выдачи задания | | | Дата представления к защите | | | |
| «\_04\_»\_апреля\_\_\_\_\_\_\_2023г. | | | «\_3\_»\_\_июня\_\_\_\_\_\_\_\_2023г. | | | |
| Студент | |  | | Ширнин К.В. | | |
| *подпись* | |
| Руководитель к.т.н., доцент | |  | | Бутусов Д.Н. | | |
| *подпись* | |
| Консультант | |  | | Рыбин В.Г. | | |
| *подпись* | |
| Консультант к.э.н., доцент | |  | | Голигузова Г.В. | | |
| *подпись* | |

календарный план выполнения

выпускной квалификационной работы

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | Утверждаю | | | | | |
|  | | | Зав. кафедрой САПР, к.т.н., доцент | | | | | |
|  | | | \_\_\_\_\_\_\_\_\_\_\_\_Бутусов Д.Н. | | | | | |
|  | | | «\_04\_»\_апреля\_\_\_2023\_ г. | | | | | |
| Студент | | Ширнин Кирилл Витальевич | | | | Группа | | 9302 |
| Тема работы: Высокопроизводительные программные средства анализа динамических систем на основе технологии CUDA | | | | | | | | |
| № п/п | Наименование работ | | | | | | Срок выполнения | |
| 1 | Обзор литературы по теме работы | | | | | | 04.04 -09.04 | |
| 2 | Разработка технического задания | | | | | | 10.04 -16.04 | |
| 3 | Анализ технического задания | | | | | | 17.04 -23.04 | |
| 4 | Проектирование | | | | | | 24.04 -30.04 | |
| 5 | Реализация | | | | | | 01.04 -20.04 | |
| 6 | Тестирование | | | | | | 21.04-26.04 | |
| 7 | Выполнение дополнительного раздела «Экономическое обоснование ВКР» | | | | | | 27.04-30.04 | |
| 8 | Оформление пояснительной записки | | | | | | 01.05-14.05 | |
| 9 | Оформление иллюстративного материала | | | | | | 15.05-18.05 | |
| 10 | Прохождение предварительной зашиты на кафедре | | | | | | 19.05.2023 | |
| 11 | Представление ВКР для проверки степени оригинальности пояснительной записки | | | | | | 22.05.2023 | |
| 12 | Представление ВКР к защите. | | | | | | 03.06.2023 | |
|  |  | | | | | |  | |
| Студент | | | |  | Ширнин К.В. | | | |
| *подпись* |
| Руководитель к.т.н., доцент | | | |  | Бутусов Д.Н. | | | |
| *подпись* |
| Консультант доцент | | | |  | Рыбин В.Г. | | | |
| *подпись* |
| Консультант к.э.н., доцент | | | |  | Голигузова Г.В. | | | |
| *подпись* |

аннотация

Проект посвящен разработке высокопроизводительных методов и средств анализа нелинейных систем.

В процессе выполнения работы были рассмотрены различные подходы к реализации инструмента для распределенных вычислений. Была разработана встраиваемая программная библиотека для распределенных вычислений и анализа динамических нелинейных систем.

При помощи инструмента удалось производить многомерный анализ нелинейных систем, а также повышать «разрешение» результирующих графиков. При сравнении инструмента с аналогами выяснилось, что скорость вычислений увеличилась в 100-1000 раз.

SUMMARY

The project is dedicated to the development of high-performance methods and tools for the analysis of nonlinear systems.

In the course of the work, various approaches to the implementation of a tool for distributed computing were considered. It was developed in the developed software library for computing and analyzing the density of nonlinear systems.

With the help of the tool, it was possible to perform a multidimensional analysis of nonlinear systems, as well as to increase the "resolution" of the resulting graphs. When using the tool with analogues of calculations, the calculation speed increased by 100-1000 times.

Оглавление

[Перечень сокращений и условных обозначений 7](#_Toc134966440)

[1. Введение 8](#_Toc134966441)

[1.1. Задачи моделирования и анализа динамических систем. Задачи, решаемые долгосрочным моделированием и диаграммами высокого разрешения 10](#_Toc134966442)

[1.2. Обоснование потребности в повышении производительности средств моделирования. Производительность расчетов на CPU. Актуальность работы 11](#_Toc134966443)

[1.3. Расчеты на графических ускорителях. Технология CUDA 11](#_Toc134966444)

[1.3.1. Понятие вычислительного параллелизма 12](#_Toc134966445)

[1.3.2. Языки и среды разработки 13](#_Toc134966446)

[1.3.3. Архитектура приложений, ориентированных на выполнение на GPU 14](#_Toc134966447)

[1.3.4. Особенности программирования на CUDA 16](#_Toc134966448)

[1.3.5. Уточненная постановка задачи исследования. Выводы по главе 17](#_Toc134966449)

[2. Основные методы анализа и численные методы 18](#_Toc134966450)

[2.1. Тестовые нелинейные задачи с типовыми особенностями, обнаруживаемыми методами анализа 19](#_Toc134966451)

[2.2. Численные методы как способ перехода от непрерывных систем к дискретным 23](#_Toc134966452)

[Список использованных источников 26](#_Toc134966453)

# Перечень сокращений и условных обозначений

LS (англ. – Lyapunov Spectrum) – спектр показателей Ляпунова

LLE (англ. – Largest Lyapunov Exponent) – наибольший показатель Ляпунова

CPU (англ. – Central Processing Unit) – центральный процессор

GPU (англ. – Graphics Processing Unit) – графический процессор (видеокарта)

CUDA (англ. – Compute Unified Device Architecture) – программно-аппаратная архитектура параллельных вычислений, которая позволяет существенно увеличить вычислительную производительность благодаря использованию графических процессоров фирмы Nvidia

API (англ. – Application Programming Interface) – описание способов взаимодействия одной компьютерной программы с другими

RAM (англ. – Random Access Memory) – оперативная память, один из видов компьютерной памяти

Хост (англ. – Host) – Центральный процессор, управляющий выполнением программы

Устройство (англ. – Device) – видеоадаптер, выступающий в роли сопроцессора центрального процессора

ПО – программное обеспечение

IDE (англ. – Integrated Development Environment) – комплекс программных средств, используемый программистами для разработки ПО

ОДУ – обыкновенное дифференциальное уравнение

QT – фреймворк для разработки кроссплатформенного программного обеспечения на языке программирования C++

CD – композитный диагональный метод численного интегрирования

# Введение

Каждый год различные группы ученых предлагают новые нелинейные системы, описывающие поведение природных объектов, электрических цепей, механических устройств и т.д. За последние несколько десятилетий был сформирован классический набор инструментов для анализа нелинейной системы, включая бифуркационный анализ [1], расчет спектра показателей Ляпунова (LS) [2], в частности, анализ наибольшей экспоненты Ляпунова (LLE) [3], рекуррентный анализ, поиск бассейнов притяжения и т.д. Однако детальный и многопараметрический анализ нелинейных систем все еще остается сложной задачей, поскольку большинство инструментов и программного обеспечения используют в основном вычислительные ресурсы центрального процессора [4]. Следует отметить, что многопараметрический анализ обычно является задачей, в которой вычисления систем с разными параметрами не зависят друг от друга, что открывает широкие возможности для сокращения общего времени выполнения анализа за счет применения параллельных вычислений. Распараллеливание обычно предполагает использование большого количества вычислительных единиц, т.е. многих ядер/процессоров, где особый интерес представляют графические процессоры (GPU).

Технологии параллельных вычислений, такие как Nvidia CUDA, в последнее время стали очень популярны в связи с возрастающей сложностью вычислительных задач. Изначально графические процессоры были разработаны для ускорения обработки изображений и графики в видеоиграх, чтобы их массивные вычисления можно было выполнять параллельно. В последние десятилетия GPU быстро проявили себя как недорогие и мощные многопоточные процессоры благодаря своей огромной вычислительной мощности. Например, компания Nvidia разработала интерфейс программирования приложений (API) для общих вычислений, намереваясь использовать GPU для выполнения программ общего назначения. CUDA — это API программирования, поддерживаемый устройствами Nvidia. Такие языки программирования, как C, C++ и Fortran, позволяют использовать его через библиотеки CUDA.

Однако разработка соответствующих алгоритмов для использования преимуществ параллельных вычислений остается важной проблемой, требующей решения. Следует отметить, что в случае сложных существующих программ эффективная реализация на GPU не всегда возможна или экономически выгодна. Многие вычислительные задачи являются последовательными, что вынуждает нас использовать последовательный подход. По этой причине команды в программе должны выполняться одна за другой.

Еще одной серьезной проблемой технологий GPU и CUDA может стать ограниченный объем памяти, доступный на видеокарте. Неэффективное использование графической памяти приводит к увеличению количества обменов данными между хостом (RAM) и вычислительным устройством CUDA (GPU memory) и значительно усложняет выполнение вычислений. Следствием представленных трудностей является снижение эффективности вычислений. Подводя итог, можно сказать, что разработка высокоэффективных алгоритмов для классических инструментов анализа с использованием распределенных вычислений является актуальной задачей.

Целью данной выпускной дипломной работы является повышение производительности труда проектировщика, за счет сокращения времени анализа нелинейных динамических систем. Для достижения цели необходимо решить следующие задачи:

1. Адаптировать стандартные алгоритмы нелинейного анализа под распределенные вычисления.
2. Оптимизировать работы с памятью видеокарты, с целью уменьшения числа перебросов между устройством и хостом.

## Задачи моделирования и анализа динамических систем. Задачи, решаемые долгосрочным моделированием и диаграммами высокого разрешения

Задачи моделирования и анализа динамических систем имеют особое значение как в теоретическом, так и в практическом плане. Они являются важной частью исследований в области физики, математики, инженерии, экономики и многих других наук [5]. Сегодня моделирование и анализ динамических систем широко используются в различных сферах человеческой деятельности для решения таких задач, как оптимизация производства [6], прогнозирование экономических [7] и социальных явлений, управление технологическими процессами и т.д.

Решаемые задачи долгосрочного моделирования обычно связаны с прогнозированием системных изменений в длительной временной перспективе. Это может быть, например, прогноз изменения климатических условий [8], роста населения [9], изменения демографических параметров [10], сценарии развития геополитической ситуации и т.д. Долгосрочное моделирование позволяет оценить возможные сценарии развития событий, разработать стратегии и принять меры для минимизации рисков и оптимизации процессов.

Диаграммы высокого разрешения – это инструменты для анализа динамических систем, которые позволяют визуализировать различные процессы и явления, происходящие в системе. На диаграммах высокого разрешения можно отобразить как статические характеристики системы, так и динамические процессы, происходящие в ней во времени. Это позволяет получить визуальное представление о состоянии и функционировании системы, а также обнаружить возможные проблемы и сделать выводы о необходимых изменениях.

Задачи моделирования и анализа динамических систем имеют большое значение во многих областях науки. Они также позволяют оценить возможности и риски, разработать стратегии и принять эффективные решения для управления системой.

## Обоснование потребности в повышении производительности средств моделирования. Производительность расчетов на CPU. Актуальность работы

Обоснование потребности в повышении производительности средств моделирования – это ключевой аспект в различных областях, от науки до инженерии. Однако, с ростом сложности задач, количество данных и требований к точности, производительность средств моделирования становится более критичной. В настоящее время, многие системы моделирования используют процессор (CPU) для вычисления и анализа данных, что ограничивает возможность обработки больших объемов данных и замедляет процесс расчетов.

Повышение производительности средств моделирования может улучшить эффективность исследований и прогнозирования, сократить время разработки и снизить затраты. Одним из способов решения этой проблемы может быть использование графических процессоров (GPU) для ускорения вычислений и снижения времени обработки данных.

Таким образом, повышение производительности средств моделирования является актуальной задачей, которая может улучшить эффективность и точность анализа и прогнозирования в разнообразных областях науки и техники.

## Расчеты на графических ускорителях. Технология CUDA

На сегодняшний день, много компаний, разработчиков и ученых занимаются разработкой ПО, адаптированным для распределенных вычислений [11]. Одной из самых важных областей использования графических ускорителей (GPU) является расчет. Расчеты, обязательные для процесса создания различных моделей и программ, могут занимать существенный объем времени, что может значительно повысить стоимость разработок, научных исследований и т.п. На помощь приходит технология CUDA, которая позволяет использовать GPU для выполнения расчетов. Она позволяет использовать двунаправленную связь между центральным процессором (CPU) и GPU для ускорения работы вычислительных задач.

Чтобы начать использовать CUDA при разработке программного обеспечения, необходимо выполнить несколько шагов.

В первую очередь необходимо выбрать GPU, который поддерживает технологию CUDA. Важными параметрами являются количество ядер и память. Кроме того, при выборе GPU нужно учитывать какие конкретно из них поддерживают технологию CUDA.

Далее, для использования процедур CUDA, необходимо установить соответствующий набор библиотек. Некоторые из них могут быть уже установлены при установке драйверов GPU для операционной системы.

После установки библиотек можно начать создание программы. Для начала необходимо создать основной файл и указать, что он будет использовать CUDA. Затем в основной файл необходимо добавить функцию, которая будет запускаться на GPU.

Также для использования процедур CUDA необходимо использовать специальную нотацию – язык CUDA C. Он позволяет использовать множество функций, которые обеспечивают управление памятью, массивами, переменными и т.д.

### Понятие вычислительного параллелизма

Вычислительный параллелизм [12] представляет собой способ использования нескольких вычислительных ресурсов (в том числе процессорных ядер) одновременно для решения одной и той же задачи.

В процессе выполнения задачи каждый ресурс (или параллельный элемент) работает над частью данных, тем самым увеличивая скорость выполнения всего процесса. Подобный подход может быть реализован в различных областях, включая научные исследования, графические приложения, базы данных и т.д.

Вычислительный параллелизм требует специальных методов и практик для эффективного использования ресурсов. Одна из ключевых задач состоит в том, чтобы правильно распределить данные между параллельными элементами и оптимизировать работу этих элементов.

Одним из наиболее распространенных примеров применения вычислительного параллелизма является использование графических процессоров (GPU) для выполнения сложных вычислительных задач, таких как научные исследования, обработка изображений и видео, моделирование и т.д. Благодаря своей высокой производительности и возможности обработки больших объемов данных, в настоящее время многие приложения с использованием вычислительного параллелизма разрабатываются с учетом возможности использования GPU.

### Языки и среды разработки

При написании программ, использующих технологию CUDA, возможно использовать несколько языков программирования, но наиболее подходящий язык для CUDA — это C++. C++ позволяет разработчикам создавать высокопроизводительные приложения, а также эффективно использовать многоядерные и графические процессоры.

Одной из основных причин, почему C++ является языком программирования, наиболее подходящим для CUDA, является возможность использования естественных низкоуровневых операций, таких как управление памятью и выполнение расчетов на видеокарте. Это позволяет программистам полностью контролировать исполнение и задействовать максимально возможные преимущества вычислительной мощности графических процессоров.

Другим преимуществом C++ является его широкое распространение и обширная документация. Нет необходимости тратить время на изучение нового языка программирования, что позволяет разработчикам быстрее начать разработку программы.

Одной из лучших сред разработки для программирования с использованием CUDA является Visual Studio от Microsoft. Visual Studio предоставляет широкие возможности для создания и отладки приложений, а также удобный интерфейс для работы с библиотеками CUDA. Он также позволяет оптимизировать и настраивать отдельные рабочие процессы и удобен для разработки и отладки больших проектов. Также Visual Studio поддерживает официальные отладочные плагины CUDA (NSight Monitor), что существенно облегчает жизнь при разработке программ, рассчитанных на распределенные вычисления.

В целом, выбор языка программирования и среды разработки для создания приложений, использующих CUDA, зависит от индивидуальных потребностей и предпочтений разработчика. Однако, использование C++ и Visual Studio может оказаться наиболее эффективным и быстрым способом разработки приложений с использованием CUDA. По этой причине, для реализации поставленных задач в этой работе был выбран именно C++ в сочетании с IDE Microsoft Visual Studio.

### Архитектура приложений, ориентированных на выполнение на GPU

Приложения, ориентированные на выполнение на GPU, обычно состоят из двух частей: хост и устройство. Хост-часть представляет собой код, который выполняется на центральном процессоре и используется для управления передачей данных и обеспечения взаимодействия между устройством и хостом. Устройство-часть представляет собой код, который выполняется непосредственно на GPU и используется для обработки данных.

Основной принцип программирования для GPU заключается в разделении задачи на параллельные части, которые могут быть выполнены на разных ядрах GPU. Чем больше вычислительных ядер в GPU, тем больше операций можно выполнить параллельно, что в свою очередь позволяет быстрее обрабатывать большие объемы данных.

Одной из особенностей реализованного инструмента является тонкий подход к архитектуре всей библиотеки. В отличие от косвенных аналогов, параллелизм в данной разработке происходит только для главных инструментов, в которых нуждаются большинство видов анализа (такие как “моделирование нелинейной системы”, “поиск пиков в смоделированной траектории”, “DBSCAN”, и т.д.). Это позволяет сделать библиотеку модульной, встраиваемой в другое ПО, так как файлы с функциями вызова анализа не будут содержать в себе элементов программного кода из CUDA библиотеки, что в свою очередь позволяет сторонним разработчикам не использовать компилятор CUDA в своих разрабатываемых ПО. Также, как показала практика, такой подход существенно облегчает наращивание библиотеки дополнительным функционалом и ее обслуживание в целом. Пример зависимостей файлов и библиотек для типичного Desktop C++/QT приложения представлен на рисунке ниже (Рисунок 1.1).

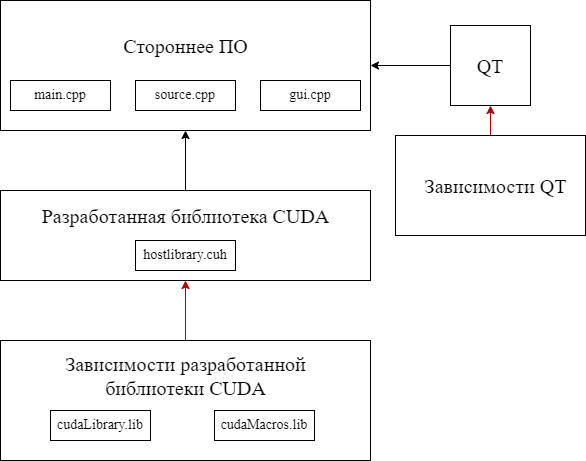


Рисунок 1.1

Разработчик, который разрабатывает “Стороннее ПО” (main.cpp, source.cpp и gui.cpp являются файлами проекта этого разработчика) может подключить разработанную в данном дипломе библиотеку (hostlibrary.cuh) точно так же, как и фреймворк пользовательского графического интерфейса QT. На рисунке красным цветом помечены файлы и зависимости разработанной библиотеки, которые не нужно включать в свое ПО стороннему разработчику. Никаких конфликтов в данном случае не произойдет, потому что архитектура реализованного инструмента устроена таким образом, что файл hostlibrary.cuh не содержит в себе CUDA кода.

### Особенности программирования на CUDA

В задаче, которая представлена в данной работе, есть ряд нюансов, которые необходимо учесть при разработке инструмента. Большинство приложений, которые рассчитаны на распределенные и многопоточные вычисления в CUDA не предполагают результат, который будет занимать объем памяти больше, чем стандартный объем памяти GPU или даже объем установленной оперативной памяти в компьютере. Поэтому для реализации инструмента был выбран подход, при которой удалось полностью уйти от ограничений, связанных с объемом занимаемой памяти результатом. Он заключается в том, чтобы разделять большие вычисления на части. Предварительно в таком подходе нужно учесть свободный объем памяти графического устройства, который сможет занять алгоритм.

Также необходимо учесть то, что библиотека CUDA основывается на языке C, который является процедурным языком программирования. Этот факт ставит жесткие рамки при разработке инструмента, потому что CUDA не поддерживает современные особенности языка C++, например, такие как STL, ООП, умные указатели, лямбда-функции и другие [13]. Из-за это приходится ограничиваться обычными указателями, одномерными массивами и процедурным стилем написания программного кода.

### Уточненная постановка задачи исследования. Выводы по главе

Подводя итоги, использование графических ускорителей для решения задач моделирования и анализа динамических систем является актуальным и перспективным направлением исследований, так как позволяет повысить производительность средств моделирования.

Основными задачами данной работы являются:

1. Разработка высокопроизводительных алгоритмов многомерного анализа динамических систем.
2. Разработка программных средств моделирования динамических систем.
3. Сравнение реализованного инструмента с существующими аналогами.

# Основные методы анализа и численные методы

Одно из ключевых понятий в науке и технике – анализ. Анализ позволяет изучить свойства объекта, движения, происходящие в нем изменения и прогнозировать будущие изменения, определять факторы, влияющие на эти изменения. Анализ в науке и технике осуществляется с помощью основных методов, которые можно разделить на две большие группы – аналитические и численные.

Аналитические методы – это методы исследования, использующие аналитические выражения для получения ответов. Такие методы используются на начальных этапах исследования, когда необходимо получить предварительный ответ. Аналитические методы чаще всего решаются в виде закрытой формулы, что значительно упрощает расчет и позволяет получить ответ быстро. Однако в некоторых случаях аналитическое решение может быть слишком сложным или невозможным, тогда используются численные методы.

Численные методы [14] – это методы решения математических задач в численном виде. Эти методы применяются, когда аналитические решения не применимы, или же аналитические решения имеют слишком высокую сложность. Однако численные методы часто более затратны и требуют большего количества времени, чем аналитические методы [15].

Среди численных методов наиболее распространены методы численного дифференцирования и интегрирования, методы конечных разностей и методы конечных элементов. Методы численного дифференцирования и интегрирования применяются для нахождения производных и интегралов в точках, а методы конечных разностей и элементов используются для нахождения решений дифференциальных уравнений и задач механики, деформируемых тел.

В результате, аналитические и численные методы являются важными инструментами для науки и техники. Анализ данных и явлений с помощью этих методов позволяет принимать решения на основе научных данных и принимать меры для повышения эффективности и оптимизации процессов в различных областях человеческой деятельности.

## Тестовые нелинейные задачи с типовыми особенностями, обнаруживаемыми методами анализа

В данной разделе рассматривается применение методов анализа для решения нелинейных задач с типовыми особенностями. Методы анализа являются одним из основных способов изучения динамического поведения систем, которые описываются нелинейными уравнениями. Например, методы анализа позволяют изучать устойчивость динамических систем, определять периодические и хаотические режимы.

В качестве тестовых нелинейных задач рассматриваются три системы:

1. Система ОДУ Рёсслера [16] (2.1);
2. Система ОДУ, описывающая поведение модифицированной электрической цепи ЧУА с мемристором;
3. Система ОДУ, описывающая поведение электрической цепи генератора Колпитца [17].

Система ОДУ Рёсслера была введена в 1976 году Рёслером в качестве модели хаотической динамики. Эта система имеет следующий вид:

, (2.1)

где , и – переменные состояния, , и – параметры системы.

Аттрактор системы Рёсслера представлен на рисунке ниже (Рисунок 2.1).

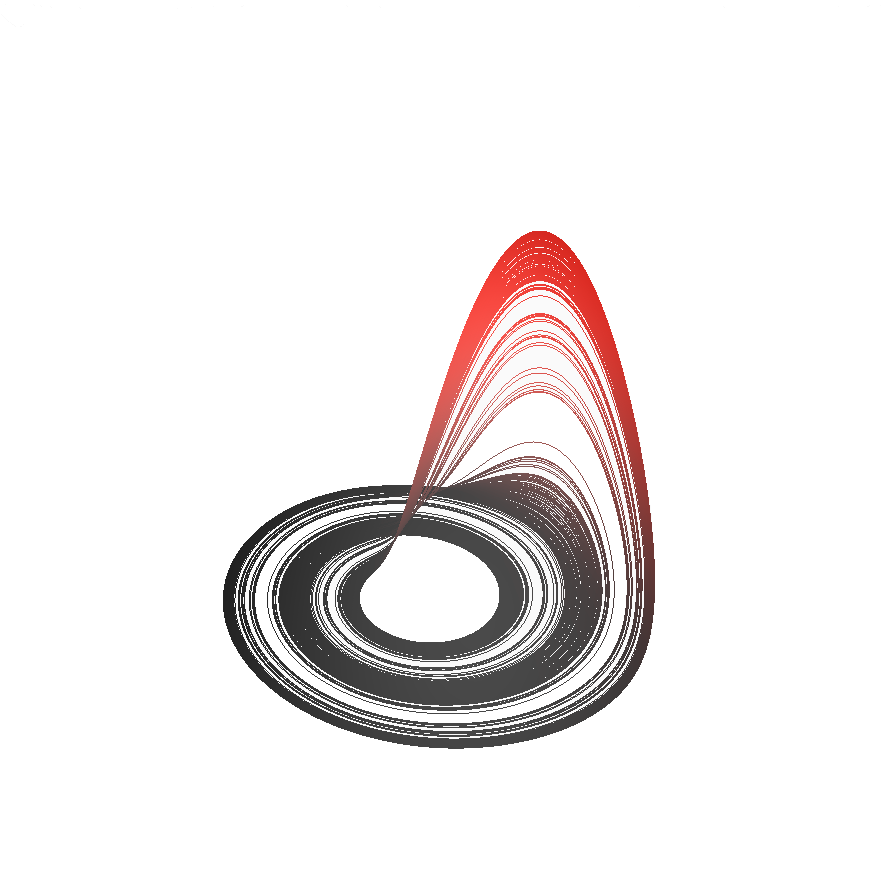


Рисунок 2.1

Одна из особенностей этой системы заключается в том, что при некоторых значениях параметров она демонстрирует хаотическое поведение. В частности, если значения параметров , и , то система Рёсслера проявляет хаотические колебания.

Еще одной особенностью этой системы является то, что она имеет две равновесные точки. Их можно найти, приравняв производные в системе уравнений Рёсслера к нулю. В результате оказывается, что существует две неподвижные точки (2.2, 2.3)

(2.2)

(2.3)

Вторая тестовая задача основана на система ОДУ, описывающая поведение модифицированной электрической цепи Чуа с мемристором, представленной на рисунке ниже (Рисунок 2.2).

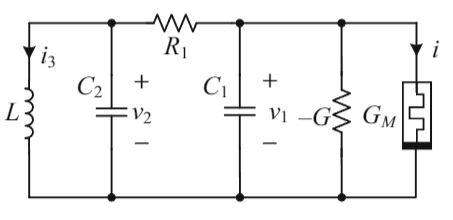


Рисунок 2.2

По первому и второму законам Кирхгофа составляется следующая система ОДУ (2.4):

(2.4)

Фазовый портрет данной системы для первых двух переменных состояния представлен на рисунке ниже (Рисунок 2.3).

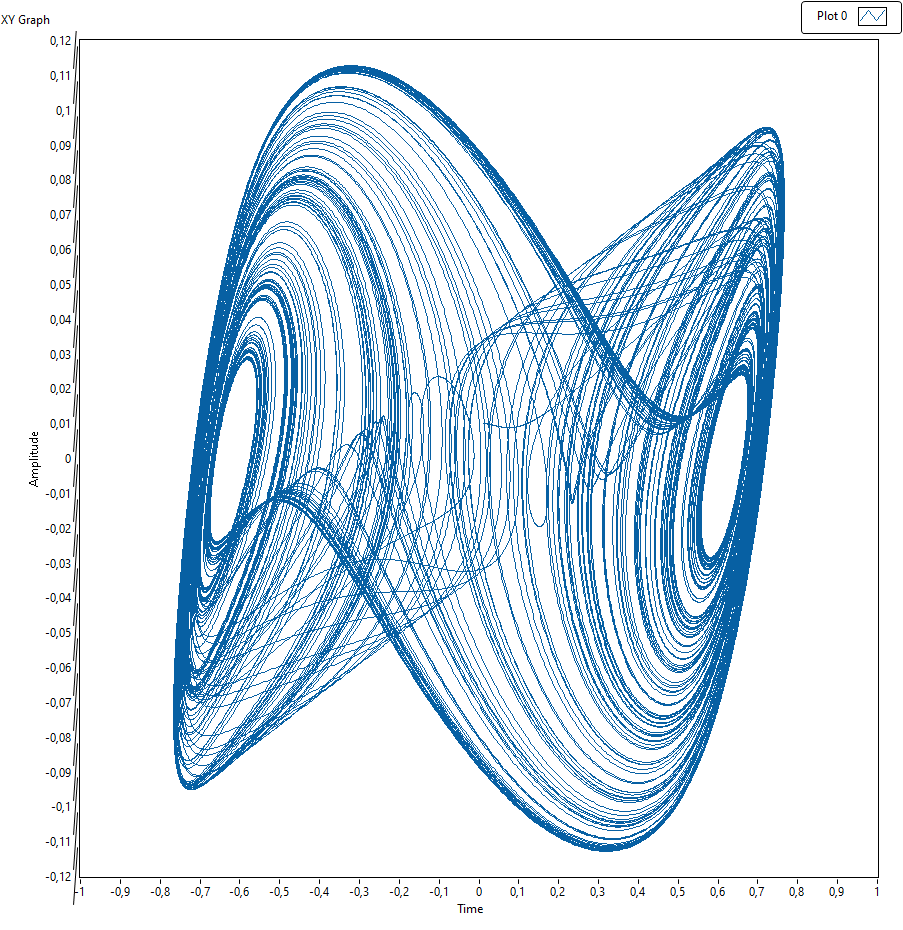


Рисунок 2.3

Третья тестовая задача основана на система ОДУ, описывающая поведение электрической цепи генератора Колпитца, представленной на рисунке ниже (Рисунок 2.4).

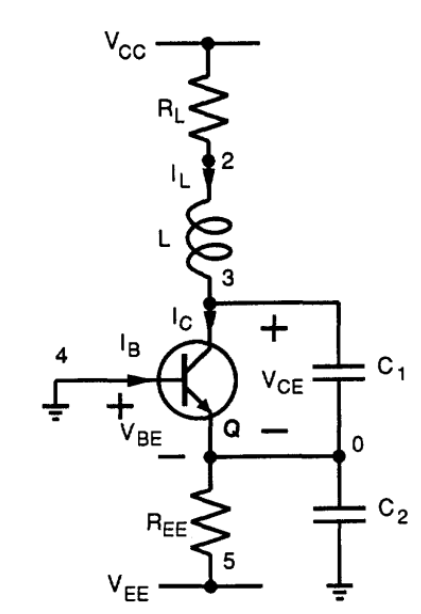


Рисунок 2.4

По первому и второму законам Кирхгофа составляется следующая система ОДУ (2.5-2.7):

(2.5)

(2.6)

(2.7)

Фазовый портрет данной системы для первых двух переменных состояния представлен на рисунке ниже (Рисунок 2.5).

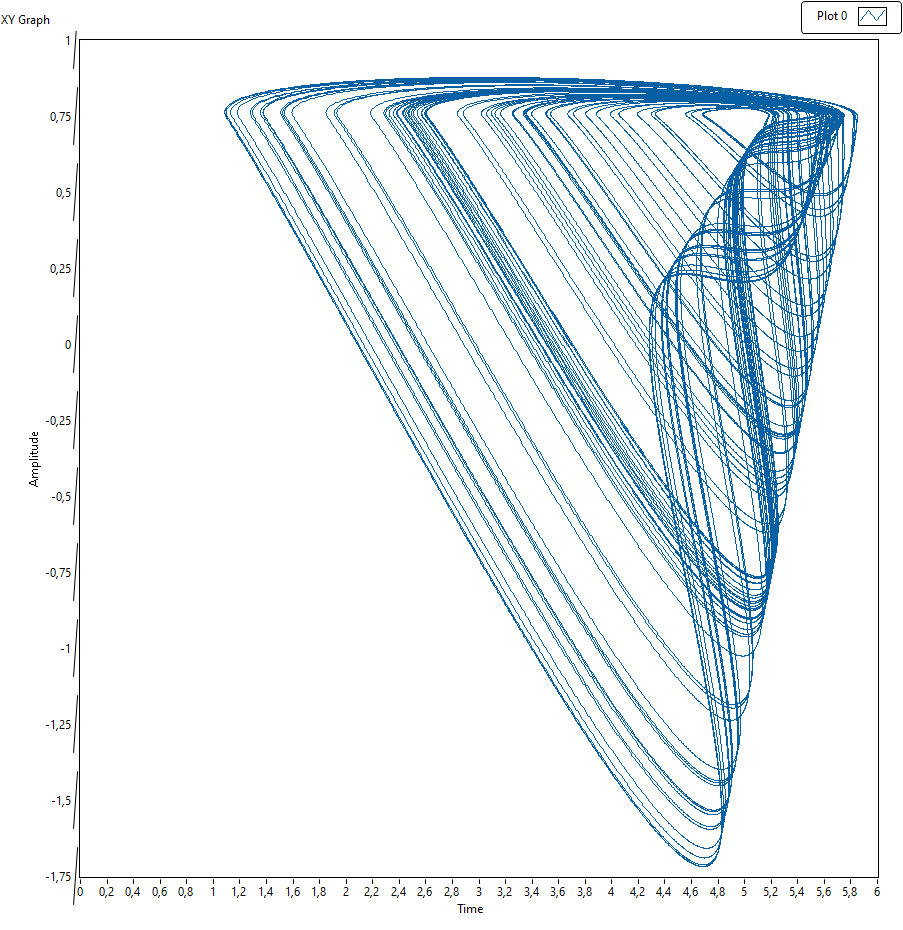


Рисунок 2.5

## Численные методы как способ перехода от непрерывных систем к дискретным

Одним из важных аспектов при решении нелинейных задач является переход от непрерывных систем к дискретным. Для этого применяются численные методы, которые позволяют аппроксимировать непрерывную систему дискретными шагами для более удобной работы с ней.

Один из наиболее распространенных методов численного решения дифференциальных уравнений - метод Эйлера-Кромера [18]. Данный метод позволяет получить дискретное решение дифференциального уравнения в виде конечной последовательности значений, которые могут быть легко и быстро вычислены.

Композитный диагональный метод - еще один метод численного решения дифференциальных уравнений, который используется для приближенного решения больших систем уравнений. Он является совокупностью методов Эйлера-Кромера и D-метода [19]. Он позволяет более точно решать дифференциальные уравнения, чем метод Эйлера-Кромера.

В данной работе мы будем применять композитный диагональный метод численного интегрирования, как более точный, по сравнению с методом Эйлера-Кромера.

Для системы Рёсслера метод CD расписывается следующим образом:

Изначально определяются шаги интегрирования для двух частей метода CD, посредством использования коэффициента симметрии (2.8, 2.9).

(2.8)

(2.9)

Затем расписывается первая часть метода CD, идентичная применению к системе метода Эйлера-Кромера (2.10-2.12).

(2.10)

(2.11)

(2.12)

После этого расписывается вторая часть метода CD, идентична применению D-метода (2.13-2.23). При существовании неявностей в уравнении их необходимо разрешить. В системе Рёсслера неявность встречается только во втором (2.18-2.22) и третьем (2.13-2.17) уравнениях.

(2.13)

(2.14)

(2.15)

(2.16)

(2.17)

(2.18)

(2.19)

(2.20)

(2.21)

(2.22)

(2.23)

В системе, описывающую модифицированную цепь Чуа с мемристором разрешение неявности аналитически невозможно, в отличие от системы Рёсслера – переменные состояния являются степенями и аргументами функций, таких как . Для разрешения таких неявностей воспользуемся методом простой итерации [27].

2.24

2.25

2.26

2.27

2.28

2.29

2.30

2.31

2.32

2.33

2.34

2.35

2.36

2.37

2.38

2.39

2.40

2.41

2.42

2.43

2.44

2.45

# Список использованных источников

Ueta T., Chen G. Bifurcation analysis of Chen's equation //International Journal of Bifurcation and Chaos. – 2000. – Т. 10. – №. 08. – С. 1917-1931.

Jin L., Lu Q. S., Twizell E. H. A method for calculating the spectrum of Lyapunov exponents by local maps in non-smooth impact-vibrating systems //Journal of sound and Vibration. – 2006. – Т. 298. – №. 4-5. – С. 1019-1033.

Rosenstein M. T., Collins J. J., De Luca C. J. A practical method for calculating largest Lyapunov exponents from small data sets //Physica D: Nonlinear Phenomena. – 1993. – Т. 65. – №. 1-2. – С. 117-134.

Sastry S. Nonlinear systems: analysis, stability, and control. – Springer Science & Business Media, 2013. – Т. 10.

Ganji D. D., Sabzehmeidani Y., Sedighiamiri A. Nonlinear systems in heat transfer. – Elsevier., 2018.

Silva T. L., Camponogara E. A computational analysis of multidimensional piecewise-linear models with applications to oil production optimization //European Journal of Operational Research. – 2014. – Т. 232. – №. 3. – С. 630-642.

Clements M. P., Franses P. H., Swanson N. R. Forecasting economic and financial time-series with non-linear models //International journal of forecasting. – 2004. – Т. 20. – №. 2. – С. 169-183.

Wang Z. X., Ye D. J. Forecasting Chinese carbon emissions from fossil energy consumption using non-linear grey multivariable models //Journal of Cleaner Production. – 2017. – Т. 142. – С. 600-612.

Livdahl T. P., Sugihara G. Non-linear interactions of populations and the importance of estimating per capita rates of change //The Journal of animal ecology. – 1984. – С. 573-580.

Pollak R. A. Two-sex demographic models //Journal of Political Economy. – 1990. – Т. 98. – №. 2. – С. 399-420.

Göddeke D. et al. Exploring weak scalability for FEM calculations on a GPU-enhanced cluster //Parallel Computing. – 2007. – Т. 33. – №. 10-11. – С. 685-699.

Parallel computing. — Текст : электронный // Wikipedia : [сайт]. — URL: https://en.wikipedia.org/wiki/Parallel\_computing (дата обращения: 13.05.2023).

CUDA Toolkit Documentation 12.1 Update 1. — Текст : электронный // Nvidia CUDA : [сайт]. — URL: https://docs.nvidia.com/cuda/ (дата обращения: 13.05.2023).

Numerical analysis. — Текст : электронный // Wikipedia : [сайт]. — URL: https://en.wikipedia.org/wiki/Numerical\_analysis (дата обращения: 13.05.2023).

Bober W. Introduction to numerical and analytical methods with MATLAB® for engineers and scientists. – CRC Press, 2013.

Rössler O. E. An equation for continuous chaos //Physics Letters A. – 1976. – Т. 57. – №. 5. – С. 397-398.

Kennedy M. P. Chaos in the Colpitts oscillator //IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications. – 1994. – Т. 41. – №. 11. – С. 771-774.

Saroja G., Nuriyah L. Numerical solution of nonlinear vibration by Euler-Cromer method //IOP Conference Series: Materials Science and Engineering. – IOP Publishing, 2019. – Т. 546. – №. 3. – С. 032029.

Butusov D. N., Ostrovskii V. Y., Tutueva A. V. Simulation of dynamical systems based on parallel numerical integration methods //2015 IEEE NW Russia Young Researchers in Electrical and Electronic Engineering Conference (EIConRusNW). – IEEE, 2015. – С. 56-59.

Iterative method. — Текст : электронный // Wikipedia : [сайт]. — URL: https://en.wikipedia.org/wiki/Iterative\_method (дата обращения: 14.05.2023).