МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное

образовательное учреждение высшего образования

**«Челябинский государственный университет»**

**(ФГБОУ ВО «ЧелГУ»)**

Математический факультет

Кафедра вычислительной математики

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

**БАКАЛАВРСКАЯ РАБОТА**

Динамическая визуализация решений дифференциальных уравнений на фазовых плоскостях и пространствах средствами OpenGL

|  |  |
| --- | --- |
| ДОПУСТИТЬ К ЗАЩИТЕ  Заведующий кафедрой,  ученая степень, ученое звание  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Ф.И.О  (подпись) | Выполнил студент Чмутов Максим Дмитриевич  Группы МП-402  очной формы обучения  направления подготовки (специальности)  «Прикладная математика и информатика»  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |

|  |  |
| --- | --- |
| Нормоконтроль  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Ф.И.О.  (подпись) | Научный руководитель  Лепчинский Михаил Германович  доцент кафедры вычислительной математики,  кандидат физико-математических наук  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |

Челябинск

2021

**Введение**

Фазовый портрет – это геометрическое представление траектории динамической системы в фазовом пространстве. Каждый набор начальных условий представляется в виде кривой или точки.

Фазовое пространство – пространство, каждая точка которого соответствует одному и только одному состоянию динамической системы.

Динамическая система – это математический объект, соответствующий реальным системам, эволюция которых однозначно определяется начальным состоянием.

Фазовые портреты используются в изучении динамических систем. По фазовым траекториям можно увидеть, присутствуют ли в системе стабильные и нестабильные точки или предельные циклы. Стабильная точка – состояние, к которому система стремится прийти. Нестабильная точка – состояние, от которого система стремится уйти. Предельный цикл – замкнутая фазовая траектория, к которой хотя бы одна траектория стремится при стремлении времени к бесконечности.

В данной работе рассматривается создание фазовых портретов динамических систем, заданных обыкновенными дифференциальными уравнениями, при помощи численного метода высокой точности решения дифференциальных уравнений.

1. **Создание фазовых портретов**
   1. **Алгоритм создания фазовых траекторий**

Пусть дано дифференциальное уравнение в явном виде:

Положим вектор – вектор значений производных:

Отсчет производится от нулевого индекса, т.е. и т.д. Далее численным методом Рунге-Кутта четвертого порядка решается задача Коши

для разных начальных условий, т.е. разных . Здесь – вектор функция,

Для каждого начального условия получаем массив пар вида , по которым рисуются фазовые траектории по точкам по возрастанию .

* 1. **Метод Рунге-Кутта четвертого порядка**

Рассмотрим задачу Коши для дифференциального уравнения первого порядка. (Далее ).

.

Тогда приближенное значение в последующих точках вычисляется по итерационной формуле:

здесь - величина шага сетки по , а также:

Этот метод имеет четвёртый порядок точности, т.е. суммарная ошибка на интервале интегрирования имеет порядок .

Пседокод алгоритма:

Функция МетодРунгеКутта (f, y0, t0, tn, h)

Вход: f = f(y,t) - вектор функция, где y – вектор,

t – число.

y0 – начальное значение искомой функции при переменной t0.

t0,tn – начало и конец интервала интегрирования, t0 < tn.

h – величина шага t.

Выход: Массив значений искомой функции, размерность элементов которого равна размерности вектора y в f(y,t).

количествоШагов = ОкруглитьВБольшую((tn-t0)/h)

tПредыдущий = t0

yПредыдущий = y0

Результат = массив векторов

Результат.ДобавитьВКонец(yПредыдущий)

Для i от 1 до количествоШагов

K1 = h\*f(tПредыдущий, yПредыдущий)

K2 = h\*f(tПредыдущий + h/2, yПредыдущий + K1/2)

K3 = h\*f(tПредыдущий + h/2, yПредыдущий + K2/2)

K4 = h\*f(tПредыдущий + h, yПредыдущий + K3)

yПредыдущий = yПредыдущий + (K1+2\*K2+2\*K3+K4)/6

tПредыдущий = tПредыдущий + h

Результат.ДобавитьВКонец(yПредыдущий)

Возврат Результат

1. **Структура графического приложения**

Для визуализации фазовых портретов было написано графическое приложение на языке C++ с использованием графической библиотеки OpenGL. Основной задачей написания приложения являлось обеспечение удобных средств манипуляции различными объектами и возможности легко добавлять различный функционал, не обязательно напрямую связанный с поставленной задачей (например, отрисовка системы координат).

* 1. **Класс «Движок»**

Класс «Движок» является основой всего графического приложения. Как его основные свойства можно выделить содержание в себе экземпляров классов «Сцена», «Регистр камер» и «Регистр поведений отрисовки» и наличие метода «Главный цикл», вызов которого запускает само приложение.

Метод «Главный цикл» при вызове создаёт окно приложения, задаёт все необходимые параметры для работы OpenGL и запускает цикл, в котором вызываются метод «Обновить все объекты» у экземпляра «Сцены» и метод «Отрисовать все камеры» у экземпляра «Регистра камер».

* 1. **Класс «Сцена»**

«Сцена» является контейнером для экземпляров класса «Объект сцены». Его основными методами являются «Создать объект» и «Обновить все объекты».

Метод «Создать объект» создает экземпляр «Объекта сцены» и возвращает на него указатель.

Метод «Обновить все объекты» обходит все объекты и, либо удаляет объект, если на нем стоит соответствующий флаг, либо вызывает у него метод «Обновить поведения».

* 1. **Класс «Объект сцены»**

«Объект сцены» является главной единицей графического приложения. В нём содержится строка с именем, экземпляр класса «Трансформация» и массив «Поведений объекта».

Метод «Обновить поведения» для каждого элемента массива «Поведений объекта» вызывает отдельными обходами методы «Проснуться» и «Начать», если ранее они не были вызваны, и методы «Обновление» и «Позднее Обновление».

* + 1. **Класс «Трансформация»**

Определяет позицию, ориентацию и масштабирование объекта, а также хранит в себе векторы направлений относительно ориентации этого объекта, то есть векторы «вперед», «вверх» и «вправо».

* 1. **Абстрактный класс «Поведение объекта»**

Определяет поведение объекта с течением времени. Внутри класса определены следующие абстрактные методы: «Проснуться» и «Начать», которые вызываются только один раз при создании экземпляра поведения, а также «Обновление» и «Позднее Обновление», вызываемые каждую итерацию «Главного цикла» приложения.

Такая структура необходима, если, допустим, одно поведение опирается на то, что другое уже обновилось. Тогда для первого необходимый функционал нужно поместить в «Позднее Обновление», а для второго – в «Обновление».

Также определен абстрактный метод «При удалении», который вызывается, при удалении поведения из массива «Поведений объекта».

* 1. **Класс «Поведение Камеры»**

Является имплементацией «Поведения объекта». Содержит в себе угол обзора (в градусах), и расстояния до ближней и дальней плоскости отрисовки объектов. Также определен метод «Получить видовую матрицу», возвращающий видовую матрицу для трансформации объекта, на котором весит это поведение.

Здесь переопределены методы «Проснуться» и «При удалении», в которых у «Регистра камер» указатель на данную камеру добавляется и удаляется соответственно.

* + 1. **«Регистр камер»**

Является контейнером указателей для всех камер на сцене. Метод «Отрисовать все камеры» обходит все камеры и вызывает у «Регистра поведений отрисовки» метод «Отрисовать все», который берет как параметр текущую камеру.

* 1. **Абстрактный класс «Поведение отрисовки»**

Наследник класса «Поведение объекта» имеющий определение абстрактного метода «Отрисовать», параметром которого является видовая матрица камеры.

Также были переопределены методы «Проснуться» и «При удалении», в которых у «Регистра поведений отрисовки» указатель на данное поведение добавляется и удаляется соответственно.

* + 1. **«Регистр поведений отрисовки»**

Является контейнером указателей для всех «Поведений отрисовки» на сцене. Метод «Отрисовать все» принимает «Поведение камеры» как параметр и обходит все указатели вызывая у них функцию «Отрисовать», передавая как параметр матрицу, полученную вызовом метода «Получить видовую матрицу» данного «Поведения камеры».

1. **Реализация создания и отрисовки фазовых портретов**
   1. **Классы «Дифференциальное уравнение» и «Символьное выражение»**

Класс «Дифференциальное уравнение» хранит в себе порядок старшей производной и экземпляр «Символьного выражения», описывающий функцию от младших производных и параметра .

Класс «Символьное выражение» содержит символьное выражение в виде строки и ассоциативный массив, сопоставляющий число строке, этот массив обозначает значения констант в выражении. Метод «Вычислить выражение» использует обратную польскую запись для вычисления значения символьного выражения. Параметром этого метода является ассоциативный массив, сопоставляющий строке функцию, берущую стек чисел и возвращающую число.

* 1. **Класс «Фазовый портрет»**

За создание и отрисовку фазовых портретов отвечает класс «Фазовый портрет», являющийся имплементацией класса «Поведение отрисовки».

Метод «Создать фазовые траектории» по заданным количеству траекторий и радиусу создаёт множество начальных точек. Форма этого множества зависит от степени дифференциального уравнения: линия, для первой степени, квадрат для второй и куб для третьей и более. Если степень уравнения больше трёх, то в графическом интерфейсе можно указать соответствие трёх любых степеней производных координатным осям, а для остальных – значение этих производных. Во время создания начального множества для каждой точки вычисляется и сохраняется в отдельный массив её фазовая траектория на заданной сетке параметра .

Переопределенный метод «Отрисовать» обходит массив фазовых траекторий и по заданным максимальной длине, минимальной длине ребра и текущему значению параметра рисует траектории в виде ломаных кривых, закрашивая их одним из трёх способов: один цвет, два цвета и радужный. Динамическая окраска траекторий в разные цвета повышает информативность построенного фазового портрета.

В переопределенном методе «Обновление» происходит приращение параметра на разницу времени между предыдущей и текущей итерацией «Главного цикла» с учетом заданной скорости визуализации.

1. **Графический интерфейс**

Для создания графического интерфейса используется библиотека «Dear ImGui», позволяющая быстро и просто создавать удобный графический интерфейс.

* 1. **Задание дифференциального уравнения**

На рис.1 представлен интерфейс задания дифференциального уравнения.

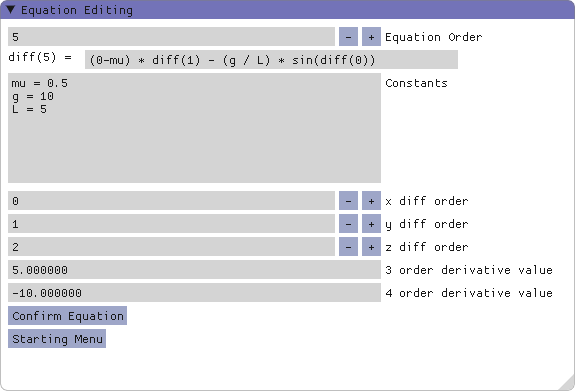


Рис.1. Интерфейс задания дифференциального уравнения.

В поля этого интерфейса входит:

1. Старший порядок производной;
2. Функция от младших производных (производная обозначается как «diff(n)», где n – порядок производной);
3. Значения констант в функции;
4. Соответствие порядков производных к координатным осям;
5. Значение производных, не уместившихся на координатные оси.

После нажатия на кнопку «Подтвердить уравнение» открывается окно проигрывателя визуализации.

* 1. **Проигрыватель визуализации**

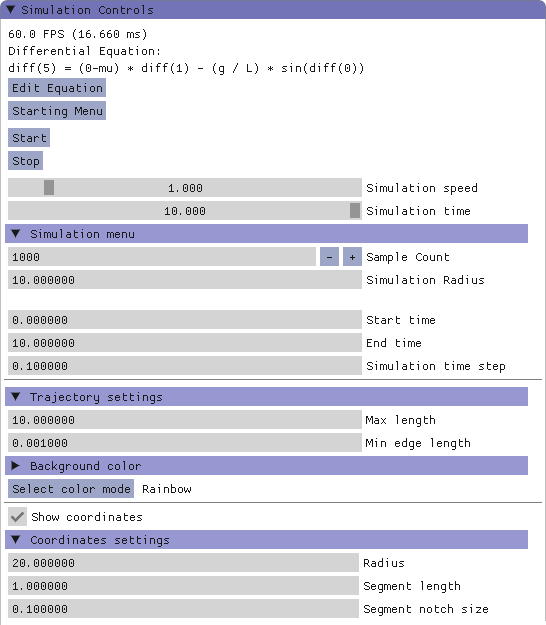
На рис.2 представлен интерфейс проигрывателя визуализации. 

Рис.2. Интерфейс проигрывателся визуализации.

В этот интерфейс входит:

1. Кнопка возврата к редактированию уравнения;
2. Кнопка «Старт», при нажатии на которую создаются фазовые траектории и начинается визуализация;
3. Кнопка «Стоп», очищающая фазовые траектории из памяти;
4. Ползунок скорости проигрывания;
5. Ползунок параметра ;
6. Меню симуляции;
   1. Количество траекторий;
   2. Радиус создания множества начальных значений;
   3. Границы сетки параметра ;
   4. Шаг сетки;
7. Настройка траекторий;
   1. Максимальная длина;
   2. Минимальная длина ребра ломаной;
   3. Цвет заднего фона;
   4. Режим закрашивания траекторий.
8. Переключатель координатных осей;
   1. Радиус осей;
   2. Размер деления;
   3. Размер метки деления.
9. **Примеры фазовых портретов**

Было подобрано несколько примеров дифференциальных уравнений, имеющих показательные фазовые портреты:

1. , пример стабильной точки (рис.3.);
2. , пример нестабильной точки (рис.4.);
3. , пример предельных циклов (рис.5.);
4. , «Осциллятор Ван дер Поля» (рис.6.);
5. , уравнение маятника (рис.7.);
6. , (рис.8.);