Министерство науки и высшего образования РФ
Федеральное государственное автономное образовательное учреждение
высшего образования «Национальный исследовательский Нижегородский
государственный университет им. Н.И. Лобачевского»
Институт информационных технологий, математики и механики

Отчет по лабораторной работе

РЕШЕНИЕ СИСТЕМЫ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ ТРЕТЬЕГО ПОРЯДКА, СОДЕРЖАЩИХ УРАВНЕНИЯ ПЕРВОГО ПОРЯДКА, МЕТОДОМ РУНГЕ - КУТТЫ

Выполнил: Власов Максим Сергеевич, студент группы 381806-1
Проверил: к. ф.-м. н., доцент кафедры ДУМЧА Эгамов А. И.

Нижний Новгород
2021

Содержание

Введение

В связи с развитием новой вычислительной техники инженерная практика наших дней все чаще и чаще встречается с математическими задачами, точное решение которых получить весьма сложно или невозможно. В этих случаях обычно прибегают к тем или иным приближенным вычислениям. Вот почему приближенные и численные методы математического анализа получили за последние годы широкое развитие и приобрели исключительно важное значение.

Данная работа направлена на изучение методов приближенного решения систем дифференциальных уравнений. Основной учебной целью работы является практическое освоение данных вычислительных методов. В ходе выполнения разрабатывается общая форма представления систем, разрабатываются программы работы со подобными структурами, которые могут быть использованы и в других областях приложений.

1. Постановка задачи

Задача: разработать и реализовать приложение, выполняющее приближенное решение задачи Коши для системы обыкновенных дифференциальных уравнений первого порядка, а также построение фазового портрета системы.

Входные данные: система обыкновенных дифференциальных уравнений, интервал, количество шагов, начальные условия для решения задачи Коши.

Выходные данные: значения системы для заданных начальных условий, фазовый портрет.

2. Руководство пользователя

В данном руководстве содержатся пошаговые инструкции по работе с программой, для того чтобы вы могли как можно быстрее приступить к использованию приложения.

1) Запустить файл **VLASOV_LAB_3_RUNGE_KUTTA.exe**. Перед вами откроется главное окно программы, в котором нужно ввести параметры для отрезка и начальные условия для задачи Коши. Система дифференциальных уравнений уже внесена в программу, отображается в левом верхнем углу. (Рис. 1)

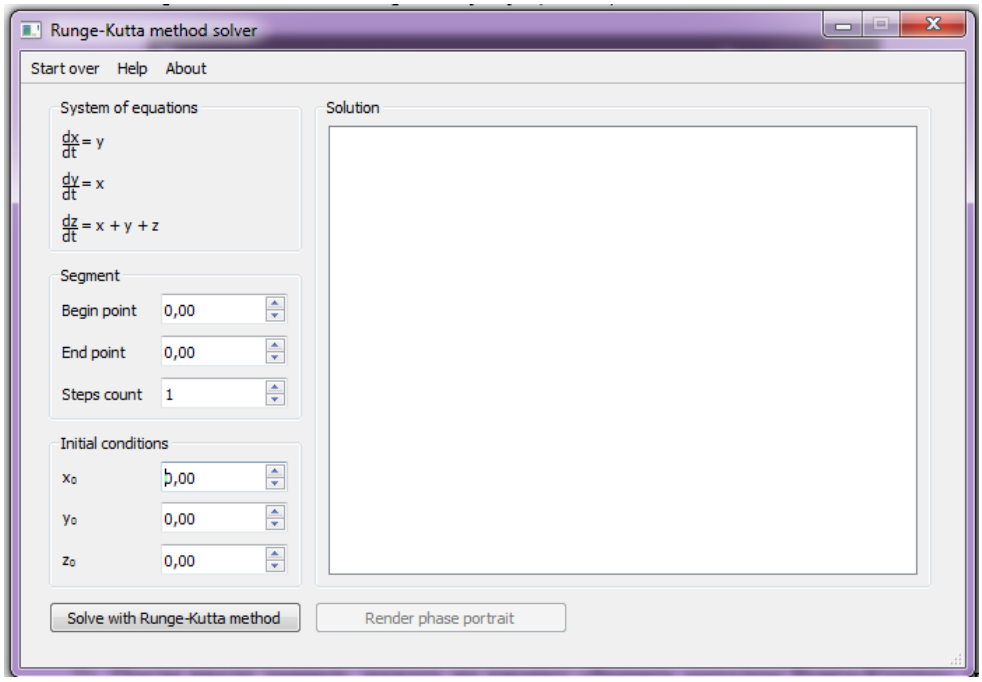


Рис. 1. Главное окно приложения

2) После ввода данных, нажать на кнопку «Solve with Runge-Kutta method», чтобы произвести вычисления для приближенного решения, которое отобразится в таблица справа. (Рис. 2)

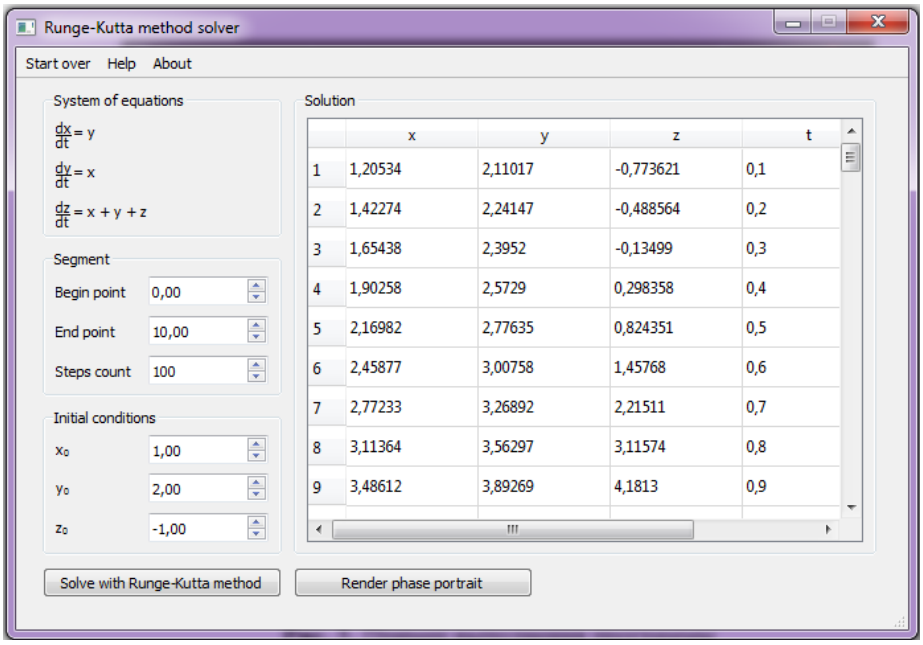


Рис. 2. Пример выполнения программы

3) Можно сравнить приближенное и точное решение, для этого нужно нажать на кнопку «Render phase portrait». После откроется окно с графиками, на котором можно масштабировать графики и двигаться по ним, а также будет выведено точно решение исходя из «Начальных условий». (Рис. 3)

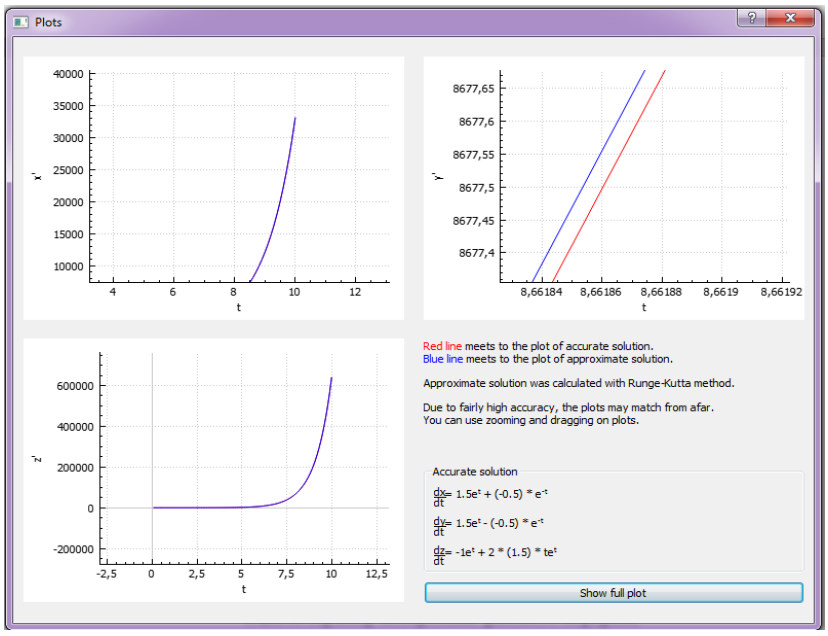


Рис. 3. Пример построения фазового портрета

3. Руководство программиста

3.1. Структура программы

Исходный код программы содержится в следующих модулях:

- main Window.h, mainWindow.cpp – модуль, реализующий главное окно программы;
- rkmethodsolver.h, rkmethodsolver.cpp – модуль, реализующий метод Рунге-Кутты 4 порядка

3.2. Описание алгоритмов

3.2.1. Решение задачи Коши для системы ОДУ методом Рунге-Кутты 4 порядка

Метод Рунге — Кутты четвёртого порядка при вычислениях с постоянным шагом интегрирования столь широко распространён, что его часто называют просто методом Рунге — Кутты.

Системой дифференциальных уравнений называется система вида

$$\begin{cases} \frac{dy_1}{dx} = f_1(x, y_1, \dots, y_n) \\ \frac{dy_2}{dx} = f_2(x, y_1, \dots, y_n) \\ \dots\dots\dots\dots\dots\dots \\ \frac{dy_n}{dx} = f_n(x, y_1, \dots, y_n) \end{cases}$$

где x – независимый аргумент, y_i – зависимая функция, $i = \overline{1,n}$, $y_i|_{x=x_0} = y_{i0}$ – начальные условия.

Тогда приближенное значение в последующих точках вычисляется по итерационной формуле:

$$y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

Вычисление нового значения проходит в четыре стадии:

$$k_{i1} = h * f_1(x_j, y_{1j}, \dots, y_{nj})$$

$$k_{i2} = h * f_1(x_j + \frac{h}{2}, y_{1j} + \frac{k_{i1}}{2}, \dots, y_{nj} + \frac{k_{n1}}{2})$$

$$k_{i3} = h * f_1(x_j + \frac{h}{2}, y_{1j} + \frac{k_{i2}}{2}, \ldots, y_{nj} + \frac{k_{n2}}{2})$$

$$k_{i4} = h * f_1(x_j + \frac{h}{2}, y_{1j} + k_{i3}, \ldots, y_{nj} + k_{n3})$$

$$y_{ij+1} = y_{ij} + \frac{h}{6}(k_{i1} + 2k_{i2} + 2k_{i3} + k_{i4})$$

$$x_{j+1} = x_j + h$$

где h — величина шага сетки по x.

Этот метод имеет четвёртый порядок точности. Это значит, что ошибка на одном шаге имеет порядок $O(h^5)$, а суммарная ошибка на конечном интервале интегрирования имеет порядок $O(h^4)$.

3.2.2. Применение алгоритма

3.2.2.1. Выбор системы ДУ и ее точное решение

Для решения была выбрана следующая система:

$$\begin{cases} \frac{dx}{dt} = y; \\ \frac{dy}{dt} = x; \\ \frac{dz}{dt} = x + y + z. \end{cases}$$

Решение:

1. Возьмем производную от первого уравнения системы. Получаем $\frac{d^2x}{dt^2} = \frac{dy}{dt} = x$;
2. Решаем линейное уравнение 2 порядка $\frac{d^2x}{dt^2} - x = 0$. Получаем $x = C_1e^t + C_2e^{-t}$.
3. Делая подстановку, полученного x во второе уравнение системы получаем, что $\frac{dy}{dt} = C_1e^t + C_2e^{-t}$. Интегрируем это выражение, тем самым получаем $y = C_1e^t - C_2e^{-t}$.
4. Аналогично поступаем с третьим уравнением системы. $z = C_3e^t + 2C_1te^t$.
5. Конечное решение системы ДУ $\begin{cases} x = C_1e^t + C_2e^{-t}; \\ y = C_1e^t - C_2e^{-t}; \\ z = C_3e^t + 2C_1te^t. \end{cases}$

3.2.2.2. Сравнение значений точного решения с полученными методом Рунге-Кутты.

Для сравнения выберем интервал, начальные условия и количество шагов, введя эти параметры в программу, получим приближенные значения решения системы.

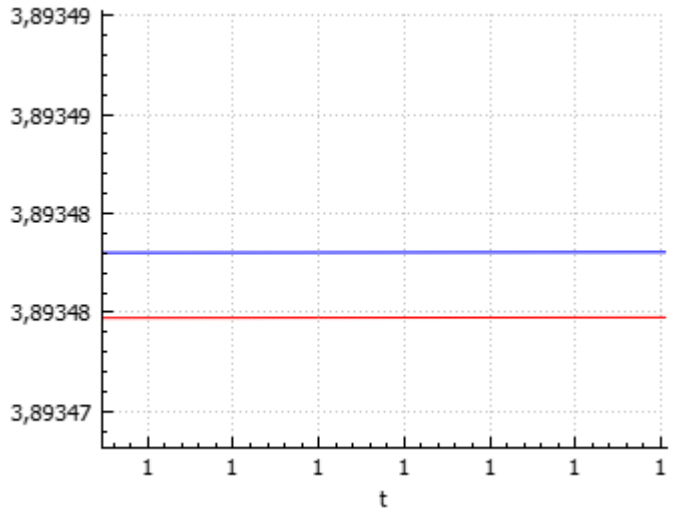
Результаты при интервале от 0 до 10, начальных условиях $x(0) = 1, y(0) = 2, z(0) = -1$, количество шагов 100.

Номер шага	**x**	**y**	**z**	**t**
1	1,20534	2,11017	-0,773621	0,1
10	3,89348	4,67268	5,43653	1
100	33039,4	33039,4	638760	10

Посчитаем значения точного решения в данных точках, учитывая начальные условия.

x	**y**	**z**	**t**
1,20534	2,11017	-0,773621	0,1
3,89348	4,67268	5,43653	1
33039,7	33039,7	638768	10

Графическое сравнение решений:



Учитывая схожесть графических решений, приведем сравнения только для x.

Вывод: проанализировав таблицы и графики двух решений приходим к тому, что решение системы ДУ методом Рунге – Кутты позволяет с высокой точностью вычислять значения системы ДУ с заданными начальными условиями.

3.3. Описание структур данных и функций

3.3.1. Класс MainWindow

```
class MainWindow : public QMainWindow
{
public:
    explicit MainWindow(AbstractEquationSystem* system, QWidget *parent =
nullptr);
    ~MainWindow();

public slots:
    void showAboutDialog();
    void showHelpDialog();
    void startOver();
    void solve();
};
```

3.3.1.1. Описание методов класса:

- `explicit MainWindow(QWidget *parent = nullptr)` – конструктор инициализации.
- `~MainWindow()` – деструктор.
- `void showAboutDialog()` – открывает окно «О программе»
- `void showHelpDialog()` – открывает окно «Помощь»
- `void startOver()` – очищает поля ввода
- `void solve()` – запускает решение методом Рунге-Кутты

3.3.2. Класс RKMethodSolver

```
class RKMethodSolver
{
    AbstractEquationSystem* m_system;
public:
    RKMethodSolver(AbstractEquationSystem* system);
    ~RKMethodSolver() = default;

    std::vector<std::array<double, 4>> solve(double a, double b, int n,
    const std::array<double, 3>& init_conditions);
};
```

3.3.2.1. Поля класса:

m_system – система обыкновенных дифференциальных уравнений

3.3.2.2. Описание методов класса:

- `RKMethodSolver(AbstractEquationSystem* system)` – конструктор;
- `~RKMethodSolver()` – деструктор;

- `std::vector<std::array<double, 4>> solve(double a, double b, int n, const std::array<double, 3>& init_conditions)` – решение методом Рунге-Кутты

3.3.3. Класс PlotsDialog

```
class PlotsDialog : public QDialog
{
public:
    explicit PlotsDialog(const std::vector<std::array<double, 4>>& approx,
                         AbstractAccurateSolution* accur,
                         QWidget *parent = nullptr);
    ~PlotsDialog();
public slots:
    void showFullPlot();
      };
```

3.3.3.1. Описание методов класса:

- `PlotsDialog(const std::vector<std::array<double, 4>>& approx, AbstractAccurateSolution* accur, QWidget *parent = nullptr)` – конструктор;
- `~PlotsDialog()` – деструктор;
- `showFullPlot` – сбрасывает масштаб графиков.

Заключение

В ходе выполнения работы была реализована программа, которая приближенно решает задачу Коши для системы обыкновенных дифференциальных уравнений первого порядка и строит фазовый портрет, изучен метод Рунге-Кутты четвертого порядка. Программа полностью соответствует описанным методам и решает поставленную задачу решения систем линейных дифференциальных уравнений.

Список используемых источников

1. https://ru.wikipedia.org/wiki/Метод_Рунге_—_Кутты
2. Самарский А. А., Гулин А. В. Численные методы: Учеб, пособие для вузов,— М.: Наука. Гл. ред. физ-мат. лит., 1989.— 432 с.
3. Самарская Елена Александровна Задачи и упражнения по численным методам: Учебное пособие. — М.: Эдиториал УРСС, 2000. - 208 с
4. Qt 5.15 Reference Pages [Электронный ресурс]: Документация к фреймворку Qt. URL: https://doc.qt.io/qt-5/reference-overview.html