ИНСТИТУТ ТРАНСПОРТА И СВЯЗИ

ФАКУЛЬТЕТ КОМПЬЮТЕРНЫХ НАУК И
ТЕЛЕКОММУНИКАЦИЙ

Лабораторная работа №4
По дисциплине
«Численные методы и прикладное программирование»

Тема:
«Численные методы решения дифференциальных уравнений
первого порядка»

Работу выполнили:

Дзенис Ричард
Кобелев Денис
Якушин Владислав

Рига
2017 г.

Содержание

1 Формулировка задания

В данной лабораторной работе требуется реализовать два метода решения дифференциальных уравнений: метод Эйлера и индивидуальный метод (метод Рунге-Кутты 4-ого порядка). Само дифференциальное уравнение выдается преподавателем во время проведения лабораторных работ. Для того чтобы реализовать алгоритмы, требуется предусмотреть ряд входных параметров.

2 Метод Эйлера

```
#include "common.h"

std::vector<point> solve(
        FTY f, double y0,
        double left, double right,
        double step)
{
    std::vector<point> points { {left, y0} };
    for (double t = left + step; t < right + step; t += step) {
        auto y_k = points.back().y;
        points.emplace_back(point{ t, y_k + step * f(t, y_k) });
    }
    return points;
}

#include "main.cpp"
```

2.1 Результаты работы метода

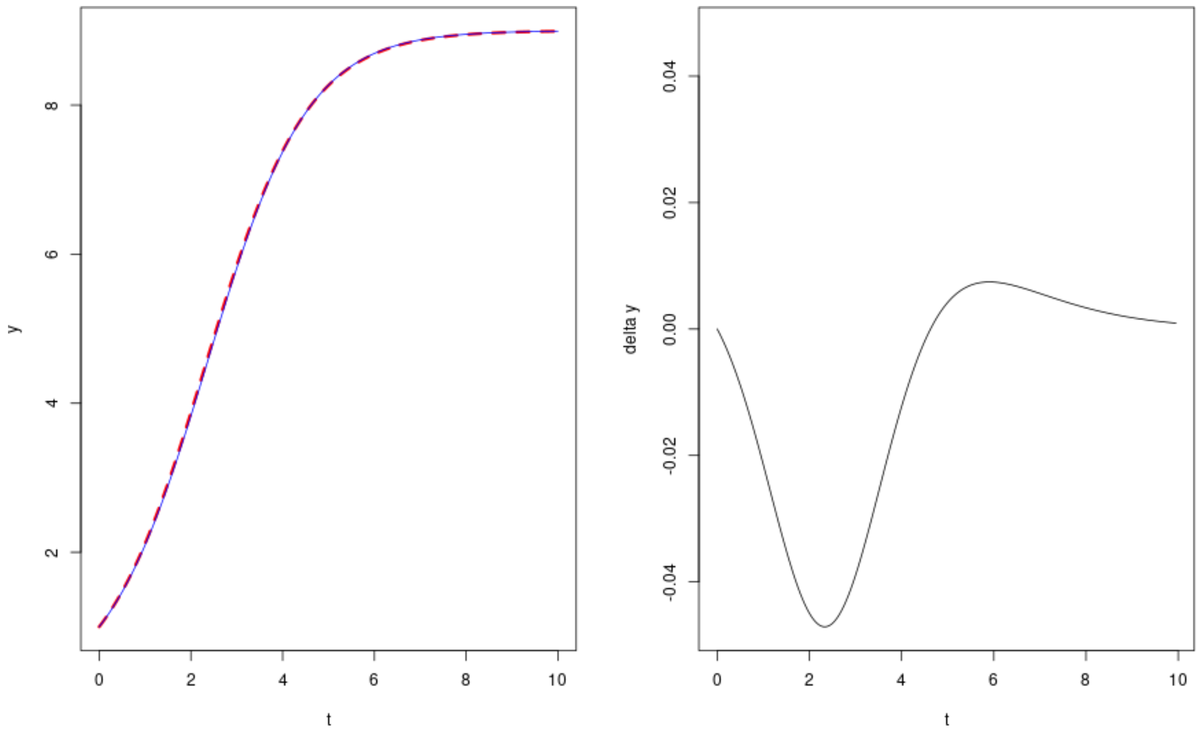

Рис. 1: График решения дифференциального уравнения $y' = 0.9y - 0.1y^2$ и ошибка решения на интервале $[0; 10]$, с начальным условием $y'(0) = 1$.

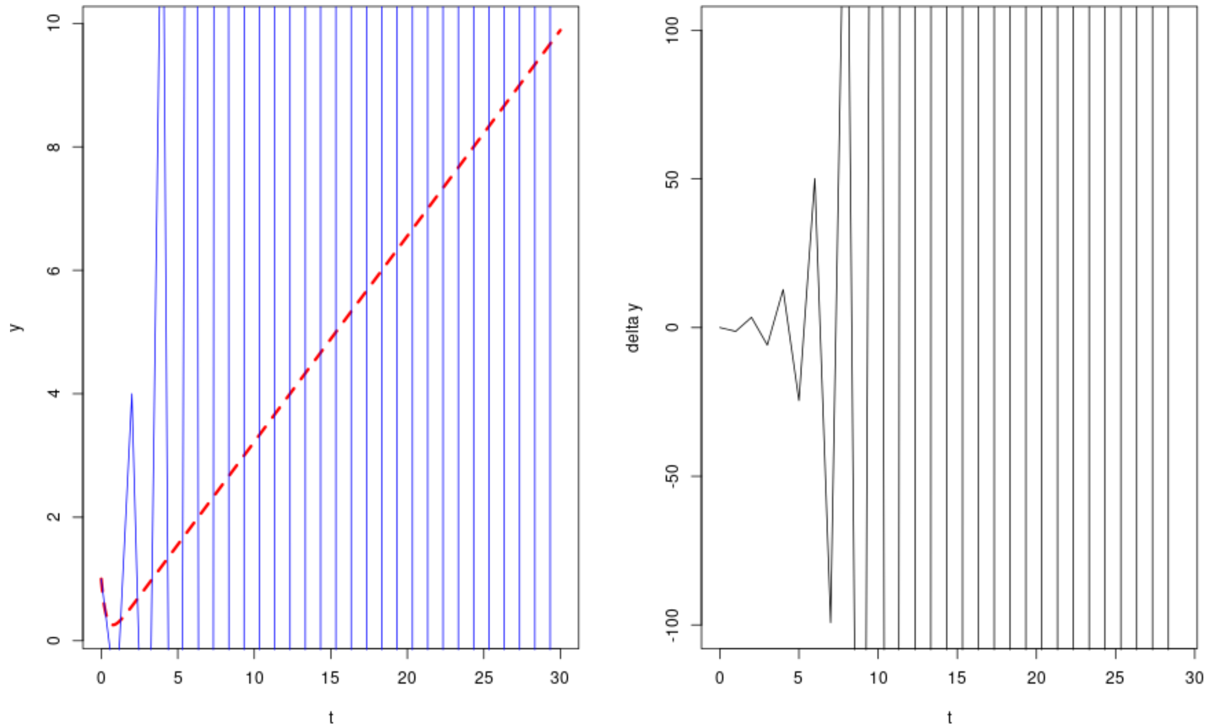

Рис. 2: График решения дифференциального уравнения $y' = -3y + t$ и ошибка решения на интервале $[0; 30]$, с начальным условием $y'(0) = 1$.

3 Метод Рунге-Кутты 4-ого порядка (оригинальный)

В данном методе для подсчёта y_{k+1} использовалась «оригинальная» формула:

$$y_{k+1} = y_k + \frac{h}{6}\left[f_1 + 2f_2 + 2f_3 + f_4\right] \tag{1}$$

```
#include "common.h"

std::vector<point> solve(
        FTY f, double y0,
        double left, double right,
        double step)
{
    std::vector<point> points { {left, y0} };
    for (double t = left + step; t < right + step; t += step) {
        auto [t_k, y_k] = points.back();
        auto f_1 = f(t_k, y_k);
        auto f_2 = f(t_k + step / 2.0, y_k + step / 2.0 * f_1);
        auto f_3 = f(t_k + step / 2.0, y_k + step / 2.0 * f_2);
        auto f_4 = f(t, y_k + step * f_3);
        points.emplace_back(point{
                t,
                y_k + step / 6.0 * (f_1 + 2 * f_2 + 2 * f_3 + f_4)
                });
    }
    return points;
}

#include "main.cpp"
```

3.1 Результаты работы метода

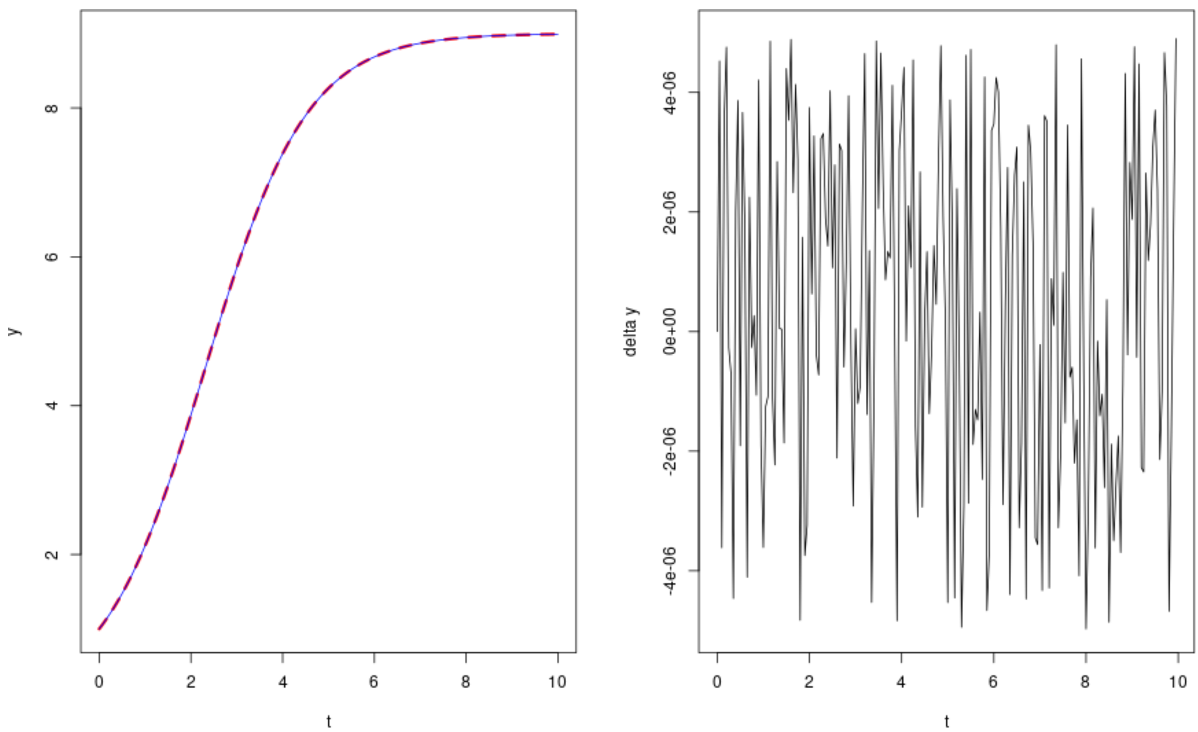

Рис. 3: График решения дифференциального уравнения $y' = 0.9y - 0.1y^2$ и ошибка решения на интервале $[0; 10]$, с начальным условием $y'(0) = 1$.

4 Метод Рунге-Кутты 4-ого порядка $(4f_3)$

В данном методе для подсчёта y_{k+1} использовалась f_3 вместо f_2:

$$y_{k+1} = y_k + \frac{h}{6}[f_1 + 4f_3 + f_4] \tag{2}$$

4.1 Результаты работы метода

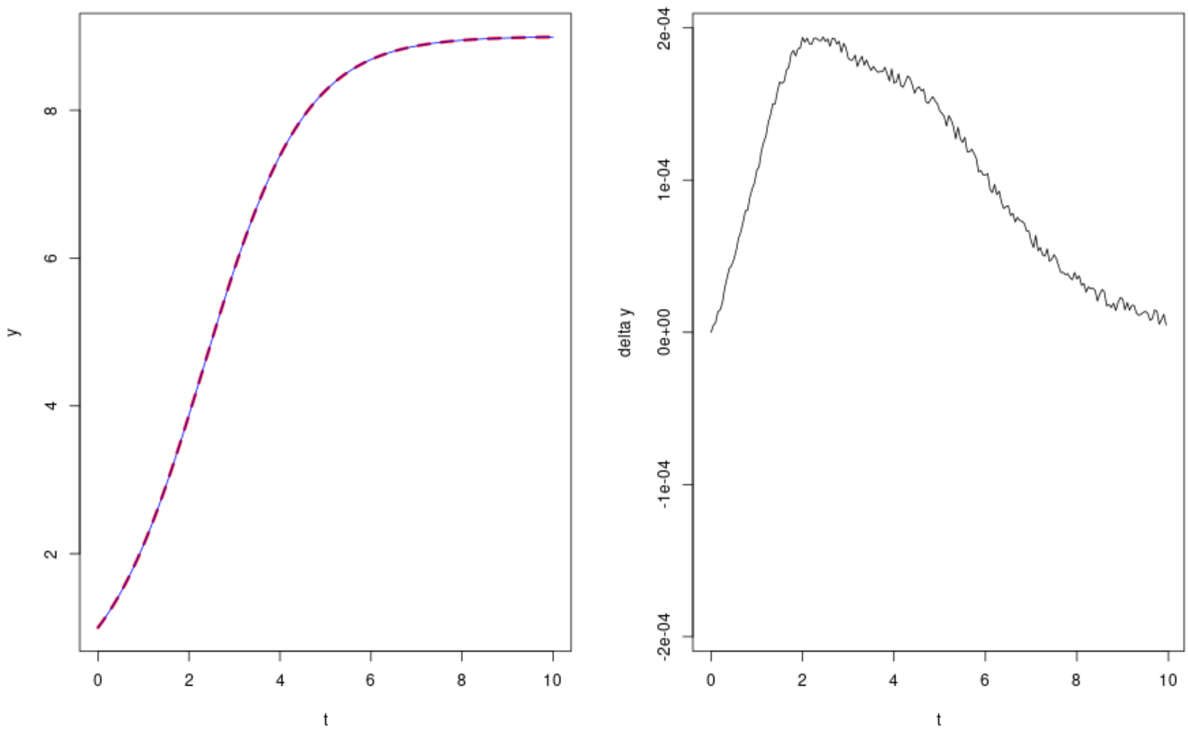

Рис. 4: График решения дифференциального уравнения $y' = 0.9y - 0.1y^2$ и ошибка решения на интервале $[0; 10]$, с начальным условием $y'(0) = 1$.

5 Метод Рунге-Кутты 4-ого порядка $(4f_2)$

В данном методе для подсчёта y_{k+1} использовалась f_2 вместо f_3:

$$y_{k+1} = y_k + \frac{h}{6}\left[f_1 + 4f_2 + f_4\right] \tag{3}$$

5.1 Результаты работы метода

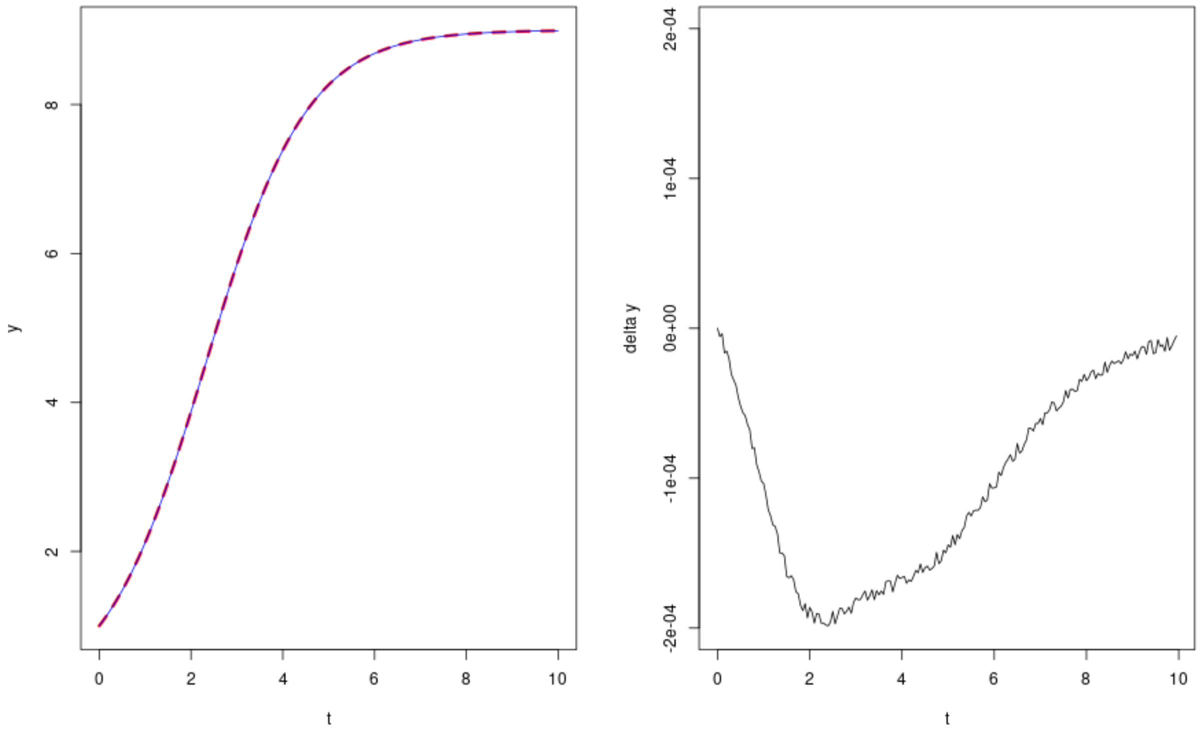

Рис. 5: График решения дифференциального уравнения $y' = 0.9y - 0.1y^2$ и ошибка решения на интервале $[0; 10]$, с начальным условием $y'(0) = 1$.

6 Выводы

В ходе данной лабораторной работы было необходимо разработать программу решающую дифференциальное уравнение методами Эйлера и Рунге-Кутты 4-ого порядка с отображением ошибки обоих методов. В результате были реализованы программы способные решать диф. ур., а также скрипты для построения графиков решения, и ошибки.

Как можно видеть из графиков, метод Эйлера при данных условиях справляется с задачей, но обладает свойством накапливать ошибку. В первой ситуации ошибка накапливается и стабилизируется на определённом уровне. Так же, как можно видеть на рис. 2 метод Эйлера в некоторых ситуациях может быстро дестабилизироваться при достаточно большом шаге.

Метод Рунге-Кутты в данном плане является более стабильным выбором и имеет гораздо меньшую ошибку, а среднее значение ошибки стремиться к 0, что гарантирует, что функция в среднем будет достаточно близка к теоретической (при отсутствии t в диф. ур.).

Для проверки особенности влияния составляющий функций в формуле Рунге-Кутты, были составлены 2 формулы: в первой использовалось только значение центральной точки, во второй только значение коррекции для центральной точки.

Глядя на рис. 4 и рис. 5, становиться ясно, что поведение ошибки в обоих случаях имеют симметричный характер относительно друг друга, и отклонены в одном направлении. При использовании стандартного подхода – ошибки от обоих функций противодействуют друг другу, что приводит среднюю ошибку к нулю. Так же это позволяет уменьшить максимальный размер ошибки в несколько раз, что можно видеть на рис. 3.