Московский Государсвенный Университет
имени М.В.Ломоносова
Факультет Вычислительной математики и Кибернетики

Компьютерный практикум по учебному курсу "Введение в численные методы"

Задание №2

Отчет
о выполненном задании

студента 204 учебной группы факультета ВМК МГУ
Воробьева Сергея Юрьевича

Москва, 2020

Оглавление

Глава 1

Введение

1.1 Цели

Изучить методы Рунге-Кутта второго и четвертого порядка точности, применяемые для численного решения задач Коши для дифференциального уравнения (или системы) первого порядка:

- Решить заданные задачи Коши для методоми Рунге-Кутта второго и четвертого порядка точности, апроксимировав дифференциальну задачу соответствующей разностной схемой на равномерной сетке; полученное конечно-разностное уравнение просчитать численно
- Найти численное решение и построить его график
- Сравнить численное решение с точным на различных тестах, используя wolframalpha.com

Изучить метод прогонки решения краевой задачи для дифференциального уравнения второго порядка:

- Решить краевую заданную задачу методом конечных разностей, аппроксимировав ее разностной схеой второго порядка точности на равномерной сетке; полученную систему конечно-разностных уранвений решить методом прогонки
- Найти численное решение задачи и построить его график
- Сравнить численное решение и с точным на различных тестах, используя wolframalpha.com

1.2 Постановка задачи

1.2.1 Задача Коши

Рассматривается ОДУ первого порядка, разрешённое относительно производной и имеющее вид, с дополнительным начальным условием в точка a:

$$\begin{cases} \frac{dy}{dx} = f(x,y) \quad a \le x \le b \\ y(a) = y_0 \end{cases}$$

Необходимо найти решение данной задачи Коши в предположении, что правая часть уравнения $f = f(x,y)$ таковы, что гарантирует существование и единственность решения задачи Коши.

1.2.2 Задача Коши для системы

Рассматривается система линейных ОДУ первого порядка, разрешённых относительно производной, с дополнительными условиями в точке a :

$$\begin{cases} \frac{dy_1}{dx} = f_1(x, y_1, y_2) \\ \frac{dy_2}{dx} = f_2(x, y_1, y_2) \qquad a \le x \le b \\ y_1(a) = y_1^0, y_2(a) = y_2^0 \end{cases}$$

Необходимо найти решение данной задачи Коши в предположении, что правые части уравнений таковы, что гарантируют существование и единственность решения задачи Коши для системы.

1.2.3 Краевая задача

Рассматривается краевая задача для дифференциального уравнения второго порядка с дополнительными условиями в граничных точках:

$$\begin{cases} y" + p(x)y' + q(x)y = f(x) \\ \sigma_1 y(a) + \gamma_1 y'(a) = \delta_1 \qquad a \le x \le b \\ \sigma_2 y(b) + \gamma_2 y'(b) = \delta_2 \end{cases}$$

Необходимо найти решение данной краевой задачи.

Глава 2

Алгоритмы решения

2.0.1 Задача Коши

Будем использовать следующие формулы для численного решения задачи Коши, приближающие точное решение с четвёртым порядком точности относительно диаметра разбиения отрезка, на котором решается поставленная задача.
Положим:

- n - число точек разбиения отрезка
- $h = \frac{a-b}{n}$ - диаметр разбиения отрезка
- $x_i = a + h * i, y_i = y(x_i), 0 \leq i \leq n$ - сетка и сеточная функция

Метод Рунге-Кутты 2 порядка точности для рекуррентного вычисления сеточной функции примет следующий вид:

$y_{i+1} = y_i + \frac{h}{2}(f(x_i + f(x_i + h, y_i + h * f(x_i))))$

Метод Рунге-Кутты 4 порядка точности для рекуррентного вычисления сеточной функции примет следующий вид:

$$\begin{cases} k_1 = f(x_i, y_i) \\ k_2 = f(x_i + \frac{h}{2}, y_i + \frac{h}{2}k_1) \\ k_3 = f(x_i + \frac{h}{2}, y_i + \frac{h}{2}k_2) \\ k_4 = f(x_i + h, y_i + hk_3) \\ y_{i+1} = y_i + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \end{cases}$$

2.0.2 Задача Коши для системы

Метод Рунге-Кутты 2 порядка для рекуррентного вычисления сеточной функции примет следующий вид:

$$\begin{cases} k_1 = f(x_i, y_1^i, y_2^i) \\ k_2 = f(x_i + \frac{h}{2}, y_1^i + \frac{h}{2}k_1, y_2^i + \frac{h}{2}k_{21}) \\ k_{21} = f(x_i, y_1^i, y_2^i) \\ k_{22} = f(x_i + \frac{h}{2}, y_1^i + \frac{h}{2}k_1, y_2^i + \frac{h}{2}k_{21}) \\ y_1^{i+1} = y_1^i + hk_2 \\ y_2^{i+1} = y_2^i + hk_{22}) \end{cases}$$

Метод Рунге-Кутты 4 порядка для рекуррентного вычисления сеточной функции примет следующий вид:

$$\begin{cases} k_1 = f(x_i, y_1^i, y_2^i) \\ k_2 = f(x_i + \frac{h}{2}, y_1^i + \frac{h}{2}k_1, y_2^i + \frac{h}{2}k_{21}) \\ k_3 = f(x_i + \frac{h}{2}, y_1^i + \frac{h}{2}k_2, y_2^i + \frac{h}{2}k_{22}) \\ k_4 = f(x_i + h, y_1^i + hk_3, y_2^i + hk_{23}) \\ k_{21} = f(x_i, y_1^i, y_2^i) \\ k_{22} = f(x_i + \frac{h}{2}, y_1^i + \frac{h}{2}k_1, y_2^i + \frac{h}{2}k_{21}) \\ k_{23} = f(x_i + \frac{h}{2}, y_1^i + \frac{h}{2}k_2, y_2^i + \frac{h}{2}k_{22}) \\ k_{24} = f(x_i + h, y_1^i + hk_3, y_2^i + hk_{23}) \\ y_1^{i+1} = y_1^i + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \\ y_2^{i+1} = y_2^i + \frac{h}{6}(k_{21} + 2k_{22} + 2k_{23} + k_{24}) \end{cases}$$

2.0.3 Краевая задача

Для решения данной задачи запишем заданное дифференциальное уравнение в узлах сетки и краевые условия:

$$\begin{cases} y_i'' + p_i y_i' + q_i y_i = f_x, x_i = a + i\frac{b-a}{n} \quad 0 \le i \le n \\ \sigma_1 y_0 + \gamma_1 y_0' = \delta_1 \\ \sigma_2 y_n + \gamma_2 y_n' = \delta_2 \end{cases}$$

Для $1 \le i \le n-1$ существует следующее разностное приближение для первой и второй производной и самой сеточной функции:

$$\begin{cases} y_i'' = \frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} \\ y_i' = \frac{y_{i+1} - y_{i-1}}{2h} \end{cases}$$

В результате подстановки этих разностных отношений в начальное уравнение в виде сеточной функции получим линейную систему из n+1 уравнений с n+1 неизвестными $y_0, y_1, ..., y_n$:

$$\begin{cases} y_i'' = \frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} + p_i\frac{y_{i+1} - y_{i-1}}{2h} + q_i y_i = f_i \quad 1 \le i \le n-1 \\ \sigma_1 y_0 + \gamma_1 \frac{y_0 - y_0}{h} = \delta_1 \\ \sigma_2 y_1 + \gamma_2 \frac{y_n - y_{n-1}}{h} = \delta_2 \end{cases}$$

Явно выписав коэффициенты перед $y_0, y_1, ..., y_n$, получим систему с трехдиагональной матрицей:

$$A = \begin{bmatrix} \sigma_1 - \gamma_1/h & \gamma_1/h & 0 & 0 & ... & \delta_1 \\ 1 - h/2p_1 & q_1 * h^2 - 2 & 1 + h/2 * p_1 & 0 & ... & f_1 h^2 \\ 0 & 1 - h/2p_2 & q_2 * h^2 - 2 & 1 + h/2 * p_2 & ... & f_2 h^2 \\ 0 & 0 & 0 & ... & ... & f_k h^2 \\ 0 & 0 & 0 & ... & ... & f_l h^2 \\ 0 & ... & 1 - h/2p_{n-1} & q_{n-1} * h^2 - 2 & 1 + h/2 * p_{n-1} & f_{n-1} h^2 \\ 0 & ... & 0 & -\gamma_2/h & \sigma_2 + \gamma_2/h & \delta_2 \end{bmatrix}$$

Для решения полученной системы используется метод прогонки. Этот метод существенно упрощает решение системы с трехдиагональной матрицей и имеет сложность O(n):

Переобозначая коэффициенты перед y получим:

$$\begin{cases} C_0 y_0 + B_0 y_1 = F_0 \\ A_i y_{i-1} + C_i y_i + B_i y_{i+1} = F_i \quad 1 \le i \le n-1 \\ A_n y_{n-1} + C_n y_n = F_n \end{cases}$$

$$\begin{cases} \alpha_0 = -\frac{B_0}{C_0} \\ \beta_0 = \frac{F_0}{C_0} \\ \alpha_i = -\frac{B_i}{C_i + A_i \alpha_{i-1}} \\ \qquad\qquad\qquad\qquad 1 \le i \le n-1 \\ \beta_i = \frac{F_i - A_i \beta_{i-1}}{C_i + A_i \alpha_{i-1}} \\ \\ y_n = \frac{F_n - A_n \beta_{n-1}}{C_n + A_n \alpha_{n-1}} \\ y_i = \beta_i + \alpha_i * y_{i+1} \quad 0 \le i \le n-1 \end{cases}$$

Глава 3

Программа

В качестве языка программирования был выбран Python3 в виду его гибкости и простоты синтаксиса. Каждый численный метод реализован отдельной функцией, которые представлены ниже. В качестве вспомогательного модуля использовался модуль NumPy. Его структуры данных позволяют удобно хранить и преобразовывать матрицы и вектора.

```
def RK2(x_0, y_0, f, l, n):
    # Рунге-Кутта 2-го порядка
    h = l / n # Шаг сетки
    grid = dict() # Храним сетку в словаре
    grid[x_0] = y_0 # Начальное условие
    for i in range(0, n): # Заполняем оставшиеся n значений
        x_i = x_0 + h*i
        y_i = grid[x_i]
        t = f(x_i, y_i)
        grid[x_0 + h*(i+1)] = y_i + (h / 2) * (t + f(x_i + h, y_i + h*t))
    return grid
```

```
def RK4(x_0, y_0, f, l, n):
    # Рунге-Кутта 4-го порядка
    h = l / n # Шаг сетки
    grid = dict() # Храним сетку в словаре

    grid[x_0] = y_0 # Начальное условие
    for i in range(0, n): # Заполняем оставшиеся n значений
        x_i = x_0 + h*i
        y_i = grid[x_i]

        k1 = f(x_i, y_i)
        k2 = f(x_i + (h/2), y_i + (h/2)*k1)
        k3 = f(x_i + (h/2), y_i + (h/2)*k2)
        k4 = f(x_i + h, y_i + h*k3)

        grid[x_0 + h*(i+1)] = y_i + (h / 6) * (k1 + 2*k2 + 2*k3 + k4)
    return grid
```

```python
def RKsys(x0, u_prev, v_prev, l, n, f1, f2):
    # Рунге-Кутта 4-го порядка для систем
    grid = [] # Храним сетку в списке
    h = l / n # Шаг сетки
    for i in range(0, n): # Заполняем оставшиеся n значений
        x_prev = x0 + i * h
        x_cur = x0 + (i + 1) * h

        k1 = h * f1(x_prev, u_prev, v_prev)
        l1 = h * f2(x_prev, u_prev, v_prev)

        k2 = h * f1(x_prev + h / 2, u_prev + k1 / 2, v_prev + l1 / 2)
        l2 = h * f2(x_prev + h / 2, u_prev + k1 / 2, v_prev + l1 / 2)

        k3 = h * f1(x_prev + h / 2, u_prev + k2 / 2, v_prev + l2 / 2)
        l3 = h * f2(x_prev + h / 2, u_prev + k2 / 2, v_prev + l2 / 2)

        k4 = h * f1(x_prev + h, u_prev + k3, v_prev + l3)
        l4 = h * f2(x_prev + h, u_prev + k3, v_prev + l3)

        u_cur = u_prev + (k1 + 2 * k2 + 2 * k3 + k4) / 6
        v_cur = v_prev + (l1 + 2 * l2 + 2 * l3 + l4) / 6

        grid.append([x_cur, u_cur, v_cur])
        u_prev = u_cur
        v_prev = v_cur
    return grid
```

```
def edge_problem(x_0, x_n, p, q, f, sigma, gamma, delta, n): # Решение краевой задачи
    h = (x_n - x_0) / n # Шаг сетки
    def A_coef(i):
        return (1/(h**2)) - (p(x_0 + i*h) / (2 * h))
    def C_coef(i):
        return -(2/(h**2)) + q(x_0 + i*h)
    def B_coef(i):
        return (1/(h**2)) + (p(x_0 + i*h) / (2 * h))
    def F_coef(i):
        return f(x_0 + i*h)
    grid = dict() # Храним сетку в словаре
    # Промежуточные коэф-ты
    alpha = np.empty(n + 1)
    beta = np.empty(n + 1)

    alpha[0] = 0
    beta[0] = 0
    alpha[1] = -gamma[0] / (sigma[0] * h - gamma[0])
    beta[1] = delta[0] / (sigma[0] - (gamma[0]/h))

    for i in range(1, n): # Вычисление прогоночных коэф-тов
        alpha[i + 1] = -B_coef(i) / (A_coef(i) * alpha[i] + C_coef(i))
        beta[i + 1] = (F_coef(i) - A_coef(i) * beta[i]) / (A_coef(i) * alpha[i] + C_coef(i))

    matrix = np.empty((2, 2)) # Коэф-ты системы
    target = np.empty(2) # Свободные члены

    matrix[0][0] = 1
    matrix[0][1] = -alpha[n]
    matrix[1][0] = 1
    matrix[1][1] = -(sigma[1] * h + gamma[1]) / gamma[1]

    target[0] = beta[n]
    target[1] = -delta[1] * h / gamma[1]
    sol = np.linalg.solve(matrix, target)
    grid[x_0 + n*h] = sol[0]
    grid[x_0 + (n-1)*h] = sol[0]
    for i in range(n - 2, -1, -1):
        grid[x_0 + i*h] = alpha[i+1] * grid[x_0 + (i+1)*h] + beta[i+1]
    return grid
```

```
# Решение краевой задачи
def y3(x):
    return 0.666667*(x**2 + 0.333333*x + 0.704467*e**(x/4)*np.sin((sqrt(47)*x)/4) + 0.927242*e**
(x/4)*np.cos((sqrt(47)*x)/4) - 0.611111)

p = lambda x: 1/2
q = lambda x: 3
f = lambda x: 2*x**2

x = np.linspace(1, 1.3, 50)
y_real = y3(x)
grid3 = method(1.3, 1, p, q, f, [1, 1], [0, -2], [1, 0.6], 50)
test3 = np.array([[i[0], i[1]] for i in grid3.items()])
plt.figure(figsize=(12,7))
plt.scatter(test3[:, 0], test3[:, 1], label = 'solve')
plt.scatter(x, y_real, label = 'Wolfram Alpha')
plt.legend()
plt.show()
```

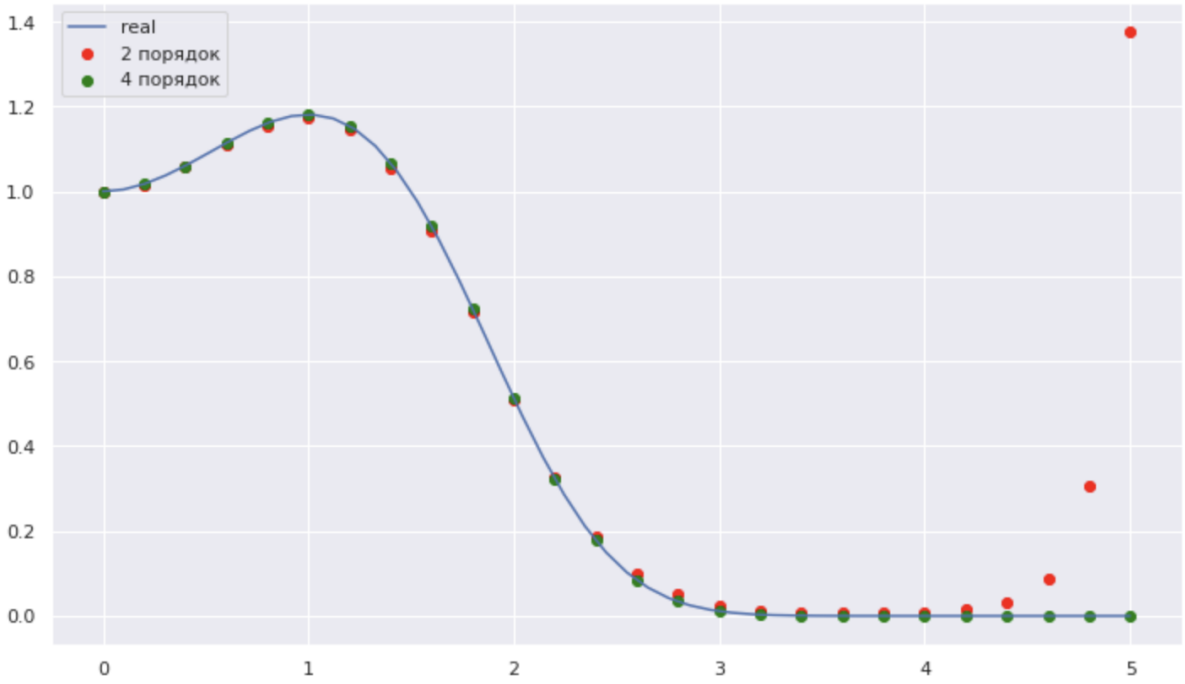
Глава 4

Тестирование

Тестирование правильности решения задач проводилось как с помощью сравнения (где это возможно) результатов работы программы с данным в условии ответом, так и с решениями системы Wolfram Alpha. Первый тест взят из таблицы 1, №6
Второй тест взят из таблицы 2, №17
Третий тест взят из подварианта 2, №9

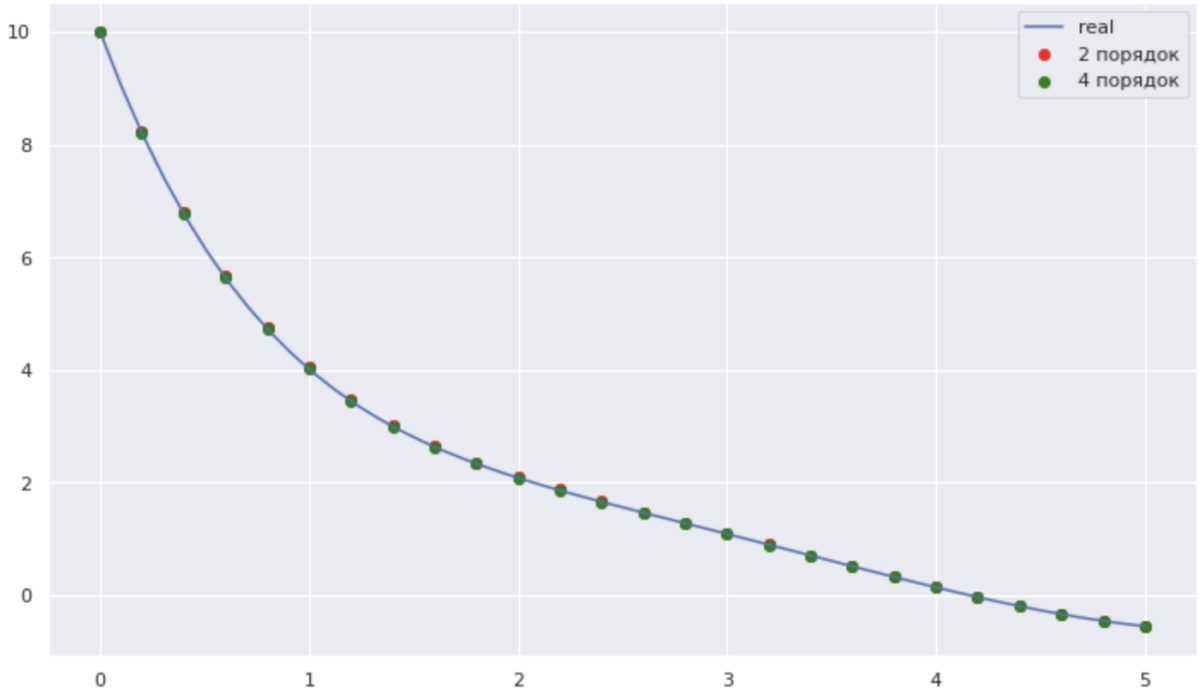
4.0.1 Тест 1

$$f(x, y) = xy - x^2y, y(0) = 3$$



4.0.2 Тест 1 (Дополнительный)

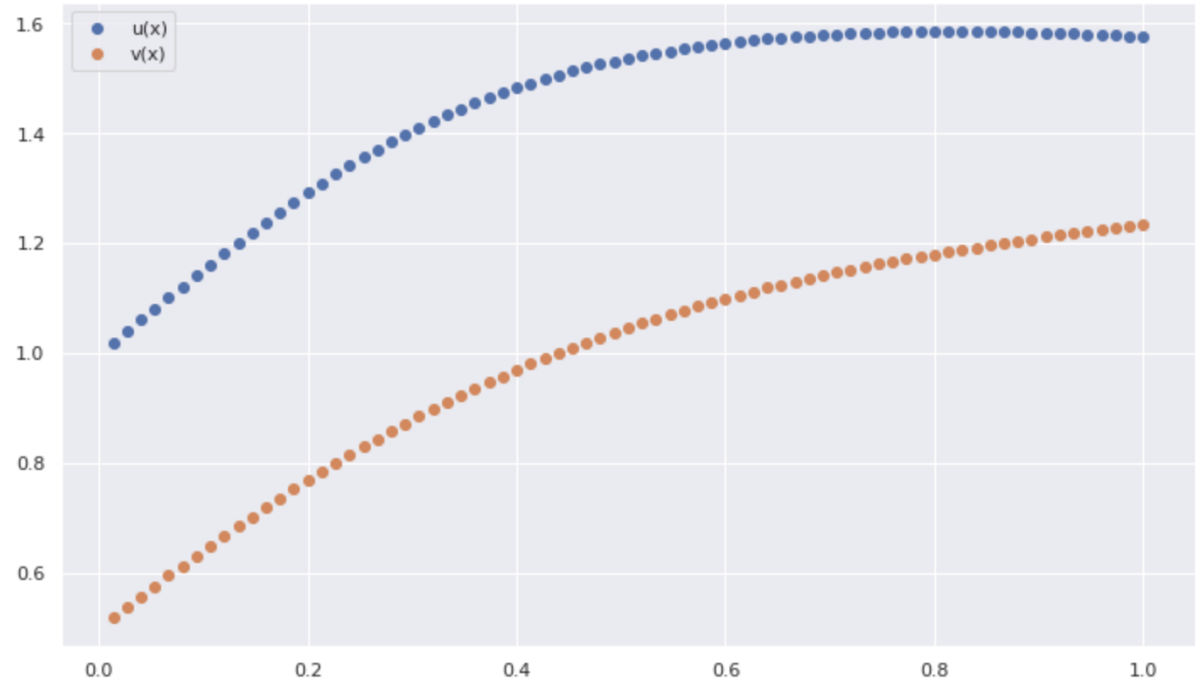
$$f(x, y) = sin(x) - y, y(0) = 10$$



4.0.3 Тест 2

$$\begin{cases} sin(1.4u^2) - x + v \\ x + u - 2.2v^2 + 1 \\ u(0) = 1 \\ v(0) = 0.5 \end{cases}$$

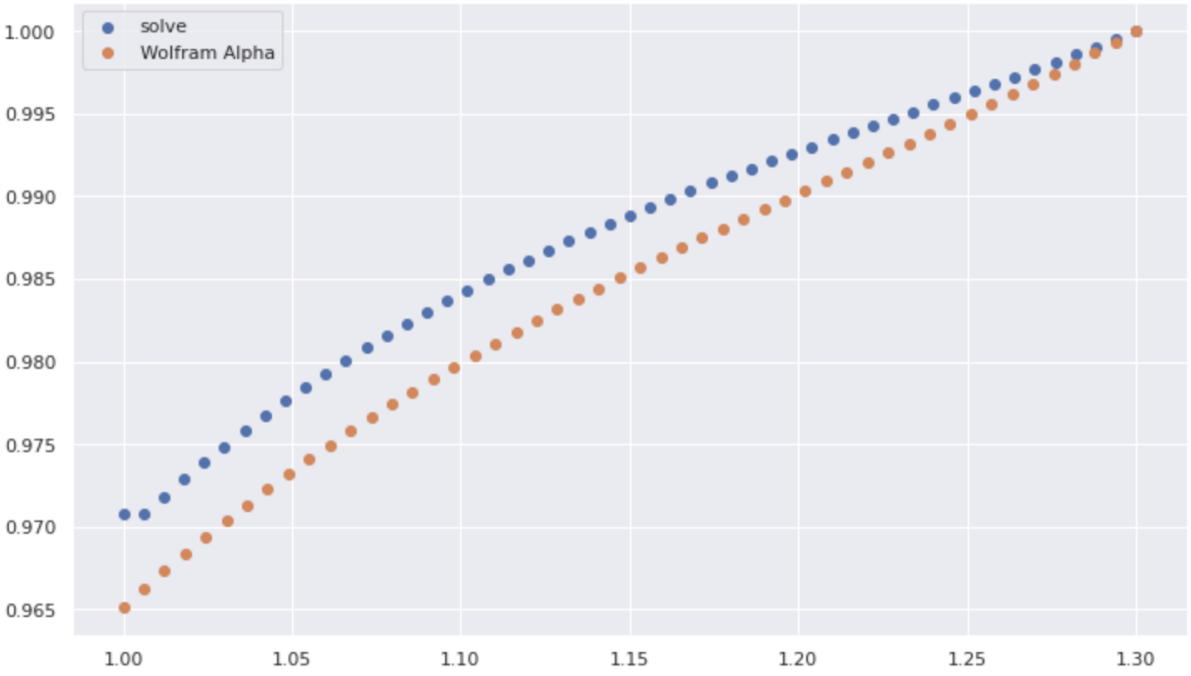
Для данного теста не удалось найти аналитическое решение.



4.0.4 Тест 3

$$y'' - \frac{y'}{2} + 3y = 2x^2 y(1) - 2y'(1) = 0.6y(1.3) = 1$$

Для данного теста также не удалось найти аналитическое решение, только приближение элементарными функциями.



Глава 5

Заключение

В данной работе был реализован метод Рунге-Кутты II и IV порядка точности для решения дифференциального уравнения первого порядка и системы дифференциальных уравнений первого порядка. По результатам работы программы, которая представлена в данной работе, продемонстрировано построение графика на основании результатов численных просчетов конечно-разностных уравнений метода Рунге-Кутты. Согласно графикам, точность IV порядка имеет меньшее отклонение от точного решения уравнения, чем график, при использовании метода II порядка точности. Таким образом, можно сделать вывод, что метод Рунге-Кутты IV порядка является более точным для решения дифференциальных уравнений первого порядка.
Также в данной работе был реализован способ решения краевой задачи методом конечных разностей, и полученная система конечно-разностных уравнений была решена методом прогонки.

Литература

[1] Самарский А.А. Введение в численные методы. СПб., 2009.