**Лабораторная работа №1**

**«Численные методы решения задачи Коши для ОДУ»**

Вариант 10

Выполнил студент 3 курса 2 группы ФПМИ

Сараев Владислав Максимович

Минск, 2020

**Постановка задачи**

Решить задачу Коши для обыкновенного дифференциального уравнения первого порядка на отрезке [0, 1] с шагом τ= 0.1 с помощью:

* Явного метода средних прямоугольников
* неявного метода Эйлера
* явного метода Рунге-Кутта 4-го порядка

Оценить погрешность численного решения с использованием правила Рунге и путем сравнения с точным решением .

Задача Коши:

**Краткие теоретические сведения**

1. **Неявный метод Эйлера**

Рассмотрим соотношение . Приблизим интеграл по правилу правых прямоугольников и получим неявное соотношение – неявный метод Эйлера. Для нахождения приближения в n+1 узле необходимо решить нелинейное уравнение, например, с помощью формулы Ньютона.

1. **Явный метод средних прямоугольников и явный метод Рунге-Кутта 4-ого порядка**

Семейство явных методов Рунге — Кутта является обобщением как явного метода Эйлера, так и классического метода Рунге — Кутта четвёртого порядка. Оно задаётся формулами:

, – величина шага сетки.

…

Коэффициенты определяются по матрице Бутчера.

Для **явного метода средних прямоугольников** имеем формулы:

Для **явного метода Рунге-Кутта четвертого порядка** имеем формулы:

1. **Правило Рунге**

**Листинг**

**import** numpy **as** np

**import** matplotlib**.**pyplot **as** plot

**from** math **import** tan

L **=** 0

R **=** 1

epsilon **=** 10 **\*\*** **(-**10**)**

u0 **=** 0

**def** u**(**t**):**

**return** tan**(**t**)** **-** t

**def** f**(**t**,** u\_t**):**

**return** **(**u\_t **+** t**)** **\*\*** 2

**def** der\_f**(**t**,** u\_t**):**

**return** 2 **\*** **(**u\_t **+** t**)**

**def** newton**(**y\_j**,** t\_j1**,** tau**):**

y\_k **=** y\_j **+** 2 **\*** epsilon

y\_k1 **=** y\_j

**while** **abs(**y\_k1 **-** y\_k**)** **>** epsilon**:**

y\_k **=** y\_k1

num **=** y\_k **-** y\_j **-** tau **\*** f**(**t\_j1**,** y\_k**)**

den **=** 1 **-** tau **\*** der\_f**(**t\_j1**,** y\_k**)**

y\_k1 **=** y\_k **-** num **/** den

**return** y\_k1

**def** explicit\_mean\_rect\_method**(**tau**,** **print):**

dots **=** np**.**linspace**(**L**,** R**,** **int((**R **-** L**)** **/** tau**)** **+** 1**)**

u\_values **=** **[**u**(**i**)** **for** i **in** dots**]**

y **=** np**.**zeros**(len(**dots**))**

y**[**0**]** **=** u0

**for** j **in** **range(**1**,** **len(**dots**)):**

k1 **=** f**(**dots**[**j **-** 1**],** y**[**j **-** 1**])**

k2 **=** f**(**dots**[**j **-** 1**]** **+** tau **/** 2**,** y**[**j **-** 1**]** **+** tau **\*** k1 **/** 2**)**

y**[**j**]** **=** y**[**j **-** 1**]** **+** tau **\*** k2

**if** **print:**

out**(**dots**,** y**,** u\_values**,** 1**)**

**return** y

**def** implicit\_Euler\_method**(**tau**,** **print):**

dots **=** np**.**linspace**(**L**,** R**,** **int((**R **-** L**)** **/** tau**)** **+** 1**)**

u\_values **=** **[**u**(**i**)** **for** i **in** dots**]**

y **=** np**.**zeros**(len(**dots**))**

y**[**0**]** **=** u0

**for** j **in** **range(**1**,** **len(**dots**)):**

y**[**j**]** **=** newton**(**y**[**j **-** 1**],** dots**[**j**],** tau**)**

**if** **print:**

out**(**dots**,** y**,** u\_values**,** 2**)**

**return** y

**def** explicit\_Runge\_Kutta\_method\_4\_order**(**tau**,** **print):**

dots **=** np**.**linspace**(**L**,** R**,** **int((**R **-** L**)** **/** tau**)** **+** 1**)**

u\_values **=** **[**u**(**i**)** **for** i **in** dots**]**

y **=** np**.**zeros**(len(**dots**))**

y**[**0**]** **=** u0

**for** j **in** **range(**1**,** **len(**dots**)):**

k1 **=** f**(**dots**[**j **-** 1**],** y**[**j **-** 1**])**

k2 **=** f**(**dots**[**j **-** 1**]** **+** tau **/** 2**,** y**[**j **-** 1**]** **+** tau **\*** k1 **/** 2**)**

k3 **=** f**(**dots**[**j **-** 1**]** **+** tau **/** 2**,** y**[**j **-** 1**]** **+** tau **\*** k2 **/** 2**)**

k4 **=** f**(**dots**[**j **-** 1**]** **+** tau**,** y**[**j **-** 1**]** **+** tau **\*** k3**)**

y**[**j**]** **=** y**[**j **-** 1**]** **+** tau **/** 6 **\*** **(**k1 **+** 2 **\*** k2 **+** 2 **\*** k3 **+** k4**)**

**if** **print:**

out**(**dots**,** y**,** u\_values**,** 3**)**

**return** y

**def** Runge\_fault**(**y\_h**,** y\_h\_2**,** p**):**

**return** **max([abs(**y\_h\_2**[**i**]** **-** y\_h**[**j**])**

**for** i**,** j **in** **zip(range(len(**y\_h\_2**)),** **range(**0**,** 2 **\*** **len(**y\_h\_2**),** 2**))])** **/** **(**2 **\*\*** p **-** 1**)**

**def** out**(**dots**,** y**,** u\_values**,** num**):**

plot**.**plot**(**dots**,** y**,** label**=**"Approximation " **+** **str(**num**))**

plot**.**plot**(**dots**,** u\_values**,** label**=**"Function"**)**

plot**.**legend**()**

plot**.**show**()**

**print(**"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"**)**

**print(**"Approximation " **+** **str(**num**)** **+** " values:"**)**

**print(**y**)**

delta **=** **[abs(**u\_values**[**i**]** **-** y**[**i**])** **for** i **in** **range(**0**,** **len(**y**))]**

**print(**"Max delta: " **+** **str(max(**delta**)))**

# Явный метод средних прямоугольников

y1\_h **=** explicit\_mean\_rect\_method**(**0.1**,** **True)**

y1\_h\_2 **=** explicit\_mean\_rect\_method**(**0.2**,** **False)**

**print(**"Runge fault 1: " **+** **str(**Runge\_fault**(**y1\_h**,** y1\_h\_2**,** 2**)))**

# Неявный метод Эйлера

y2\_h **=** implicit\_Euler\_method**(**0.01**,** **True)**

y2\_h\_2 **=** implicit\_Euler\_method**(**0.02**,** **False)**

**print(**"Runge fault 2: " **+** **str(**Runge\_fault**(**y2\_h**,** y2\_h\_2**,** 1**)))**

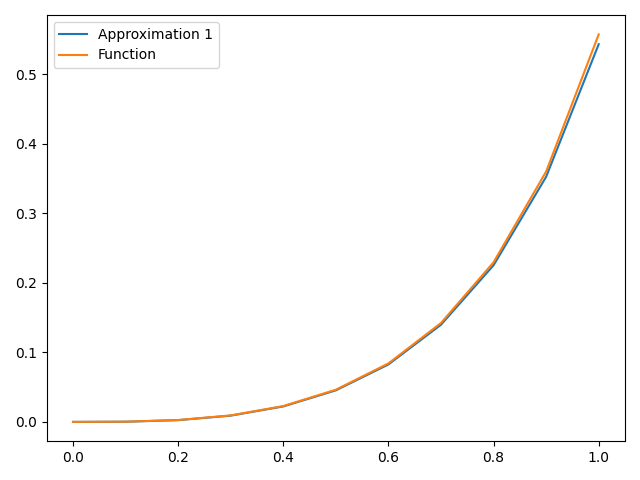
# Явный метод Рунге-Кутта 4-ого порядка

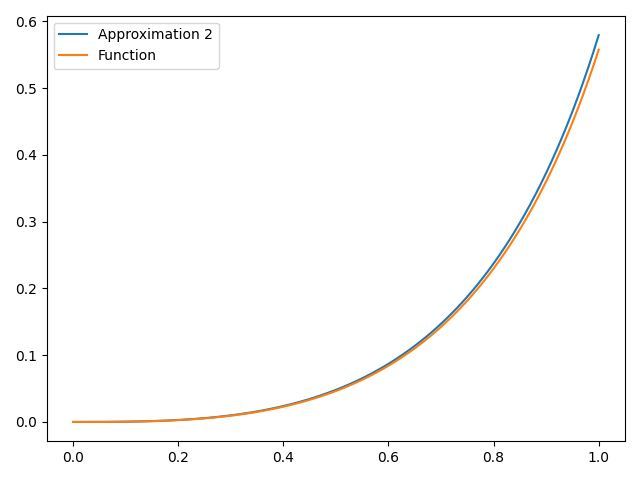
y3\_h **=** explicit\_Runge\_Kutta\_method\_4\_order**(**0.1**,** **True)**

y3\_h\_2 **=** explicit\_Runge\_Kutta\_method\_4\_order**(**0.2**,** **False)**

**print(**"Runge fault 3: " **+** **str(**Runge\_fault**(**y3\_h**,** y3\_h\_2**,** 4**)))**

**Результаты**

****

****

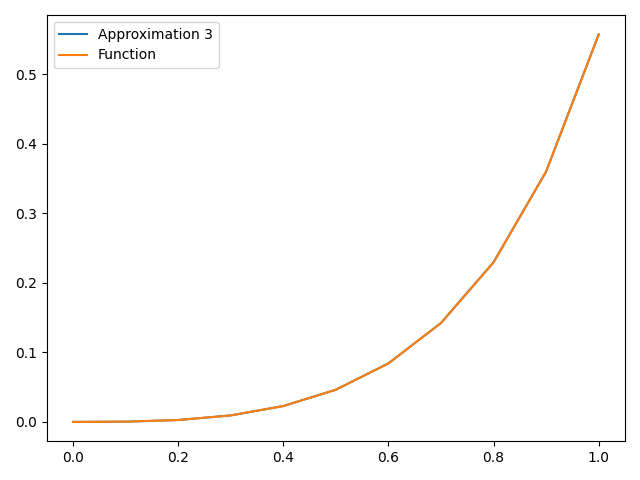
****

Таблица 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| j |  | Точное решение | Численное решение задачи Коши с шагом 0.1 | | |
| Метод 1 | Метод 2 | Метод 3 |
|  |  |  |  |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0.1 | 3.34672085e-04 | 2.50000000e-04 | 3.87015598e-04 | 3.34589078e-04 |
| 2 | 0.2 | 2.71003551e-03 | 2.52263172e-03 | 2.92372185e-03 | 2.70987823e-03 |
| 3 | 0.3 | 9.33624961e-03 | 9.00339345e-03 | 9.84455367e-03 | 9.33603934e-03 |
| 4 | 0.4 | 2.27932187e-02 | 2.22368039e-02 | 2.37762683e-02 | 2.27929929e-02 |
| 5 | 0.5 | 4.63024898e-02 | 4.53874324e-02 | 4.80226611e-02 | 4.63023076e-02 |
| 6 | 0.6 | 8.41368083e-02 | 8.26291353e-02 | 8.69994654e-02 | 8.41367567e-02 |
| 7 | 0.7 | 1.42288380e-01 | 1.39771889e-01 | 1.46953804e-01 | 1.42288569e-01 |
| 8 | 0.8 | 2.29638557e-01 | 2.25340444e-01 | 2.37243033e-01 | 2.29639061e-01 |
| 9 | 0.9 | 3.60158218e-01 | 3.52557773e-01 | 3.72786529e-01 | 3.60158783e-01 |
| 10 | 1 | 5.57407725e-01 | 5.43274653e-01 | 5.79206593e-01 | 5.57406443e-01 |

Таблица 2

|  |  |  |  |
| --- | --- | --- | --- |
|  | Метод 1 | Метод 2 | Метод 3 |
| Оценка погрешности по правилу Рунге | 0.010638632938512 | 0.023339168991571 | 3.650112490549e-06 |
|  | 0.014133072083173 | 0.021798868596310 | 1.2818099061062e-06 |

**Выводы**

Для решения данной задачи были применены методы с различной степенью точности. Наиболее точным оказался метод Рунге-Кутта 4-ого порядка