**Лабораторная работа №2**

**«Метод стрельбы решения граничной задачи для ОДУ»**

Вариант 10

Выполнил студент 3 курса 2 группы ФПМИ

Сараев Владислав Максимович

Минск, 2020

**Постановка задачи**

Найти численное решение граничной задачи методом стрельбы с шагом h = 0.01. Для численного решения задач Коши использовать явный метод средних прямоугольников. Оценить погрешность полученного численного решения с помощью правила Рунге. Сравнить найденное численное решение с точным решением . В одной системе координат построить график функции и график полученного численного решения.

Граничная задача:

**Краткие теоретические сведения**

Задание сводится к решению двух задач Коши:

Переходя от ДУ второго порядка к системе имеем:

Исходное решение будет иметь вид:

, где

Правило Рунге:

Задачи Коши решаются с помощью явного метода средних прямоугольников.

**Листинг**

# coding=utf-8

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

**from** matplotlib **import** pyplot **as** plot

# Искомая функция

**def** f**(**x**):**

**return** 1 **/** **(**x **+** 1**)**

# f1(x) из первой и второй задач Коши

**def** f1**(**t\_j**,** y1\_t\_j**,** y2\_t\_j**):**

**return** y2\_t\_j

# f2 из первой задачи Коши

**def** f12**(**t\_j**,** y1\_t\_j**,** y2\_t\_j**):**

**return** 3 **+** 3 **\*** **((**t\_j **+** 1**)** **\*\*** 2**)** **\*** y2\_t\_j **+** **(**2 **/** **((**t\_j **+** 1**)** **\*\*** 2**))** **\*** y1\_t\_j

# f2 из второй задачи Коши

**def** f22**(**t\_j**,** y1\_t\_j**,** y2\_t\_j**):**

**return** 3 **\*** **((**t\_j **+** 1**)** **\*\*** 2**)** **\*** y2\_t\_j **+** **(**2 **/** **((**t\_j **+** 1**)** **\*\*** 2**))** **\*** y1\_t\_j

# Явный метод cредних прямоугольников для системы из двух ДУ

**def** explicit\_mean\_rect\_method\_for\_system**(**dots**,** f\_1**,** f\_2**,** y1\_t0**,** y2\_t0**,** h**):**

y1 **=** np**.**empty**(len(**dots**))**

y2 **=** np**.**empty**(len(**dots**))**

y1**[**0**]** **=** y1\_t0

y2**[**0**]** **=** y2\_t0

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

k1 **=** f\_1**(**dots**[**j **-** 1**],** y1**[**j **-** 1**],** y2**[**j **-** 1**])**

q1 **=** f\_2**(**dots**[**j **-** 1**],** y1**[**j **-** 1**],** y2**[**j **-** 1**])**

k2 **=** f\_1**(**dots**[**j **-** 1**]** **+** h **/** 2**,** y1**[**j **-** 1**]** **+** k1 **\*** h **/** 2**,** y2**[**j **-** 1**]** **+** q1 **\*** h **/** 2**)**

q2 **=** f\_2**(**dots**[**j **-** 1**]** **+** h **/** 2**,** y1**[**j **-** 1**]** **+** k1 **\*** h **/** 2**,** y2**[**j **-** 1**]** **+** q1 **\*** h **/** 2**)**

y1**[**j**]** **=** y1**[**j **-** 1**]** **+** h **\*** k2

y2**[**j**]** **=** y2**[**j **-** 1**]** **+** h **\*** q2

**return** y1**,** y2

# Погрешность по правилу Рунге

**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**)**

L **=** 0

R **=** 1

dots\_h **=** np**.**linspace**(**L**,** R**,** **int((**R **-** L**)** **/** 0.01**)** **+** 1**)**

dots\_2h **=** np**.**linspace**(**L**,** R**,** **int((**R **-** L**)** **/** 0.02**)** **+** 1**)**

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

f\_2h\_values **=** **[**f**(**i**)** **for** i **in** dots\_2h**]**

**(**u0**,** der\_u0**)** **=** explicit\_mean\_rect\_method\_for\_system**(**dots\_h**,** f1**,** f12**,** 1**,** 0**,** 0.01**)**

**(**u1**,** der\_u1**)** **=** explicit\_mean\_rect\_method\_for\_system**(**dots\_h**,** f1**,** f22**,** 0**,** 1**,** 0.01**)**

C **=** **(**1 **-** u0**[-**1**]** **+** 2 **\*** der\_u0**[-**1**])** **/** **(**u1**[-**1**]** **-** 2 **\*** der\_u1**[-**1**])**

u **=** **[**u0**[**i**]** **+** C **\*** u1**[**i**]** **for** i **in** **range(len(**dots\_h**))]**

plot**.**plot**(**dots\_h**,** u**,** label**=**"Approximation"**)**

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

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

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

**(**u0\_2h**,** der\_u0\_2h**)** **=** explicit\_mean\_rect\_method\_for\_system**(**dots\_2h**,** f1**,** f12**,** 1**,** 0**,** 0.02**)**

**(**u1\_2h**,** der\_u1\_2h**)** **=** explicit\_mean\_rect\_method\_for\_system**(**dots\_2h**,** f1**,** f22**,** 0**,** 1**,** 0.02**)**

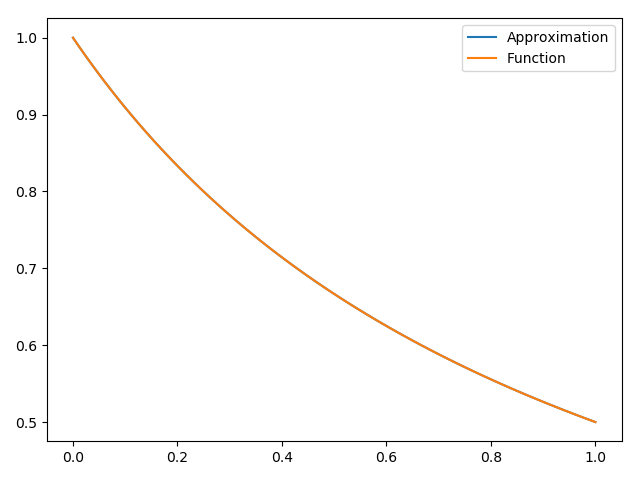
C\_2h **=** **(**1 **-** u0\_2h**[-**1**]** **+** 2 **\*** der\_u0\_2h**[-**1**])** **/** **(**u1\_2h**[-**1**]** **-** 2 **\*** der\_u1\_2h**[-**1**])**

u\_2h **=** **[**u0\_2h**[**i**]** **+** C\_2h **\*** u1\_2h**[**i**]** **for** i **in** **range(len(**dots\_2h**))]**

**print(**"Runge fault: " **+** **str(**Runge\_fault**(**u**,** u\_2h**,** 2**)))**

**print(**"Max fault: " **+** **str(max(**u**[**i**]** **-** f\_h\_values**[**i**]** **for** i **in** **range(len(**dots\_h**)))))**

**Результаты**

****

Погрешность по правилу Рунге: 1.8461715872793622e-05

: 1.8367104189564998e-05

**Выводы**

Если для решения получаемых задач Коши использовать численные методы с высокой точностью, то метод стрельбы позволяет получать решение граничной задачи для ОДУ также с высокой точностью.

Правило Рунге дает адекватную оценку.