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 $u(x) = \frac{1}{x+1}$. В одной системе координат построить график функции $u(x)$ и график полученного численного решения.

Граничная задача:

$$\begin{cases} u'' - 3(x+1)^2 u' - \dfrac{2}{(x+1)^2} u = 3 \\ u(0) = 1 \\ u(1) - 2u'(1) = 1 \end{cases}$$

Краткие теоретические сведения

Задание сводится к решению двух задач Коши:

$$u_0:\begin{cases} u'' - 3(x+1)^2u' - \dfrac{2}{(x+1)^2}u = 3 \\ u(0) = 1 \\ u'(0) = 0 \end{cases}$$

$$u_1:\begin{cases} u'' - 3(x+1)^2u' - \dfrac{2}{(x+1)^2}u = 0 \\ u(0) = 0 \\ u'(0) = 1 \end{cases}$$

Переходя от ДУ второго порядка к системе имеем:

$$u_0:\begin{cases} u_{11} = u \\ u_{12} = u' \\ u'_{12} - 3(x+1)^2u_{12} - \dfrac{2}{(x+1)^2}u_{11} = 3 \\ u_{11}(0) = 1 \\ u_{12(0)} = 0 \end{cases}$$

$$u_1:\begin{cases} u_{21} = u \\ u_{22} = u' \\ u'_{22} - 3(x+1)^2u_{22} - \dfrac{2}{(x+1)^2}u_{21} = 0 \\ u_{21}(0) = 0 \\ u_{22(0)} = 1 \end{cases}$$

Исходное решение будет иметь вид:

$$u(x) = u_0(x) + Cu_1(x)\text{ , где } C = \frac{\gamma_1 - \alpha_1 u_0(b) - \beta_1 u_0{\prime}(b)}{\alpha_1 u_1(b) + \beta_1 u_1{\prime}(b)} = \frac{1 - u_0(1) + 2u_0{\prime}(1)}{u_1(1) - 2u_1{\prime}(1)}$$

Правило Рунге:

$$\mathrm{R}_{\frac{\tau}{2}} = \frac{\max(|\mathrm{y}_{\mathrm{i},\frac{\tau}{2}} - \mathrm{y}_{\mathrm{i},\tau}|)}{2^{\mathrm{p}} - 1}$$

Задачи Коши решаются с помощью явного метода средних прямоугольников.

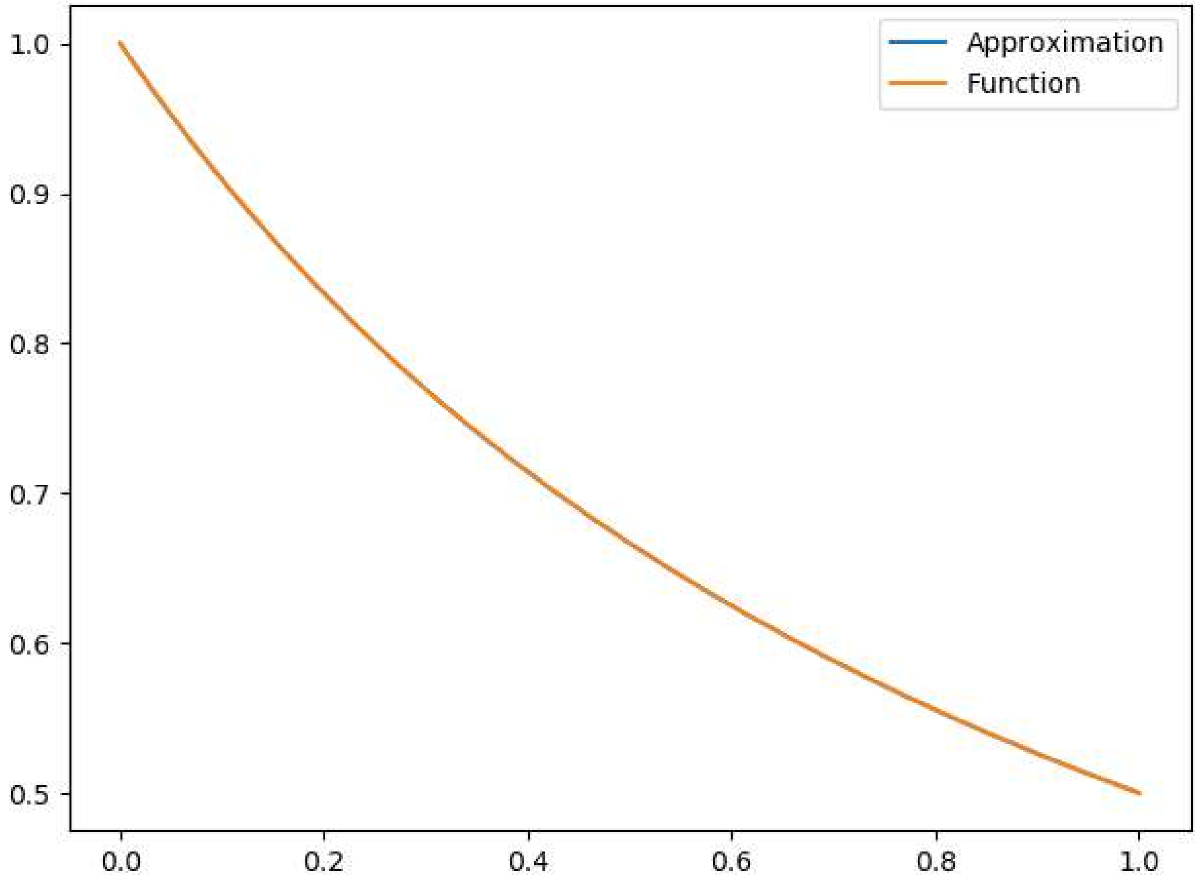
$$\begin{cases} k_1 = f_1(t_j, u_1(t_j), u_2(t_j)) \\ q_1 = f_2(t_j, u_1(t_j), u_2(t_j)) \\ k_2 = f_1(t_j + \dfrac{h}{2}, u_1(t_j) + \dfrac{h}{2}k_1, u_2(t_j) + \dfrac{h}{2}q_1) \\ q_2 = f_2(t_j + \dfrac{h}{2}, u_1(t_j) + \dfrac{h}{2}k_1, u_2(t_j) + \dfrac{h}{2}q_1) \\ u_1(t_{j+1}) = u_1(t_j) + hk_2 \\ u_2(t_{j+1}) = u_2(t_j) + hq_2 \end{cases}$$

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 средних 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

$\max\limits_{j=0,N}(|u(t_j) - y_j|)$: 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.