МИНОБРНАУКИ РОССИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ**

**ВЫСШЕГО ОБРАЗОВАНИЯ**

**“ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ”**

Факультет компьютерных наук

Кафедра цифровых технологий

Отчетполабораторной №3 ЧМ

Направление 02.03.01 Математика и компьютерные науки

Студент 3-его курса 7.2 группы – Гузенко Алексей Михайлович

Вариант 3

Воронеж 2020

**Содержание**

[**Постановка задачи** 3](#_Toc56174082)

[**Приведение краевой задачи к задаче Коши** 3](#_Toc56174083)

[**Схема вычисления значений для используемого метода решения задачи Коши** 3](#_Toc56174084)

[**Выбор первоначальных значений параметра стрельбы** 3](#_Toc56174085)

[**Исходный код решения (Python 3.8)** 4](#_Toc56174086)

[**График, отражающий результат каждой “стрельбы”** 5](#_Toc56174087)

[**Полученное приближенное решение в форме таблицы** 7](#_Toc56174088)

**Постановка задачи**

Реализовать метод стрельбы для приближенного решения краевой задачи

Соответствующую задачу Коши решить методом Адамса 4 порядка с шагом , а параметр вычислить методом деления отрезка пополам. Использовать точность решения

**Приведение краевой задачи к задаче Коши**

Обозначим

Тогда

Введем функцию ошибки , где – решение краевой задачи при параметре

**Схема вычисления значений для используемого метода решения задачи Коши**

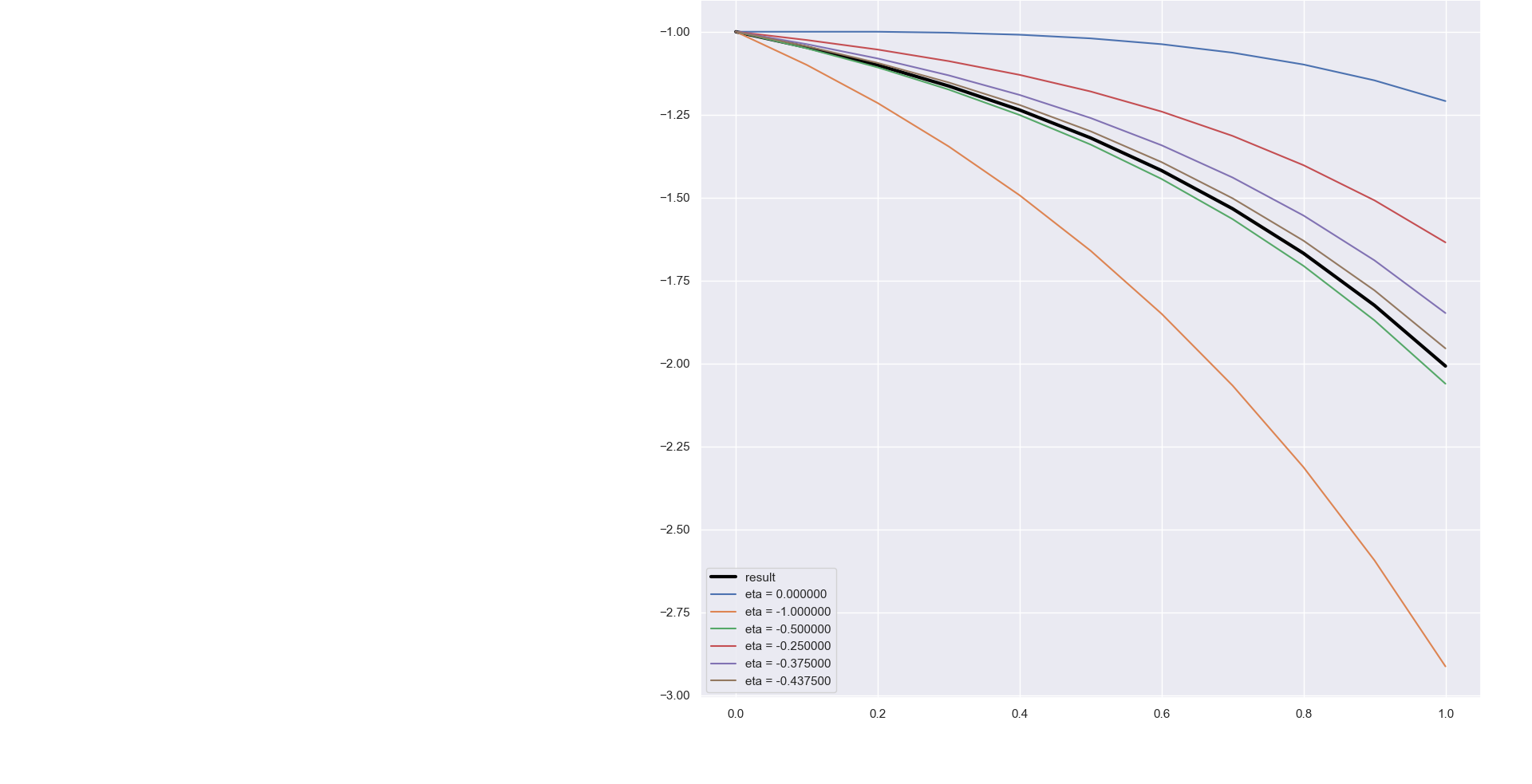
**Выбор первоначальных значений параметра стрельбы**

# **Исходный код решения (Python 3.8)**

import matplotlib.pyplot as plt  
import numpy as np  
import seaborn  
  
a, b = 0, 1  
A, B = -1, -2  
  
  
def f(x, u):  
 return u[1] - x  
  
def F(f):  
 return lambda x, u: np.append(u[1:], f(x, u))  
  
def adams4(f, a, b, u0, h):  
 func = F(f)  
 x = a  
 u = np.array(u0)  
 res = [(x, u[0])]  
 prev = [(x, u)]  
 while x + h <= b:  
 if len(prev) == 1:  
 u = u + h \* func(\*prev[-1])  
 elif len(prev) == 2:  
 u = u + h / 2 \* (3 \* func(\*prev[-1]) - func(\*prev[-2]))  
 elif len(prev) == 3:  
 u = u + h / 12 \* (23 \* func(\*prev[-1]) - 16 \* func(\*prev[-2]) + 5 \* func(\*prev[-3]))  
 else:  
 u = u + h / 24 \* (55 \* func(\*prev[-1]) - 59 \* func(\*prev[-2]) + 37 \* func(\*prev[-3]) - 9 \* func(\*prev[-4]))  
 x = x + h  
 res.append((x, u[0]))  
 prev.append((x, u))  
 return res  
  
  
def err(actual, res):  
 return actual - res[-1][1]  
  
eta\_1 = 0  
eta\_2 = (B - A) / (b - a)  
  
def solve(f, a, b, A, B, eta\_1, eta\_2, h=0.01, eps=0.01):  
 eta = (eta\_1 + eta\_2) / 2  
 hist = [  
 (eta\_1, adams4(f, a, b, [A, eta\_1], h)),  
 (eta\_2, adams4(f, a, b, [A, eta\_2], h))  
 ]  
 errors = {  
 eta\_1: err(B, hist[0][1]),  
 eta\_2: err(B, hist[1][1])  
 }  
 hist.append((eta, adams4(f, a, b, [A, eta], h)))  
 errors[eta] = err(B, hist[-1][1])  
 while (abs(errors[eta]) > eps):  
 if (errors[eta] \* errors[eta\_1] < 0):  
 eta\_1, eta\_2 = min(eta, eta\_1), max(eta, eta\_1)  
 else:  
 eta\_1, eta\_2 = min(eta, eta\_2), max(eta, eta\_2)  
 eta = (eta\_1 + eta\_2) / 2  
 hist.append((eta, adams4(f, a, b, [A, eta], h)))  
 errors[eta] = err(B, hist[-1][1])  
 return hist[-1], hist  
  
res, hist = solve(f, a, b, A, B, eta\_1, eta\_2, h=0.1, eps=0.01)  
arr = np.array(res[1])  
seaborn.set()  
plt.rcParams["figure.figsize"] = (20, 15)  
  
fig, ax = plt.subplots()  
  
arr = np.array(res[1])  
x = arr[:, 0]  
y = arr[:, 1]  
line = ax.plot(x, y, label='result', linewidth=3, color='black')  
  
for eta, r in hist[:-1]:  
 arr = np.array(r)  
 x = arr[:, 0]  
 y = arr[:, 1]  
 line = ax.plot(x, y, label='eta = {:f}'.format(eta))  
  
ax.legend()  
plt.show()

# 

# **График, отражающий результат каждой “стрельбы”**

****

# 

# **Полученное приближенное решение в форме таблицы**

|  |  |
| --- | --- |
|  |  |
| 0 | -1.20893811 |
| -1.0 | -2.91287621 |
| -0.5 | -2.06090716 |
| -0.25 | -1.63492263 |
| -0.375 | -1.8479149 |
| -0.4375 | -1.95441103 |
| -0.46875 | -2.0076590946865616 |