**Московский авиационный институт**

**(национальный исследовательский университет)**

Факультет: «Компьютерные науки и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

**Лабораторная работа №4**

по курсу «Численные методы»

Тема: «Численные методы решения обыкновенных дифференциальных уравнений»

Студент: Мариничев И. А.

Группа: М8О-308Б-19

Преподаватель: Пивоваров Д. Е.

Оценка:

Москва

2022

**1. Постановка задачи.**

**Вариант №12**.

4.1. Реализовать методы Эйлера, Рунге-Кутты и Адамса 4-го порядка в виде программ, задавая в качестве входных данных шаг сетки . С использованием разработанного программного обеспечения решить задачу Коши для ОДУ 2-го порядка на указанном отрезке. Оценить погрешность численного решения с использованием метода Рунге – Ромберга и путем сравнения с точным решением.

4.2. Реализовать метод стрельбы и конечно-разностный метод решения краевой задачи для ОДУ в виде программ. С использованием разработанного программного обеспечения решить краевую задачу для обыкновенного дифференциального уравнения 2-го порядка на указанном отрезке. Оценить погрешность численного решения с использованием метода Рунге – Ромберга и путем сравнения с точным решением.

**2. Описание методов.**

**4.1. Метод Эйлера, метод Рунге-Кутты 4-го порядка, метод Адамса 4-го порядка.**

Решаемая задача: Задача Коши для ОДУ 2-го порядка.

**4.2. Метод стрельбы, конечно-разностный метод.**

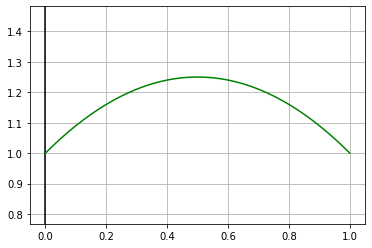
Решаемая задача: Краевая задача для ОДУ 2-го порядка.

**3. Демонстрация работы программы.**

**3.1. Задача Коши**

Посмотрим на график точного решения на отрезке [0, 1]:

import numpy as np  
import matplotlib.pyplot as plt  
  
def y\_true(x):  
 return x - x \*\* 2 + 1  
  
l = 0.0  
r = 1.0  
  
plt.axis('equal')  
plt.grid(True, which='both')  
plt.axvline(x=0, color='k')  
  
X = np.linspace(l, r, 100)  
plt.plot(X, y\_true(X), "-g")  
  
plt.show()



# Численное решение

Теперь запустите в терминале программу на **С++** при помощи команд

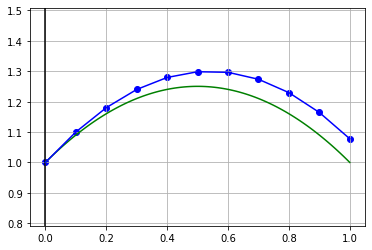
make  
make run

В появившемся файле answer\_NN.txt можно увидеть полученные численные решения. Проверим их, построив полученные точки на графике

def draw(l, r, x, y):  
 plt.axis('equal')  
 plt.grid(True, which='both')  
 plt.axvline(x=0, color='k')  
  
 X = np.linspace(l, r, 100)  
 plt.plot(X, y\_true(X), "-g")  
  
 plt.scatter(x, y, c ="blue")  
 plt.plot(x, y, "-b")  
   
 plt.show()

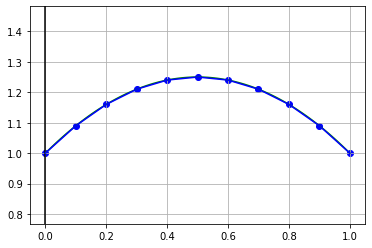
Метод Эйлера

X = [0.000000, 0.100000, 0.200000, 0.300000, 0.400000, 0.500000, 0.600000, 0.700000, 0.800000, 0.900000, 1.000000]  
Y = [1.000000, 1.100000, 1.180000, 1.239802, 1.279212, 1.298042, 1.296116, 1.273267, 1.229341, 1.164197, 1.077706]  
draw(l, r, X, Y)



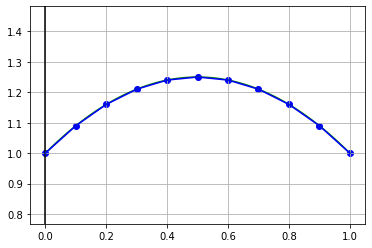
Метод Рунге-Кутты

X = [0.000000, 0.100000, 0.200000, 0.300000, 0.400000, 0.500000, 0.600000, 0.700000, 0.800000, 0.900000, 1.000000]  
Y = [1.000000, 1.090000, 1.160000, 1.210000, 1.240000, 1.250001, 1.240001, 1.210002, 1.160002, 1.090003, 1.000004]  
draw(l, r, X, Y)



Метод Адамса

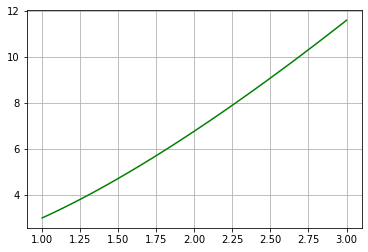
X = [0.000000, 0.100000, 0.200000, 0.300000, 0.400000, 0.500000, 0.600000, 0.700000, 0.800000, 0.900000, 1.000000]  
Y = [1.000000, 1.090000, 1.160000, 1.210000, 1.240001, 1.250001, 1.240002, 1.210002, 1.160003, 1.090003, 1.000004]  
draw(l, r, X, Y)



**3.2. Краевая задача**

Посмотрим на график точного решения на отрезке [1, 3]:

import numpy as np  
import matplotlib.pyplot as plt  
  
def y\_true(x):  
 return 2 + x + 2 \* x \* np.log(np.abs(x))  
  
a = 1.0  
b = 3.0  
  
plt.grid(True, which='both')  
X = np.linspace(a, b, 100)  
plt.plot(X, y\_true(X), "-g")  
plt.show()



# Численное решение

Теперь запустите в терминале программу на **С++** при помощи команд

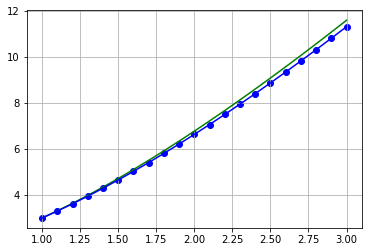
make  
make run

В появившемся файле answer\_NN.txt можно увидеть полученные численные решения. Проверим их, построив полученные точки на графике

def draw(l, r, x, y):  
 plt.grid(True, which='both')  
  
 X = np.linspace(l, r, 100)  
 plt.plot(X, y\_true(X), "-g")  
  
 plt.scatter(x, y, c ="blue")  
 plt.plot(x, y, "-b")  
   
 plt.show()

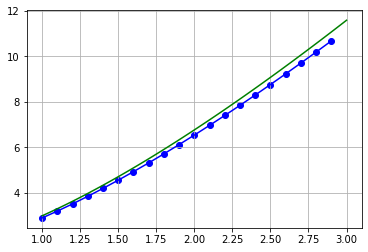
Метод стрельбы

X = [1.000000, 1.100000, 1.200000, 1.300000, 1.400000, 1.500000, 1.600000, 1.700000, 1.800000, 1.900000, 2.000000, 2.100000, 2.200000, 2.300000, 2.400000, 2.500000, 2.600000, 2.700000, 2.800000, 2.900000, 3.000000]  
Y = [3.000000, 3.300000, 3.615308, 3.945925, 4.291212, 4.650418, 5.022795, 5.407643, 5.804312, 6.212204, 6.630776, 7.059527, 7.497999, 7.945772, 8.402458, 8.867698, 9.341163, 9.822545, 10.311559, 10.807938, 11.311435]  
  
draw(a, b, X, Y)



Метод конечных разностей

X = [1.000000, 1.100000, 1.200000, 1.300000, 1.400000, 1.500000, 1.600000, 1.700000, 1.800000, 1.900000, 2.000000, 2.100000, 2.200000, 2.300000, 2.400000, 2.500000, 2.600000, 2.700000, 2.800000, 2.900000]  
Y = [2.905120, 3.205120, 3.522371, 3.855435, 4.203096, 4.564312, 4.938179, 5.323905, 5.720794, 6.128226, 6.545644, 6.972551, 7.408494, 7.853063, 8.305882, 8.766608, 9.234924, 9.710538, 10.193181, 10.682601]  
  
draw(a, b, X, Y)



**4. Выводы.**

В ходе данной лабораторной работы я изучил базовые численные методы, решающие обыкновенные дифференциальные уравнения.

Стоит отметить, что те варианты реализаций, которые были написаны мной на С++ носят скорее учебный характер, так как передо мной стояла задача понять алгоритмы при реализации, а не написать максимально оптимальные решатели тех или иных задач.