**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра алгоритмической математики**

**КУРСОВАЯ РАБОТА**

**по дисциплине «Дифференциальные уравнения»**

**Тема: Соскальзывание цепочки**

|  |  |  |
| --- | --- | --- |
| Студенты гр. 8382 |  | Кобенко В.П.  Черницын П.А. |
| Преподаватель |  | Павлов Д.А. |

Санкт-Петербург

2021

**ЗАДАНИЕ**

**НА КУРСОВУЮ РАБОТУ**

|  |  |  |
| --- | --- | --- |
| Студент Кобенко В.П.  Студент Черницын П.А. | | |
| Группа 8382 | | |
| Тема работы: Соскальзывание цепочки | | |
| Исходные данные:  Соскальзывание цепочки | | |
| Содержание пояснительной записки:   «Содержание», «Введение», «2-ой Закон Ньютона», «Прямой метод Ньютона», «Обратный метод Ньютона», «Метод Рунге-Кутты-Фельберга 4-5-го порядка», «Метод Хойна», «Метод Адамса», «Графический интерфейс», «Заключение», «Список использованных источников». | | |
| Предполагаемый объем пояснительной записки:  Не менее 20 страниц. | | |
| Дата выдачи задания: 09.05.2021 | | |
| Дата сдачи курсовой работы: 14.06.2021 | | |
| Дата защиты курсовой работы: 14.06.2021 | | |
| Студенты |  | Кобенко В.П.  Черницын П.А. |
| Преподаватель |  | Павлов Д.А. |

**АННОТАЦИЯ**

В курсовой работе рассмотрена задача соскальзывание цепочки. Для этого использовался 2-ой Закон Ньютона, его дифференциальная формулировка. Для решения поставленной задачи было использовано несколько методов: «Прямой метод Ньютона», «Обратный метод Ньютона», «Метод Рунге-Кутты-Фельберга 4-5-го порядка», «Метод Хойна», «Метод Адамса». Результаты решения данного уравнения были представлены в виде графиков в графическом интерфейсе.

**SUMMARY**

In the course work, the problem of chain slip. For this, the 2nd Newton law, its differential formulation, was used. To solve the problem, several methods were used: "Forward Newton's method", "Backward Newton's method", "Runge-Kutta-Felberg method of the 4-5th order", "Heun's method", "Adams method". The results of solving this equation were presented in the form of graphs in the graphical interface.

**СОДЕРЖАНИЕ**

[**ЗАДАНИЕ** 2](#_Toc75635569)

[**НА КУРСОВУЮ РАБОТУ** 2](#_Toc75635570)

[**АННОТАЦИЯ** 3](#_Toc75635571)

[**Введение** 5](#_Toc75635572)

[**2-ой Закон Ньютона** 6](#_Toc75635573)

[**Прямой метод Эйлера** 8](#_Toc75635574)

[**Обратный метод Эйлера** 10](#_Toc75635575)

[**Метод Рунге-Кутты-Фельберга 4-5 порядка** 12](#_Toc75635576)

[**Метод Хойна** 13](#_Toc75635577)

[**Метод Адамса-Башфорта** 14](#_Toc75635578)

[**GUI** 16](#_Toc75635579)

[**Вывод** 19](#_Toc75635580)

[**Используемая литература** 20](#_Toc75635581)

[**ПРИЛОЖЕНИЕ А. Код программы** 21](#_Toc75635582)

**Введение**

Дифференциальное уравнение является одним из фундаментальных понятий математики, широко применяемое в различных областях современных наук. Оно также применимо в физических процессах, один из которых рассматривается в данной курсовой работе. Соскальзывание цепочки является этим процессом. Были использованы методы интегрирования дифференциальных уравнений динамических систем для решения 2-ого закона Ньютона, такие как: «Прямой метод Ньютона», «Обратный метод Ньютона», «Метод Рунге-Кутты-Фельберга 4-5-го порядка», «Метод Хойна», «Метод Адамса».

**2-ой Закон Ньютона**

2-ой Закон Ньютона устанавливает связь между силой **F**, действующей на тело массы **m**, и ускорением **a**, которое приобретает тело под действием этой силы. Дифференциальная формулировка выглядит так:

Где **F** – сила, **t** – время, **p** – импульс.

В предположении, что движение одномерное, второй закон Ньютона в этом случае записывается в виде дифференциального уравнения второго порядка:

Согласно второму закону Ньютона, дифференциальное уравнение движения цепочки имеет вид:

Отсюда:

Поделим обе части уравнения на m:

Получаем:

Проинтегрируем и получим функцию f(x,t), использующуюся в численных методах.

Где - коэффициент трения, L – длина цепочки, g – ускорение свободного падения.

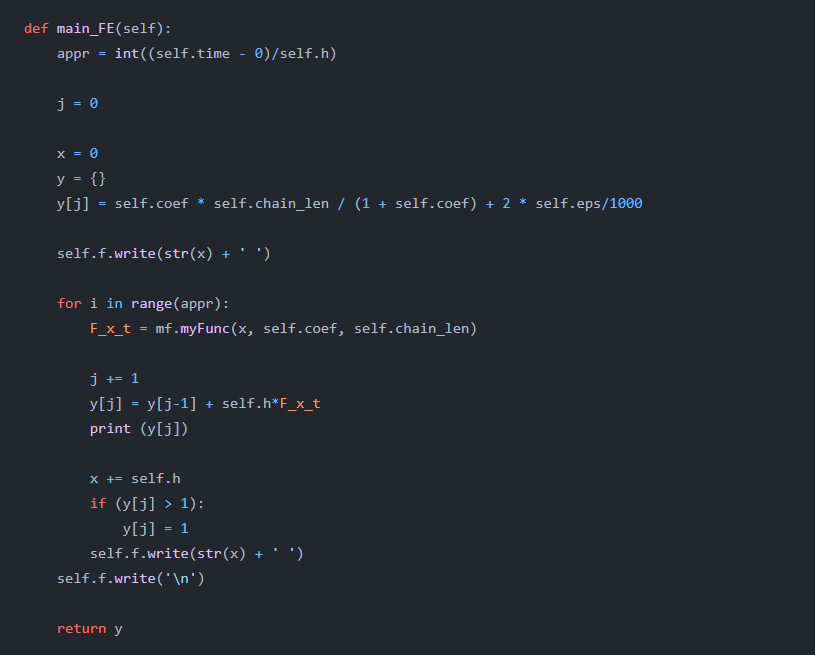
Для получения аналитической формулы должно быть известно начальное условие:

Длина свисающей части цепочки при равновесии составляет:

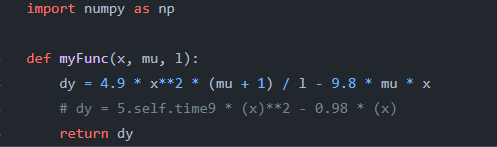
Скольжение цепочки описывается законом:

**Прямой метод Эйлера**

В нашем коде этот метод был реализован так:

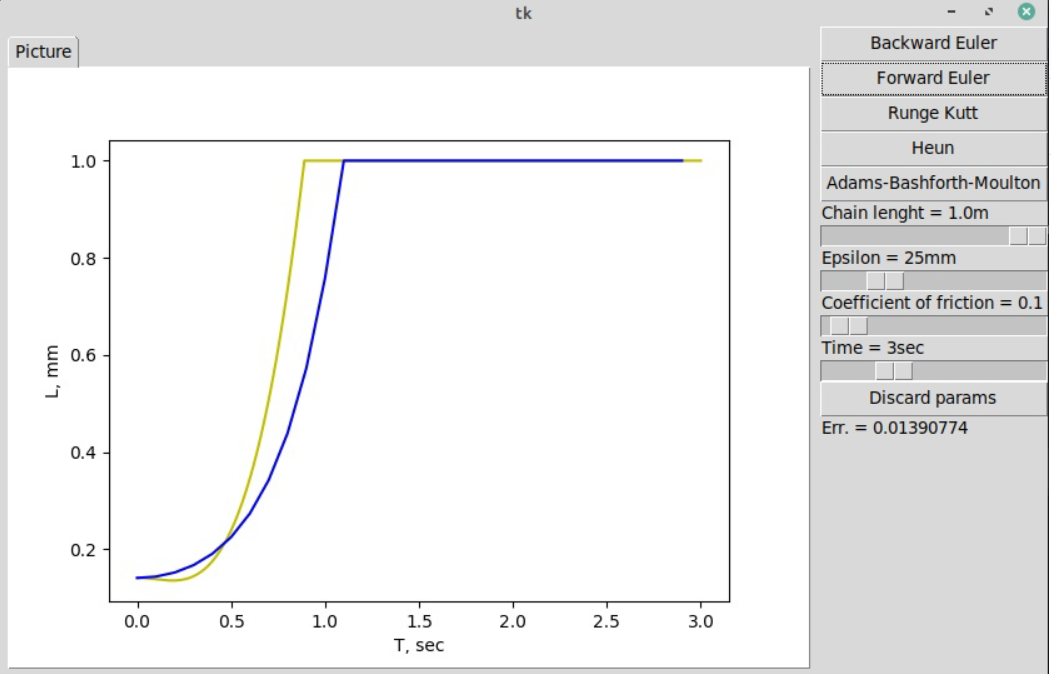


F\_x\_t – функция f(x, t), self.h – это длина шага по x, my\_func.py выглядит так:

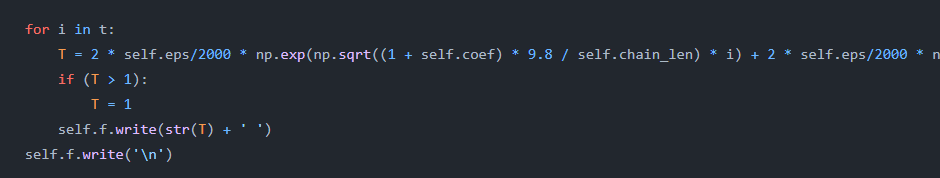
,

Где mu – это коэффициент трения, а 9.8 – ускорение свободного падения.

Для сравнения с аналитическим решением использовался график, желтая линия – график метода, синяя – аналитическое решение.



Реализация аналитического решения выглядит так (для всех методов она одинаковая):



Также после выполнения метода, на экран выводится его глобальная ошибка для шага h = 0.01.

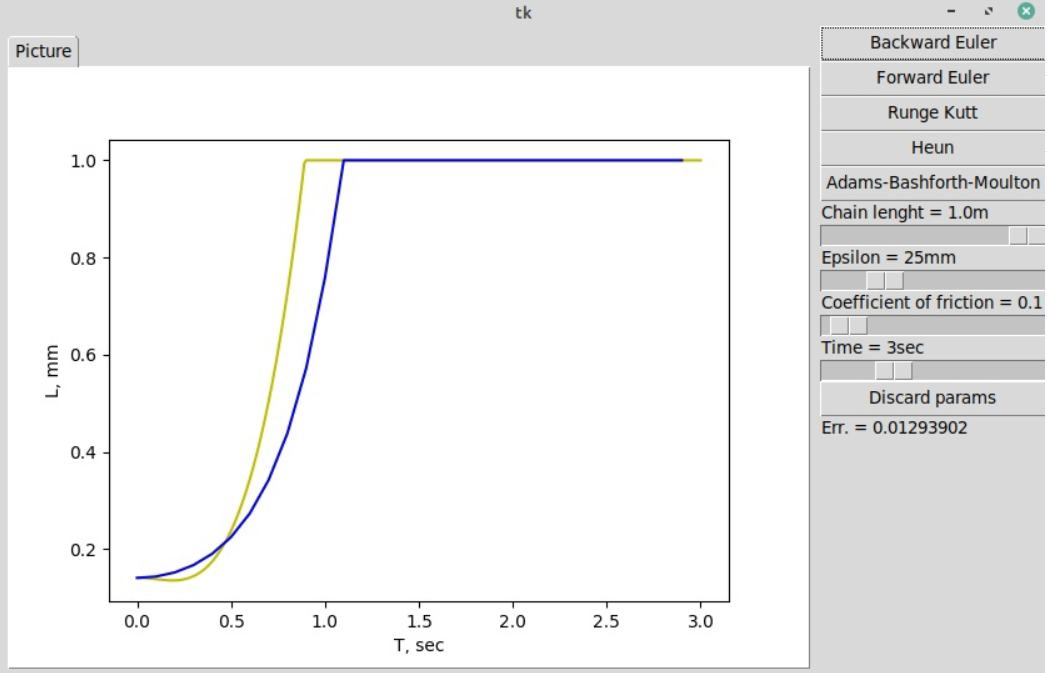
**Обратный метод Эйлера**

В нашем коде этот метод был реализован так:



Где F\_x\_t – функция f(x, t), self.h – это длина шага по x.

Для сравнения с аналитическим решением использовался график, желтая линия – график метода, синяя – аналитическое решение.



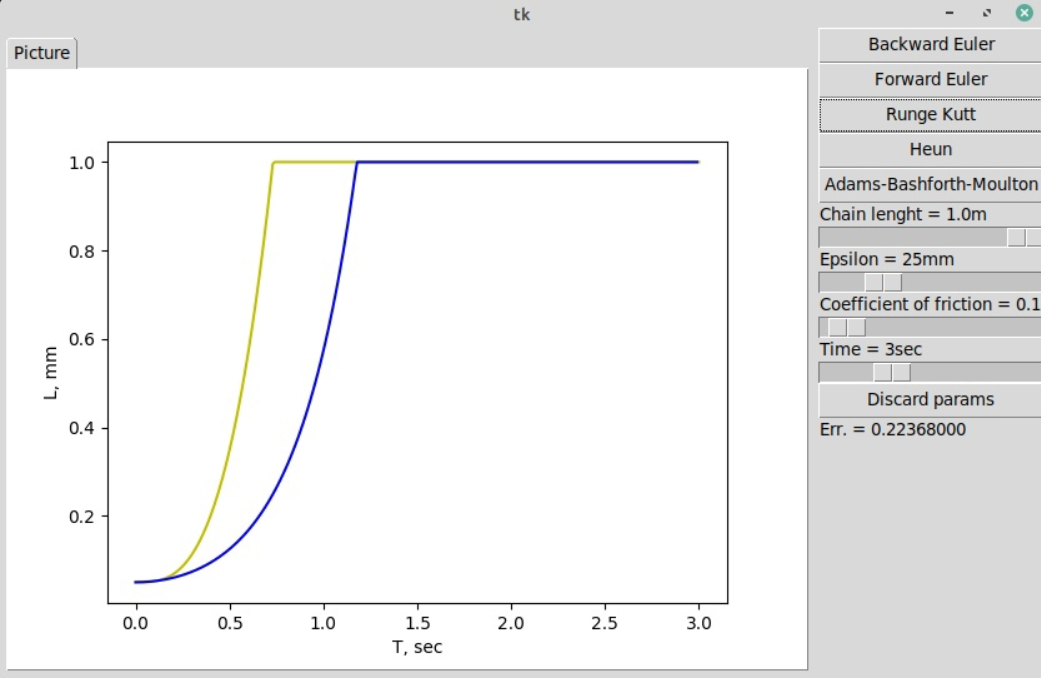
Также после выполнения метода, на экран выводится его глобальная ошибка для шага h = 0.01.

**Метод Рунге-Кутты-Фельберга 4-5 порядка**

Этот метод был реализован таким образом:



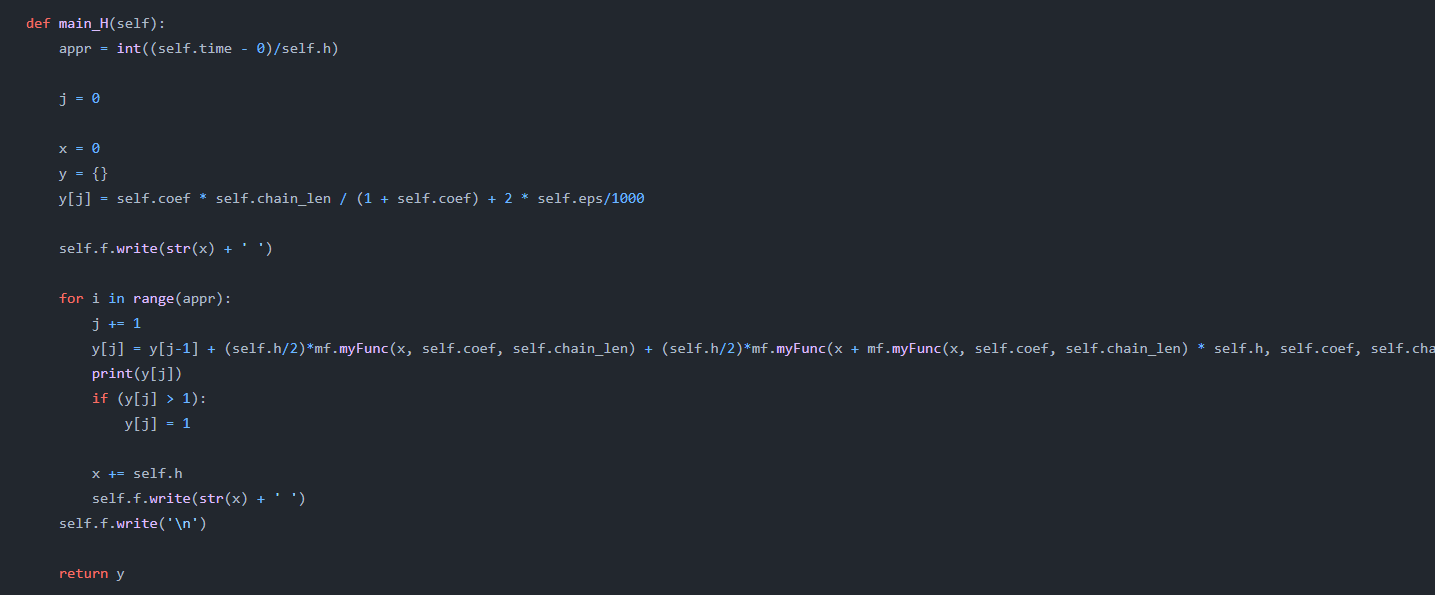
Для сравнения с аналитическим решением использовался график, желтая линия – график метода, синяя – аналитическое решение.



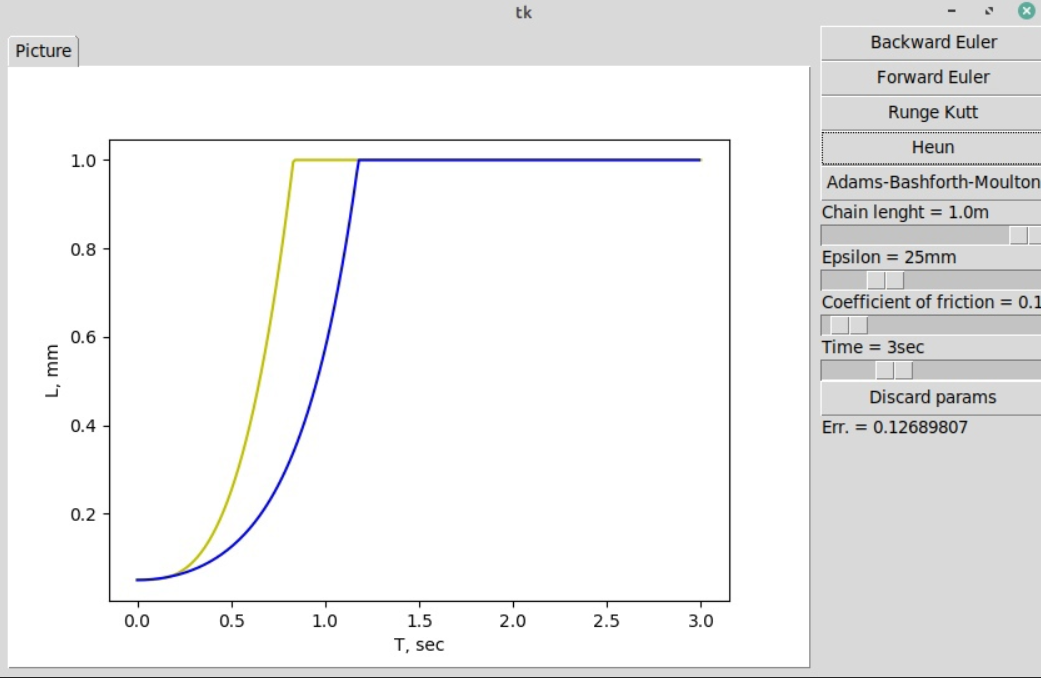
Также после выполнения метода, на экран выводится его глобальная ошибка для шага h = 0.01.

**Метод Хойна**

В работе метод был реализован так:



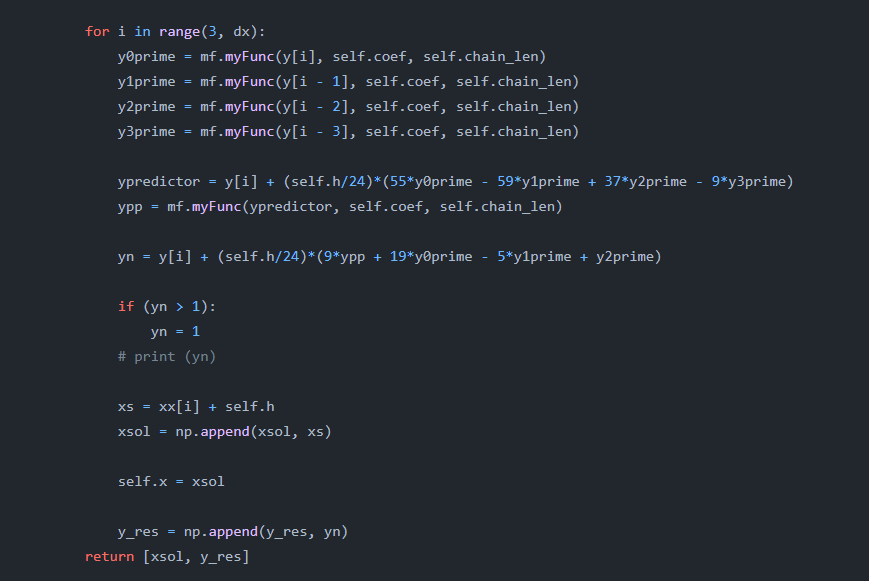
Для сравнения с аналитическим решением использовался график, желтая линия – график метода, синяя – аналитическое решение.



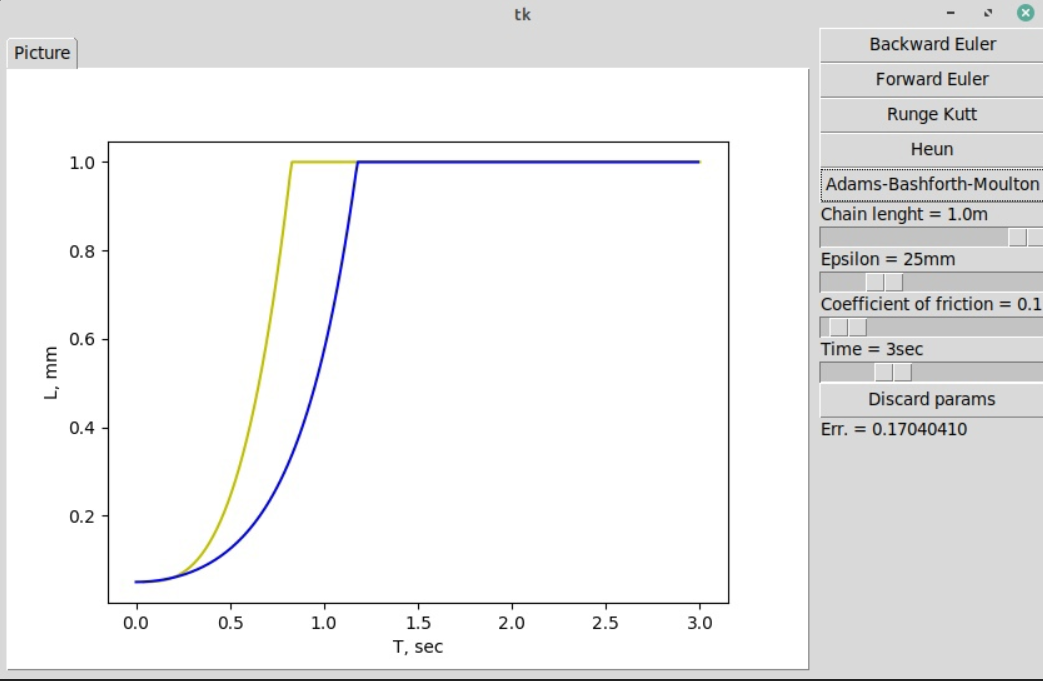
Также после выполнения метода, на экран выводится его глобальная ошибка для шага h = 0.01.

**Метод Адамса-Башфорта**

В работе метод был реализован так:



Для сравнения с аналитическим решением использовался график, желтая линия – график метода, синяя – аналитическое решение.



Также после выполнения метода, на экран выводится его глобальная ошибка для шага h = 0.01.

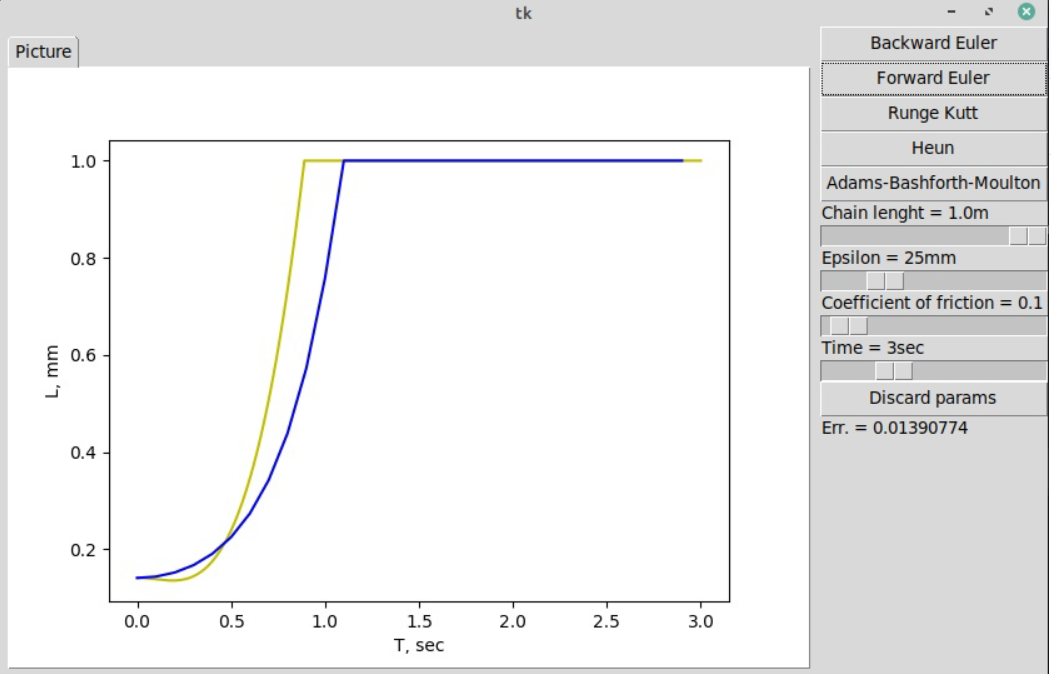
**GUI**

Был реализован графический интерфейс на языке Python с помощью библиотеки PySimpleGUI. Код программы представлен в приложении А.

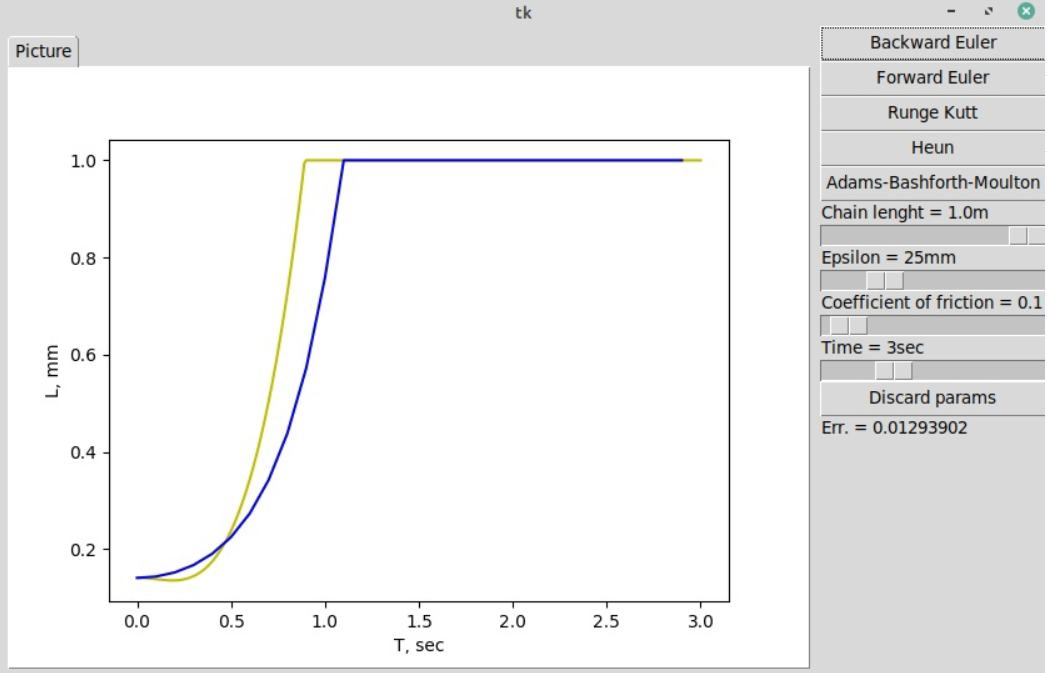
Интерфейс программы включает в себя 6 кнопок, 3 слайдера, окно для графиков и лэйбл с выводом информации об успешном/неуспешном запуске программы:

Первые пять кнопок в интерфейсе вызывают численные методы:

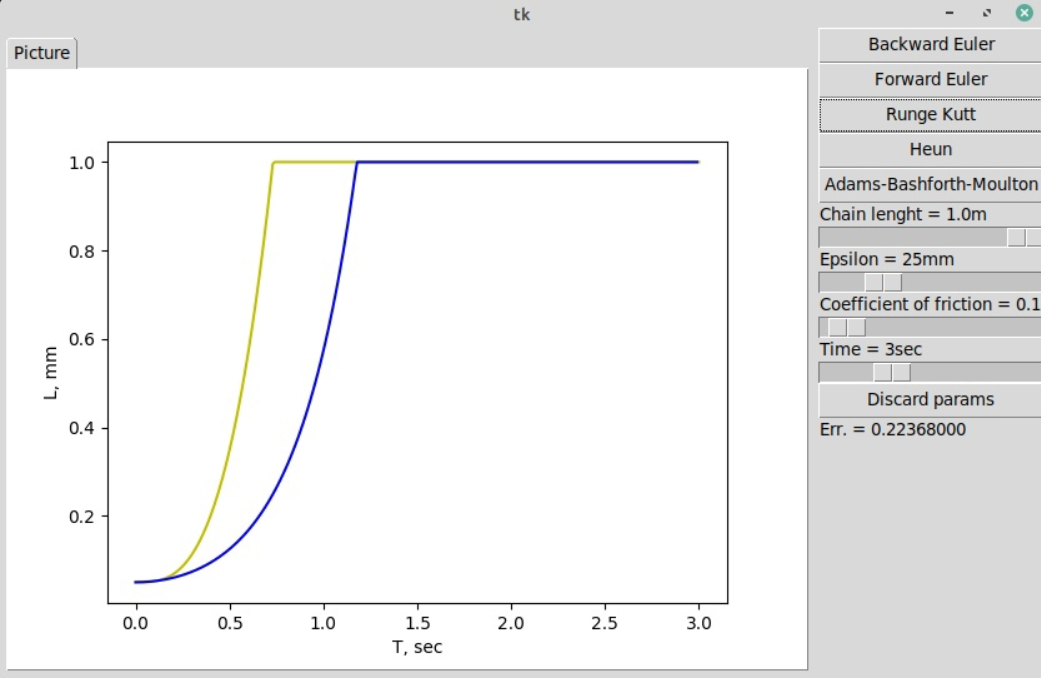
Forward Euler:



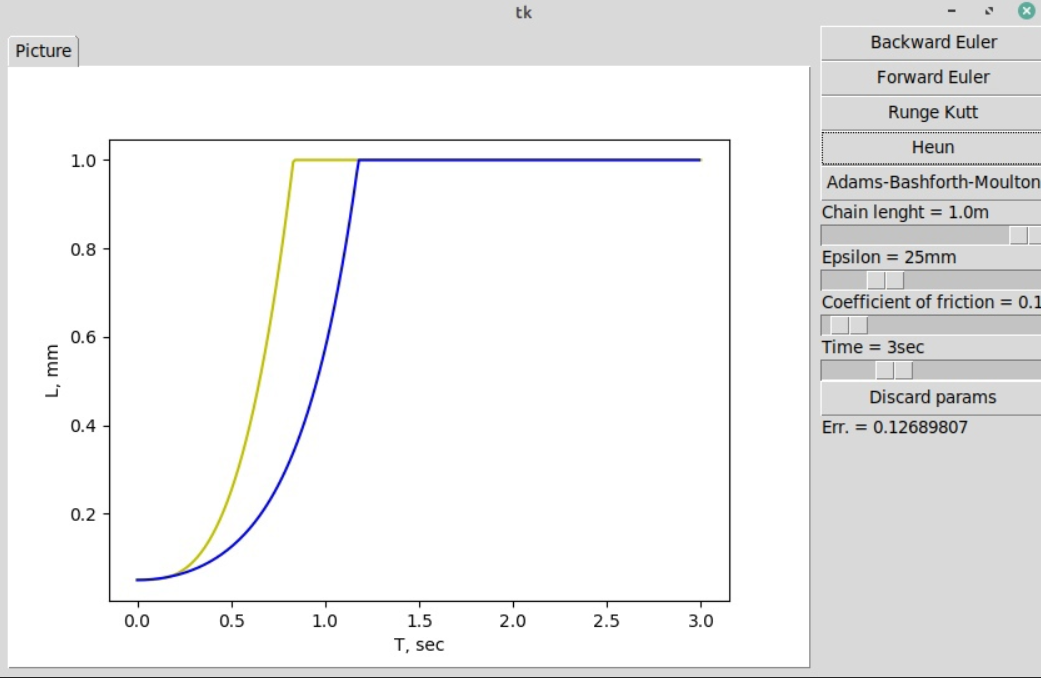
Backward Euler:



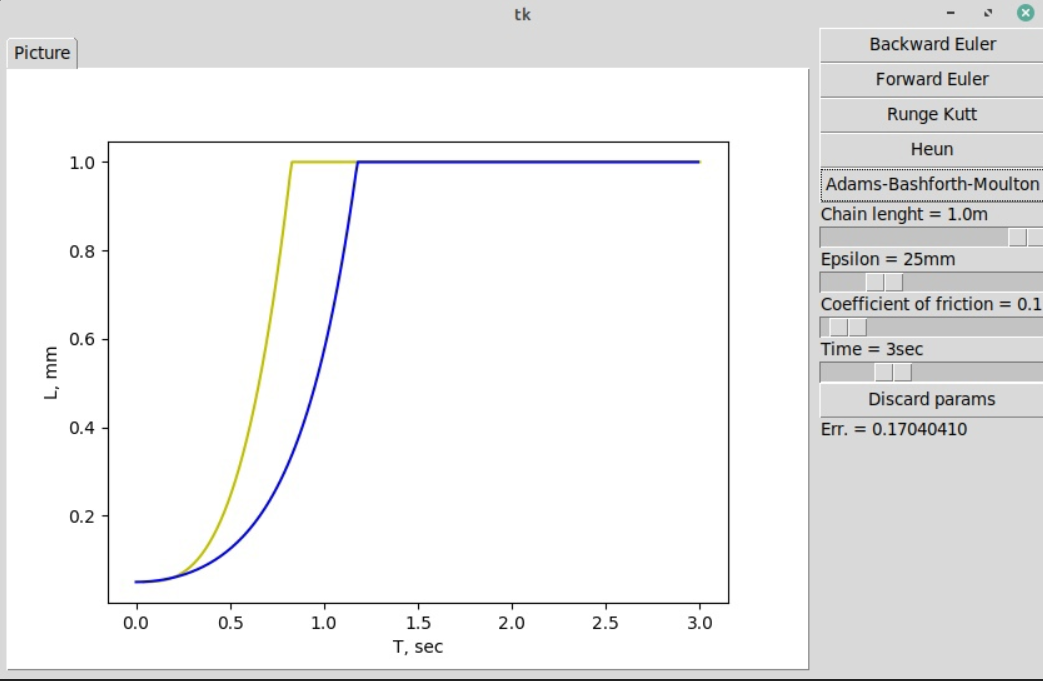
Runge Kutt:



Heun:



Adams-Bashfourth:



С помощью слайдеров можно менять условия поставленной задачи.

Chain length отвечает за длину цепочки, Epsilon за смещение цепочки относительно точки равновесия, Time – за время, кнопка Discard params – сбрасывает значения на исходные.

Также под графиком выводится глобальная ошибка для конкретного метода с шагом h = 0.001

**Вывод**

В курсовой работе был рассмотрен процесс соскальзывания цепочки. Для решения задачи были использованы такие методы, как: «Прямой метод Ньютона», «Обратный метод Ньютона», «Метод Рунге-Кутты-Фельберга 4-5-го порядка», «Метод Хойна», «Метод Адамса». Было написано приложение на языке Python, решающее поставленную задачу данными методами и сравнивающее решение с аналитическим. Графический интерфейс позволяет увидеть отличия между методами на графиках.

**Используемая литература**

<https://old.math.tsu.ru/EEResources/pdf/diff_equation.pdf>

<http://math.smith.edu/~callahan/cic/ch4.pdf>

<https://en.wikipedia.org/wiki/Newton%27s_law_of_cooling>

<https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%D0%90%D0%B4%D0%B0%D0%BC%D1%81%D0%B0>

<http://w.ict.nsc.ru/books/textbooks/akhmerov/nm-ode_unicode/1-3.html>

https://tftwiki.ru/wiki/Heun%27s\_method

<https://pysimplegui.readthedocs.io/en/latest/>

<https://www.python.org/>

**ПРИЛОЖЕНИЕ А. Код программы**

Файл main.py:

import tkinter as tk

import gui

root = tk.Tk()

gui.MainApplication(root)

root.mainloop()

Файл gui.py:

from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg

from matplotlib.figure import Figure

from tkinter import ttk

import tkinter as tk

import numpy as np

import backward\_euler

import forward\_euler

import runge\_kutta

import heun

import adams\_bashforth\_moulton

import os

class Plotter(FigureCanvasTkAgg):

def \_\_init\_\_(self, master):

self.figure = Figure(dpi=100)

super().\_\_init\_\_(self.figure, master=master)

self.axes = self.figure.add\_subplot(111)

self.get\_tk\_widget().grid(column=0, row=0, sticky='nsew')

def draw\_lists(self, flag):

self.axes.clear()

x = [[],[],[],[]]

i = 0

f = open('tmp.txt', 'r')

for line in f:

if (line == '\n'):

continue

for elem in line.split(' '):

if (elem == '\n'):

continue

x[i].append(float(elem))

i+=1

f.close()

self.axes.plot(x[0], x[1], color='y')

self.axes.plot(x[2], x[3], color='b')

self.axes.set\_xlabel('T, sec')

self.axes.set\_ylabel('L, mm')

self.draw\_idle()

class MainApplication(ttk.Frame):

def \_\_init\_\_(self, master, \*args, \*\*kwargs):

super().\_\_init\_\_(master)

self.grid(column=0, row=0, sticky='nsew')

frame = ttk.Frame(self, borderwidth=8)

frame.grid(column=0, row=0, sticky='nsew')

frame.rowconfigure(0, weight=1)

notes = ttk.Notebook(frame)

notes.grid(column=0, row=0, sticky='nsew')

notes.rowconfigure(0, weight=1)

page = ttk.Frame(notes)

notes.add(page, text='Picture')

self.plot = Plotter(page)

input\_frame = ttk.Frame(self)

input\_frame.grid(column=1, row=0, sticky='nsew')

label\_chain\_lenght = ttk.Label(input\_frame)

self.slider\_chain\_lenght = ttk.Scale(input\_frame, from\_ = 0, to\_ = 5,

command=lambda x:

label\_chain\_lenght.config(text = "Chain lenght = " +

str("{0:.1f}".format(self.slider\_chain\_lenght.get())) + "m"))

self.slider\_chain\_lenght.set(1)

label\_chain\_lenght.config(text = "Chain lenght = " + str("{0:.1f}".format(self.slider\_chain\_lenght.get())) + "m")

label\_epsilon = ttk.Label(input\_frame)

self.slider\_epsilon = ttk.Scale(input\_frame, from\_ = 1, to\_ = 500,

command=lambda x:

label\_epsilon.config(text = "Epsilon = " + str(int(self.slider\_epsilon.get())) + "mm"))

self.slider\_epsilon.set(25)

label\_epsilon.config(text = "Epsilon = " + str(int(self.slider\_epsilon.get())) + "mm")

label\_coef = ttk.Label(input\_frame)

self.slider\_coef = ttk.Scale(input\_frame, from\_ = 0, to\_ = 2,

command=lambda x:

label\_coef.config(text = "Coefficient of friction = " +

str("{0:.1f}".format(self.slider\_coef.get()))))

self.slider\_coef.set(0.1)

label\_coef.config(text = "Coefficient of friction = " + str("{0:.1f}".format(self.slider\_coef.get())))

label\_time = ttk.Label(input\_frame)

self.slider\_time = ttk.Scale(input\_frame, from\_ = 0.1, to\_ = 10,

command=lambda x:

label\_time.config(text = "Time = " +

str("{0:.1f}".format(self.slider\_time.get())) + "sec"))

self.slider\_time.set(3)

label\_time.config(text = "Time = " + str(int(self.slider\_time.get())) + "sec")

self.label\_err\_msg = ttk.Label(input\_frame)

self.label\_err\_msg.config(text = "")

button\_BE = ttk.Button(input\_frame, text='Backward Euler', command = self.button\_BE\_clicked)

button\_FE = ttk.Button(input\_frame, text='Forward Euler', command = self.button\_FE\_clicked)

button\_RK = ttk.Button(input\_frame, text='Runge Kutt', command = self.button\_RK\_clicked)

button\_H = ttk.Button(input\_frame, text='Heun', command = self.button\_H\_clicked)

button\_ADM = ttk.Button(input\_frame, text='Adams-Bashforth-Moulton', command = self.button\_ADM\_clicked)

button\_discard\_param = ttk.Button(input\_frame, text='Discard params', command = self.button\_discard\_param\_clicked)

button\_BE.grid(column=0, row=0, columnspan=2, sticky='ew')

button\_FE.grid(column=0, row=1, columnspan=2, sticky='ew')

button\_RK.grid(column=0, row=2, columnspan=2, sticky='ew')

button\_H.grid(column=0, row=3, columnspan=2, sticky='ew')

button\_ADM.grid(column=0, row=4, columnspan=2, sticky='ew')

label\_chain\_lenght.grid(column=0, row=5, columnspan=2, sticky='ew')

self.slider\_chain\_lenght.grid(column=0, row=6, columnspan=2, sticky='ew')

label\_epsilon.grid(column=0, row=7, columnspan=2, sticky='ew')

self.slider\_epsilon.grid(column=0, row=8, columnspan=2, sticky='ew')

label\_coef.grid(column=0, row=9, columnspan=2, sticky='ew')

self.slider\_coef.grid(column=0, row=10, columnspan=2, sticky='ew')

label\_time.grid(column=0, row=11, columnspan=2, sticky='ew')

self.slider\_time.grid(column=0, row=12, columnspan=2, sticky='ew')

button\_discard\_param.grid(column=0, row=13, columnspan=2, sticky='ew')

self.label\_err\_msg.grid(column=0, row=14, columnspan=2, sticky='ew')

def button\_BE\_clicked(self):

self.plot.draw\_lists(backward\_euler.BackwardEuler(

0.01, self.slider\_coef.get(), self.slider\_chain\_lenght.get(),

self.slider\_epsilon.get(), self.slider\_time.get()

).execute())

def button\_FE\_clicked(self):

self.plot.draw\_lists(forward\_euler.ForwardEuler(

0.01, self.slider\_coef.get(), self.slider\_chain\_lenght.get(),

self.slider\_epsilon.get(), self.slider\_time.get()

).execute())

def button\_RK\_clicked(self):

self.plot.draw\_lists(runge\_kutta.Runge\_Kutt(

0.01, int(self.slider\_coef.get()), self.slider\_chain\_lenght.get(),

self.slider\_epsilon.get(), self.slider\_time.get()

).execute())

def button\_H\_clicked(self):

self.plot.draw\_lists(heun.Heun(

0.01, int(self.slider\_coef.get()), self.slider\_chain\_lenght.get(),

self.slider\_epsilon.get(), self.slider\_time.get()

).execute())

def button\_ADM\_clicked(self):

self.plot.draw\_lists(adams\_bashforth\_moulton.ABM(

0.01, int(self.slider\_coef.get()), self.slider\_chain\_lenght.get(),

self.slider\_epsilon.get(), self.slider\_time.get()

).execute())

def button\_discard\_param\_clicked(self):

self.slider\_epsilon.set(2 \* self.eps)

self.slider\_chain\_lenght.set(100)

self.slider\_coef.set(200)

self.check\_sliders()

def check\_sliders(self):

# print("hi")

if (self.slider\_chain\_lenght.get() <= self.slider\_epsilon.get()):

self.label\_err\_msg.config(text = "\n\nStatus: ERROR!\nTop Temp <= Env Temp")

return False

else:

self.label\_err\_msg.config(text = "\n\nStatus: DONE!")

return True

def \_\_del\_\_(self):

# os.remove('tmp.txt')

Pass

Файл adams\_bashforth\_moulton.py:

import numpy as np

import matplotlib.pyplot as plt

import my\_func as mf

class ABM:

def \_\_init\_\_(self, \_h = 0.01, \_coef = 0.1, \_chain\_len = 1, \_eps = 25, time\_ = 3):

self.h = \_h

self.coef = \_coef

self.chain\_len = \_chain\_len

self.eps = \_eps

self.x = np.array([0.0])

self.time = time\_

self.f = open('tmp.txt', 'w')

def \_\_del\_\_(self):

self.f.close()

pass

def RungeKutta4thOrder(self, x):

appr = int((self.time - 0)/self.h)

x = 0

y = self.coef \* self.chain\_len / (1 + self.coef) + 2 \* self.eps/1000

xsol = np.empty((0))

xsol = np.append(xsol, x)

y\_res = np.empty((0))

y\_res = np.append(y\_res, y)

for i in range(appr):

yp2 = x + mf.myFunc(x, self.coef, self.chain\_len)\*(self.h/2)

yp3 = x + mf.myFunc(yp2, self.coef, self.chain\_len)\*(self.h/2)

yp4 = x + mf.myFunc(yp3, self.coef, self.chain\_len)\*self.h

y = y + (self.h/6)\*(mf.myFunc(x, self.coef, self.chain\_len) + 2\*mf.myFunc(yp2, self.coef, self.chain\_len) + 2\*mf.myFunc(yp3, self.coef, self.chain\_len) + mf.myFunc(yp4, self.coef, self.chain\_len))

if (y > 1):

y = 1

x = x + self.h

xsol = np.append(xsol, x)

y\_res = np.append(y\_res, y)

return [xsol, y\_res]

def ABM4thOrder(self):

dx = int((self.time - 0) / self.h)

xrk = [self.x[0] + k \* self.h for k in range(dx + 1)]

[xx, yy] = self.RungeKutta4thOrder((xrk[0], xrk[3]))

print (xx)

print (yy)

self.x = xx

xsol = np.empty(0)

# xsol = np.append(xsol, self.x)

y = yy

yn = yy[0]

y\_res = np.empty(0)

# y\_res = np.append(y\_res, y)

for i in range(3, dx):

y0prime = mf.myFunc(y[i], self.coef, self.chain\_len)

y1prime = mf.myFunc(y[i - 1], self.coef, self.chain\_len)

y2prime = mf.myFunc(y[i - 2], self.coef, self.chain\_len)

y3prime = mf.myFunc(y[i - 3], self.coef, self.chain\_len)

ypredictor = y[i] + (self.h/24)\*(55\*y0prime - 59\*y1prime + 37\*y2prime - 9\*y3prime)

ypp = mf.myFunc(ypredictor, self.coef, self.chain\_len)

yn = y[i] + (self.h/24)\*(9\*ypp + 19\*y0prime - 5\*y1prime + y2prime)

if (yn > 1):

yn = 1

# print (yn)

xs = xx[i] + self.h

xsol = np.append(xsol, xs)

self.x = xsol

y\_res = np.append(y\_res, yn)

return [xsol, y\_res]

def execute(self):

[ts, ys] = self.ABM4thOrder()

print(len(ts))

for index in ts:

self.f.write(str(index) + ' ')

self.f.write('\n')

for index in ys:

self.f.write(str(index) + ' ')

self.f.write('\n')

t = np.arange(0, self.time, self.h)

for index in t:

self.f.write(str(index) + ' ')

self.f.write('\n')

for i in t:

T = 2 \* self.eps/2000 \* np.exp(np.sqrt((1 + self.coef) \* 9.8 / self.chain\_len) \* i) + 2 \* self.eps/2000 \* np.exp(-np.sqrt((1 + self.coef) \* 9.8 / self.chain\_len) \* i) + self.coef \* self.chain\_len / (1 + self.coef)

if (T > 1):

T = 1

self.f.write(str(T) + ' ')

self.f.write('\n')

return 0

Файл backward\_euler.py:

import matplotlib.pyplot as plt

import numpy as np

import my\_func as mf

class BackwardEuler:

def \_\_init\_\_(self, \_h = 0.01, \_coef = 0.1, \_chain\_len = 1, \_eps = 25, time\_ = 3):

self.h = \_h

self.coef = \_coef

self.chain\_len = \_chain\_len

self.eps = \_eps

self.time = time\_

self.f = open('tmp.txt', 'w')

def \_\_del\_\_(self):

self.f.close()

pass

def main\_BE(self):

appr = int((self.time - 0)/self.h)

j = 0

x = 0

y = {}

y[j] = self.coef \* self.chain\_len / (1 + self.coef) + 2 \* self.eps/1000

self.f.write(str(x) + ' ')

for i in range(appr):

F\_x\_t = mf.myFunc(x, self.coef, self.chain\_len)/(1+self.h)

j += 1

y[j] = y[j-1] + self.h\*F\_x\_t

print(y[j])

x += self.h

if (y[j] > 1):

y[j] = 1

self.f.write(str(x) + ' ')

self.f.write('\n')

return y

def execute(self):

ys = self.main\_BE()

for index in ys:

self.f.write(str(ys[index]) + ' ')

self.f.write('\n')

t = np.arange(0, self.time, self.coef)

for index in t:

self.f.write(str(index) + ' ')

self.f.write('\n')

for i in t:

T = 2 \* self.eps/2000 \* np.exp(np.sqrt((1 + self.coef) \* 9.8 / self.chain\_len) \* i) + 2 \* self.eps/2000 \* np.exp(-np.sqrt((1 + self.coef) \* 9.8 / self.chain\_len) \* i) + self.coef \* self.chain\_len / (1 + self.coef)

if (T > 1):

T = 1

self.f.write(str(T) + ' ')

self.f.write('\n')

return 0

Файл forward\_euler.py:

import matplotlib.pyplot as plt

import numpy as np

import my\_func as mf

class ForwardEuler:

def \_\_init\_\_(self, \_h = 0.01, \_coef = 0.1, \_chain\_len = 1, \_eps = 25, time\_ = 3):

self.h = \_h

self.coef = \_coef

self.chain\_len = \_chain\_len

self.eps = \_eps

self.time = time\_

self.f = open('tmp.txt', 'w')

def \_\_del\_\_(self):

self.f.close()

pass

def main\_FE(self):

appr = int((self.time - 0)/self.h)

j = 0

x = 0

y = {}

y[j] = self.coef \* self.chain\_len / (1 + self.coef) + 2 \* self.eps/1000

self.f.write(str(x) + ' ')

for i in range(appr):

F\_x\_t = mf.myFunc(x, self.coef, self.chain\_len)

j += 1

y[j] = y[j-1] + self.h\*F\_x\_t

print (y[j])

x += self.h

if (y[j] > 1):

y[j] = 1

self.f.write(str(x) + ' ')

self.f.write('\n')

return y

def execute(self):

ys = self.main\_FE()

for index in ys:

self.f.write(str(ys[index]) + ' ')

self.f.write('\n')

t = np.arange(0, self.time, self.coef)

for index in t:

self.f.write(str(index) + ' ')

self.f.write('\n')

for i in t:

T = 2 \* self.eps/2000 \* np.exp(np.sqrt((1 + self.coef) \* 9.8 / self.chain\_len) \* i) + 2 \* self.eps/2000 \* np.exp(-np.sqrt((1 + self.coef) \* 9.8 / self.chain\_len) \* i) + self.coef \* self.chain\_len / (1 + self.coef)

if (T > 1):

T = 1

self.f.write(str(T) + ' ')

self.f.write('\n')

return 0

Файл heun.py:

import matplotlib.pyplot as plt

import numpy as np

import my\_func as mf

class Heun:

def \_\_init\_\_(self, \_h = 0.01, \_coef = 0.1, \_chain\_len = 1, \_eps = 25, time\_ = 3):

self.h = \_h

self.coef = \_coef

self.chain\_len = \_chain\_len

self.eps = \_eps

self.time = time\_

self.f = open('tmp.txt', 'w')

def \_\_del\_\_(self):

self.f.close()

pass

def main\_H(self):

appr = int((self.time - 0)/self.h)

j = 0

x = 0

y = {}

y[j] = self.coef \* self.chain\_len / (1 + self.coef) + 2 \* self.eps/1000

self.f.write(str(x) + ' ')

for i in range(appr):

j += 1

y[j] = y[j-1] + (self.h/2)\*mf.myFunc(x, self.coef, self.chain\_len) + (self.h/2)\*mf.myFunc(x + mf.myFunc(x, self.coef, self.chain\_len) \* self.h, self.coef, self.chain\_len)

print(y[j])

if (y[j] > 1):

y[j] = 1

x += self.h

self.f.write(str(x) + ' ')

self.f.write('\n')

return y

def execute(self):

ys = self.main\_H()

for index in ys:

self.f.write(str(ys[index]) + ' ')

self.f.write('\n')

t = np.arange(0, self.time, self.h)

for index in t:

self.f.write(str(index) + ' ')

self.f.write('\n')

for i in t:

T = 2 \* self.eps/2000 \* np.exp(np.sqrt((1 + self.coef) \* 9.8 / self.chain\_len) \* i) + 2 \* self.eps/2000 \* np.exp(-np.sqrt((1 + self.coef) \* 9.8 / self.chain\_len) \* i) + self.coef \* self.chain\_len / (1 + self.coef)

if (T > 1):

T = 1

self.f.write(str(T) + ' ')

self.f.write('\n')

return 0

Файл runge\_kutta.py:

import matplotlib.pyplot as plt

import numpy as np

import my\_func as mf

class Runge\_Kutt:

def \_\_init\_\_(self, \_h = 0.01, \_coef = 0.1, \_chain\_len = 1, \_eps = 25, time\_ = 3):

self.h = \_h

self.coef = \_coef

self.chain\_len = \_chain\_len

self.eps = \_eps

self.time = time\_

self.f = open('tmp.txt', 'w')

def \_\_del\_\_(self):

self.f.close()

pass

def RKF45(self):

appr = int((self.time - 0)/self.h)

j = 0

x = 0

y = {}

y[j] = self.coef \* 1 / (1 + self.coef) + 2 \* self.eps/1000

self.f.write(str(x) + ' ')

for i in range(appr):

yp2 = x + mf.myFunc(x, self.coef, self.chain\_len)\*(self.h/5)

yp3 = x + mf.myFunc(x, self.coef, self.chain\_len)\*(3\*self.h/40) + mf.myFunc(yp2, self.coef, self.chain\_len)\*(9\*self.h/40)

yp4 = x + mf.myFunc(x, self.coef, self.chain\_len)\*(3\*self.h/10) - mf.myFunc(yp2, self.coef, self.chain\_len)\*(9\*self.h/10) + mf.myFunc(yp3, self.coef, self.chain\_len)\*(6\*self.h/5)

yp5 = x - mf.myFunc(x, self.coef, self.chain\_len)\*(11\*self.h/54) + mf.myFunc(yp2, self.coef, self.chain\_len)\*(5\*self.h/2) - mf.myFunc(yp3, self.coef, self.chain\_len)\*(70\*self.h/27) + mf.myFunc(yp4, self.coef, self.chain\_len)\*(35\*self.h/27)

yp6 = x + mf.myFunc(x, self.coef, self.chain\_len)\*(1631\*self.h/55296) + mf.myFunc(yp2, self.coef, self.chain\_len)\*(175\*self.h/512) + mf.myFunc(yp3, self.coef, self.chain\_len)\*(575\*self.h/13824) + mf.myFunc(yp4, self.coef, self.chain\_len)\*(44275\*self.h/110592) + mf.myFunc(yp5, self.coef, self.chain\_len)\*(253\*self.h/4096)

j += 1

y[j] = y[j-1] + self.h\*(37\*mf.myFunc(x, self.coef, self.chain\_len)/378 + 22 \* self.eps\*mf.myFunc(yp3, self.coef, self.chain\_len)/621 + 125\*mf.myFunc(yp4, self.coef, self.chain\_len)/594 + 512\*mf.myFunc(yp6, self.coef, self.chain\_len)/1771)

x += self.h

if (y[j] > 1):

y[j] = 1

self.f.write(str(x) + ' ')

self.f.write('\n')

return y

def execute(self):

ys = self.RKF45()

for index in ys:

self.f.write(str(ys[index]) + ' ')

self.f.write('\n')

t = np.arange(0, self.time, self.h)

for index in t:

self.f.write(str(index) + ' ')

self.f.write('\n')

for i in t:

T = 2 \* self.eps/2000 \* np.exp(np.sqrt((1 + self.coef) \* 9.8 / self.chain\_len) \* i) + 2 \* self.eps/2000 \* np.exp(-np.sqrt((1 + self.coef) \* 9.8 / self.chain\_len) \* i) + self.coef \* self.chain\_len / (1 + self.coef)

if (T > 1):

T = 1

self.f.write(str(T) + ' ')

self.f.write('\n')

return 0

Файл my\_func.py:

import numpy as np

def myFunc(x, mu, l):

dy = 4.9 \* x\*\*2 \* (mu + 1) / l - 9.8 \* mu \* x

# dy = 5.self.time9 \* (x)\*\*2 - 0.98 \* (x)

return dy