БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

**Лабораторная работа №3**

**Решение задачи Коши**

**Вариант 12**

Выполнил: Белоушко Степан

3 курс 7 группа

Преподаватель:

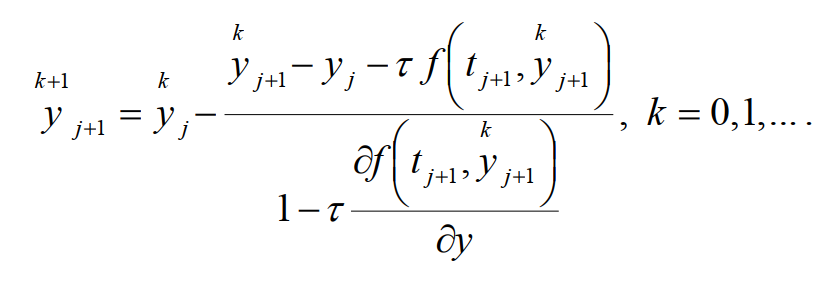
Будник Анатолий Михайлович

**Неявный метод Эйлера**

Неявный метод Эйлера имеет вид:



Для его реализации методом Ньютона, запишем для нахождения неизвестной величины итерационный метод Ньютона:



Листинг программы:

import math

import numpy as np

def f(y, t):

    return -t\*\*2 \* y\*\*2 + (t\*\*2 - 0.5)/(1+0.5\*t)\*\*2

def f\_(y, t):

    return -2 \* t\*\*2 \* y

def function(yk, y, i, h, a):

    return yk - (yk-y[i]-h\*f(yk, a+(i+1)\*h)) / (1-h\*f\_(yk, a+(i+1)\*h))

def newtonMethod(y, i, h, a):

    yk1 = y[i]

    yk2 = function(yk1, y, i, h, a)

    while (abs(yk2 - yk1) > h\*\*2):

        yk1 = yk2

        yk2 = function(yk1, y, i, h, a)

    return yk2

def eulerMethod(a, b, N, u0):

    h = (b-a)/N

    y = np.array([])

    y = np.append(y, u0)

    for i in range(N):

        yk = newtonMethod(y, i, h, a)

        y = np.append(y, y[i] + h\*f(yk, a+(i+1)\*h))

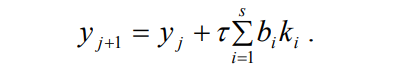
    return y

y = eulerMethod(0, 1, 10, 1)

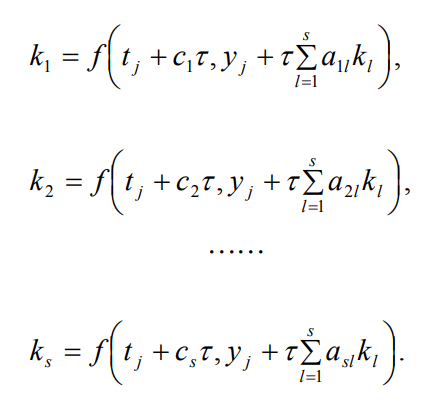
print(y)

Метод Рунге-Кутта

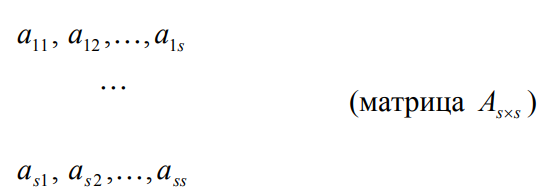
Метод Рунге-Кутта имеет вид:

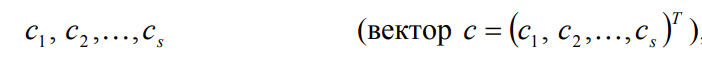


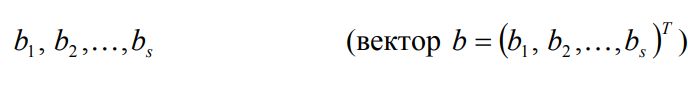
где



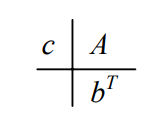
а все необходимые коэффициенты:





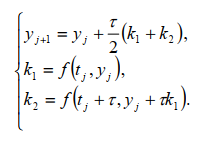


задаются в виде таблицы Бутчера следующего вида:

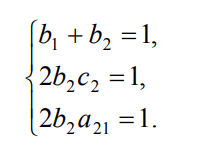


В моём случае таблица Бутчера выглядит следующим образом:

Запишу в виде системы



Определим порядок точности метода. Проверим условия для второго порядка точности:



Условия выполнены. Для третьего порядка точности условия выполнены не будут. Следовательно, имеем метод второго порядка точности с локальной погрешностью

Листинг программы:

import math

import numpy as np

def f(y, t):

    return -t\*\*2 \* y\*\*2 + (t\*\*2 - 0.5)/(1+0.5\*t)\*\*2

def rungeKuttMethod(a, b, N, u0):

    h = (b-a)/N

    y = np.array([])

    y = np.append(y, u0)

    for i in range(N):

        k1 = f(y[i], a+h\*i)

        k2 = f(y[i]+h\*k1, a+(i+1)\*h)

        y = np.append(y, y[i] + h/2\*(k1+k2))

    return y

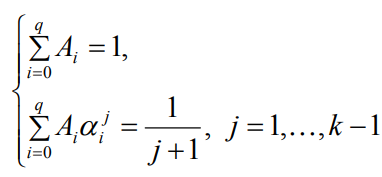
y= rungeKuttMethod(0, 1, 10, 1)

print(y)

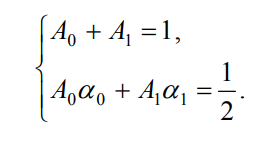
**Метод последовательного повышения порядка точности**

В общем случае метод будет иметь вид:

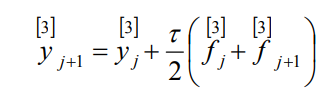
где для порядка точности находятся из системы:



В нашем случае: k = 2, q = 1,



При получим .



По итогу метод в реализации совпадает с методом Рунге-Кутта из прошлого пункта.

Порядок точности данного метода – второй, локальная погрешность будет

Листинг программы:

import math

import numpy as np

def f(t, y):

    return -t\*\*2 \* y\*\*2 + (t\*\*2 - 0.5)/(1+0.5\*t)\*\*2

def MPPPT(a, b, N, u0):

    h = (b-a)/N

    y = np.array([])

    y = np.append(y, u0)

    for i in range(N):

        y = np.append(y, y[i]+h/2\*(f(a+h\*i, y[i])+

        f(a+h\*(i+1), y[i]+h\*f(a+h\*i, y[i]))))

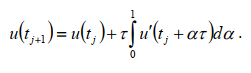
    return y

y= MPPPT(0, 1, 10, 1)

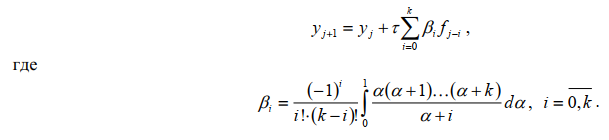
print(y)

**Экстраполяционный метод Адамса**

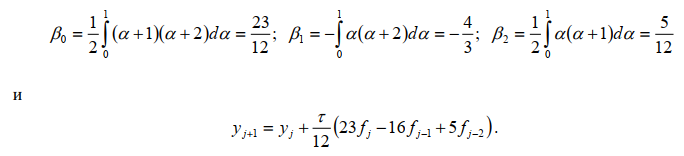
Делая вновь, как и в способе Рунге-Кутта, замену переменных в интегральном тождестве, перепишу его в виде



Тогда описанная выше процедура приведет к методу

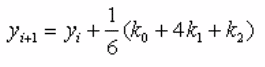


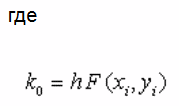
При порядке, равном 3, коэффициенты будут иметь вид

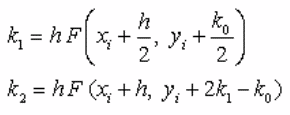


Так как метод третьего порядка, то локальная погрешность метода будет .

Метод Адамса использует 3 предыдущих значения функции, поэтому их нужно найти другим методом. В моём случае, это метод Рунге-Кутта 3 порядка. Значения в трёх узлах будут считаться по формулам:







Листинг программы:

import math

import numpy as np

def f(t, y):

    return -t\*\*2 \* y\*\*2 + (t\*\*2 - 0.5)/(1+0.5\*t)\*\*2

def getRungeKutt(a, h, i, y):

    k0 = h\*f(a+i\*h, y)

    k1 = h\*f(a+i\*h+h/2, y+k0/2)

    k2 = h\*f(a+(i+1)\*h, y+2\*k1-k0)

    return y+1/6\*(k0+4\*k1+k2)

def adamsMethod(a, b, N, u0):

    h = (b-a)/N

    y = np.array([])

    y = np.append(y, u0)

    y = np.append(y, getRungeKutt(a, h, 0, y[0]))

    y = np.append(y, getRungeKutt(a, h, 1, y[1]))

    for i in range(2, N):

        y = np.append(y, y[i]+h/12\*(

        23\*f(a+i\*h, y[i])

        -16\*f(a+(i-1)\*h, y[i-1])

        +5\*f(a+(i-2)\*h, y[i-2])))

    return y

y = adamsMethod(0, 1, 10, 1.)

print(y)

**Результаты**

Метод Эйлера

[1. 0.95464421 0.91329128 0.87539258 0.84047899 0.80815128

0.77807294 0.74996436 0.72359727 0.69878885 0.6753952 ]

Метод Рунге-Кутта

[1. 0.95232653 0.90899737 0.86944773 0.83320647 0.79987727

0.76912414 0.74066016 0.71423874 0.68964667 0.66669874]

Метод ПППТ

[1. 0.95232653 0.90899737 0.86944773 0.83320647 0.79987727

0.76912414 0.74066016 0.71423874 0.68964667 0.66669874]

Метод Адамса

[1. 0.95238036 0.90908922 0.86952344 0.83326088 0.79990474

0.76911971 0.74062018 0.71416129 0.6895318 0.66654846]

Невязки по методам относительно метода Адамса:

Метод Эйлера

[0. 0.00226385 0.00420206 0.00586914 0.00721811 0.00824654

0.00895323 0.00934418 0.00943598 0.00925705 0.00884674]

Метод Рунге-Кутта

[0.00000000e+00 5.38322222e-05 9.18496877e-05 7.57070954e-05

5.44074478e-05 2.74684618e-05 4.42931403e-06 3.99846674e-05

7.74480479e-05 1.14866569e-04 1.50283010e-04]

Метод ПППТ

[0.00000000e+00 5.38322222e-05 9.18496877e-05 7.57070954e-05

5.44074478e-05 2.74684618e-05 4.42931403e-06 3.99846674e-05

7.74480479e-05 1.14866569e-04 1.50283010e-04]

**Выводы**

Погрешность метода Эйлера самая большая из всех. Это подтверждается и теорией, так как у него локальная погрешность равна . Метод Эйлера вычисляет решение с наименьшей точностью.

Метод Рунге-Кутты имеет второй порядок точности, его локальная погрешность равна . . Невязки гораздо лучше, чем у метода Эйлера, что и ожидалось. Однако, погрешности имеют тенденцию к накоплению.

Метод ПППТ имеет такую же локальную погрешность, как и метод Рунге-Кутта, В моём случае он полностью совпал с предыдущим методом, поэтому и погрешности одинаковые.

Локальная погрешность метода Адамса самая маленькая среди остальных методов, , что равно Поэтому его выбор в качестве точного решения является оптимальным. Однако, у него тоже есть погрешность, поэтому значения невязок предыдущих методов могут быть занижены относительно реальных невязок.