Университет ИТМО

Факультет программной инженерии и компьютерной техники

Направление подготовки 09.03.04 Программная инженерия

Дисциплина «Вычислительная математика»

**Отчет**

По лабораторной работе №6

Вариант 6

Выполнил:

*Манжиков Н.C*

Преподаватель: Рыбаков Степан Дмитриевич

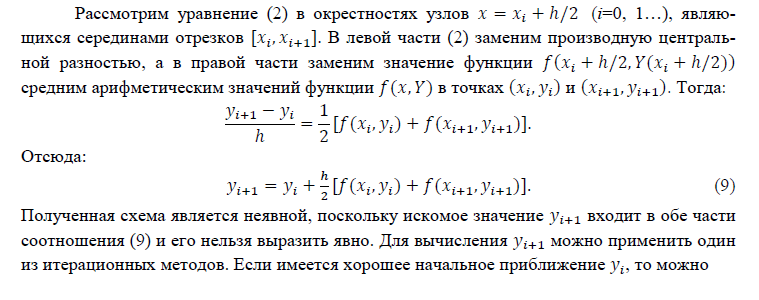
Санкт-Петербург, 2023 г.

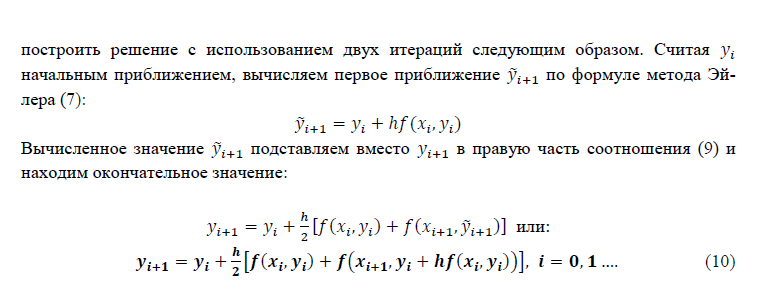
Цель работы

Решить задачу Коши численными методами.

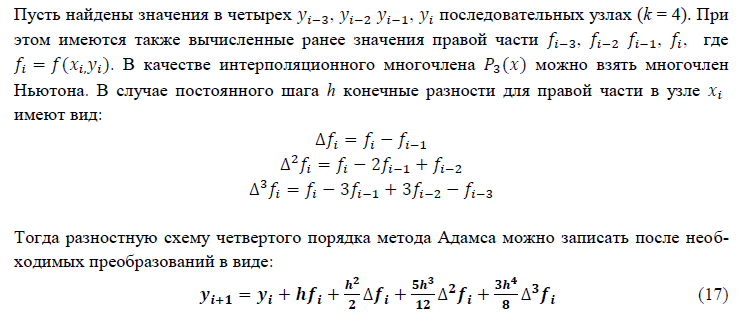
Для решения использовать одношаговые и многошаговые методы.

Модифицированный метод Эйлера

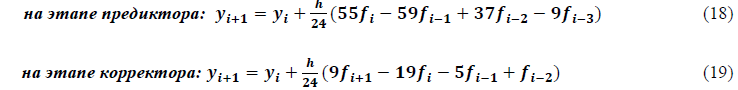




Метода Адамса



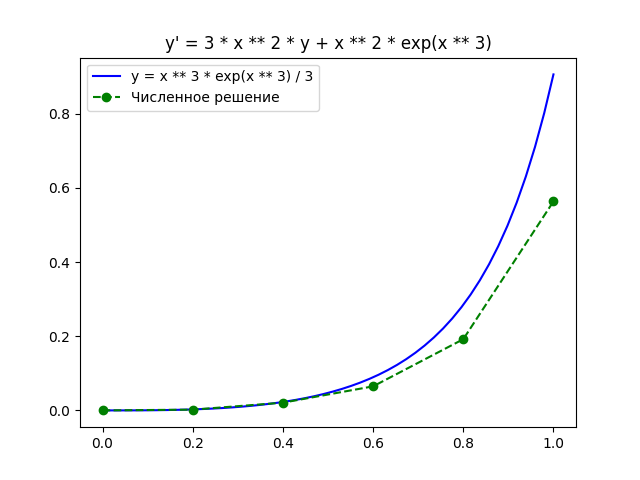
Рабочая формула метода



Пример работы программы

Изображение выглядит как текст, снимок экрана, Мультимедийное программное обеспечение

Автоматически созданное описание



import numpy as np  
import matplotlib.pyplot as plt  
from prettytable import PrettyTable  
  
from methods import improved\_euler\_method, adams\_method, runge\_rule, runge\_kutta\_method  
from util import Equation, Fun, format\_float as ff  
  
EQUATIONS = [  
 Equation(  
 dif=Fun('y + (1 + x) \* y \*\* 2'),  
 ex=Fun('-1 / x'),  
 a=1,  
 b=1.5,  
 y0=-1  
 ),  
 Equation(  
 dif=Fun('3 \* x \*\* 2 \* y + x \*\* 2 \* exp(x \*\* 3)'),  
 ex=Fun('x \*\* 3 \* exp(x \*\* 3) / 3'),  
 a=0,  
 b=1,  
 y0=0  
 ),  
 Equation(  
 dif=Fun('cos(x) - y'),  
 ex=Fun('(cos(x) + sin(x)) / 2'),  
 a=0,  
 b=0.1,  
 y0=0.5  
 )  
]  
  
METHODS = [  
 ('Усовершенствованный метод Эйлера', improved\_euler\_method, 2),  
 ('Метод Адамса', adams\_method, 4),  
 ('Метод Рунге-Кутта 4-го порядка', runge\_kutta\_method, 4)  
]  
  
  
def main():  
 for i, eq in enumerate(EQUATIONS, 1):  
 print(f"{i}.\ty' = {eq.dif!s}\n"  
 f"\tx[{eq.a}; {eq.b}], y({eq.a}) = {eq.y0}")  
 eq = EQUATIONS[int(input('Выберите дифференциальное уравнение: ')) - 1]  
  
 for i, (name, \*\_) in enumerate(METHODS, 1):  
 print(f"{i}. {name}")  
 method = METHODS[int(input('Выберите метод решения: ')) - 1]  
 fun, p = method[1], method[2]  
  
 h = float(input('Выберите шаг: '))  
  
 x, y = fun(eq, h)  
 f\_dif = [eq.dif(xi, yi) for xi, yi in zip(x, y)]  
 f\_ex = [eq.ex(xi) for xi in x]  
 eps = [abs(yi - ex) for yi, ex in zip(y, f\_ex)]  
  
 table = PrettyTable(['i', 'x', 'y', 'f(x, y)', 'Точное решение', 'eps'])  
 table.add\_rows([  
 (i, ff(x[i]), ff(y[i]), ff(f\_dif[i]), ff(f\_ex[i]), ff(eps[i]))  
 for i in range(len(x))  
 ])  
 print(table)  
  
 runge = runge\_rule(y[-1], fun(eq, 2 \* h)[1][-1], p)  
  
 print('Погрешность:', max(eps))  
 print('По Рунге:', runge)  
  
 x\_ex = np.linspace(eq.a, eq.b)  
 y\_ex = np.vectorize(eq.ex)(x\_ex)  
  
 plt.title(f"y' = {eq.dif}")  
 plt.plot(x\_ex, y\_ex, 'blue', label=f"y = {eq.ex}")  
 plt.plot(x, y, 'go--', label="Численное решение")  
 plt.legend()  
 plt.show()  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

def improved\_euler\_method(eq, h):  
 n = int((eq.b - eq.a) / h)  
 x, y = [eq.a], [eq.y0]  
 for i in range(n):  
 y0 = y[i] + h / 2 \* eq.dif(x[i], y[i])  
 y.append(y[i] + h \* eq.dif(x[i] + h / 2, y0))  
 x.append(x[i] + h)  
  
 return x, y  
  
  
def adams\_method(eq, h):  
 n = int((eq.b - eq.a) / h)  
 x, y = improved\_euler\_method(eq, h)  
  
  
 f = eq.dif  
 for i in range(3, len(x)):  
 k = [f(x[i - q], y[i - q]) for q in range(4)]  
 df = k[0] - k[1]  
 d2f = k[0] - 2 \* k[1] + k[2]  
 d3f = k[0] - 3 \* k[1] + 3 \* k[2] - k[3]  
 y[i] = (  
 y[i - 1] +  
 1 \* h \*\* 1 \* k[1] +  
 1 \* h \*\* 2 \* df / 2 +  
 5 \* h \*\* 3 \* d2f / 12 +  
 3 \* h \*\* 4 \* d3f / 8  
 ) # вычисление следующего значения y на основе полученных производных и коэффициентов  
  
 return x, y  
  
def runge\_kutta\_method(eq, h):  
 n = int((eq.b - eq.a) / h)  
 x, y = [eq.a], [eq.y0]  
 f = eq.dif  
 for i in range(n):  
 y.append(rk4\_step(x[i], y[i], h, f))  
 x.append(x[i] + h)  
 return x, y  
def rk4\_step(x, y, h, f):  
 k1 = h \* f(x, y)  
 k2 = h \* f(x + h/2, y + k1/2)  
 k3 = h \* f(x + h/2, y + k2/2)  
 k4 = h \* f(x + h, y + k3)  
 return y + (k1 + 2\*k2 + 2\*k3 + k4)/6  
  
def runge\_rule(yh, y2h, p):  
 return abs((yh - y2h) / (2 \*\* p - 1))

from dataclasses import dataclass  
from math import \*  
  
  
class Fun:  
 def \_\_init\_\_(self, expr):  
 self.\_\_expr = expr  
  
 def \_\_call\_\_(self, x, y=None):  
 return eval(self.\_\_expr)  
  
 def \_\_str\_\_(self):  
 return self.\_\_expr  
  
  
@dataclass  
class Equation:  
 dif: Fun  
 ex: Fun  
 a: float  
 b: float  
 y0: float  
  
  
def format\_float(x):  
 return f'{x:.5f}'

Вывод

Во время выполнения лабораторной работы я изучил работу одношаговых и многошаговых методом решения задачи Коши. Точность этих методом примерно схожая, но метод Эйлера является мене устойчивым из-за отсутствия коррекции на каждом шаге.