Университет ИТМО

Факультет программной инженерии и компьютерной техники

**Лабораторная работа №6**

по «Вычислительной математике»

Численное решение обыкновенных дифференциальных уравнений

Выполнил:

Студент группы P32151

Кортыш А.О.

Преподаватель:

Малышева Т.А.

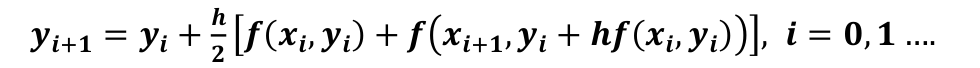
Санкт-Петербург

2023

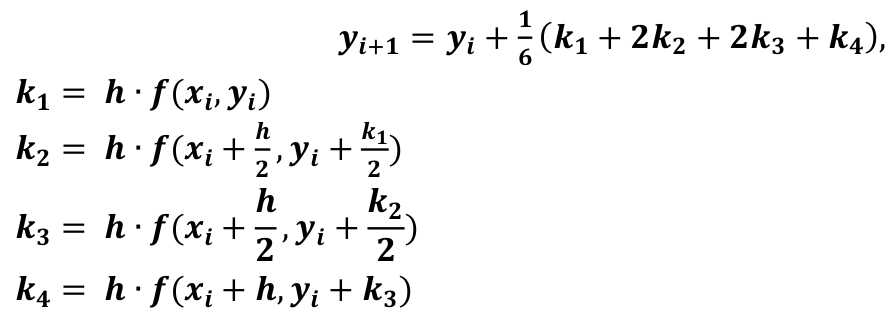
**Цель работы.**

Решить задачу Коши для обыкновенных дифференциальных уравнений численными методами.

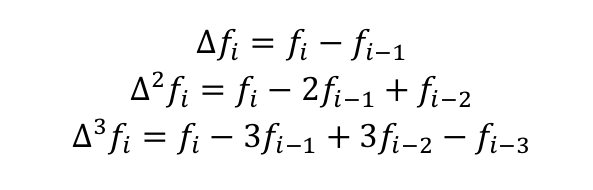
## Рабочие формулы используемых методов.

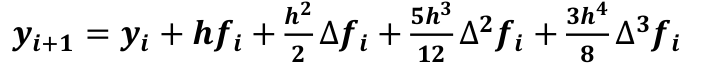
Модифицированный метод Эйлера: 

Метод Рунге-Кутта четвертого порядка:



Метод Адамса:





**Листинг программы.**

def solve\_single\_turn(method: Callable[[Callable[[float, float], float], float, float, float], float],

equation\_number: int, x\_0: float, y\_0: float, h: float, x\_n: float, epsilon: float, rang: int) -> List[float]:

ys = [y\_0]

cur\_x = x\_0

cur\_h = h

y\_h = 1e9

y\_h\_2 = -1e9

cur\_y = y\_0

while abs(cur\_x - x\_n) > epsilon:

while abs(y\_h - y\_h\_2) / (2\*\*rang - 1) > epsilon:

y\_h = solve\_without\_correction(method, equation\_number, cur\_x, cur\_y, cur\_h, cur\_x+h, epsilon)[-1]

y\_h\_2 = solve\_without\_correction(method, equation\_number, cur\_x, cur\_y, cur\_h/2, cur\_x+h, epsilon)[-1]

cur\_h /= 2

ys.append(y\_h\_2)

cur\_x += h

cur\_y = ys[-1]

y\_h = 1e9

y\_h\_2 = -1e9

cur\_h = h

return ys

def solve\_without\_correction(method: Callable[[Callable[[float, float], float], float, float, float], float],

equation\_number: int, x\_0: float, y\_0: float, h: float, x\_n: float, epsilon: float) -> List[float]:

f\_x\_y = lambdify([x, y], equations[equation\_number], 'numpy')

ys = [y\_0]

cur\_x = x\_0

while abs(cur\_x - x\_n) > epsilon:

y\_h = method(f\_x\_y, cur\_x, ys[-1], h)

ys.append(y\_h)

cur\_x += h

return ys

def modified\_euler(f\_x\_y: Callable[[float, float], float], x\_0: float, y\_0: float, h: float) -> float:

return y\_0 + h/2\*(f\_x\_y(x\_0, y\_0) + f\_x\_y(x\_0 + h, y\_0 + h\*f\_x\_y(x\_0, y\_0)))

def runge\_kutt(f\_x\_y: Callable[[float, float], float], x\_0: float, y\_0: float, h: float) -> float:

k1 = h \* f\_x\_y(x\_0, y\_0)

k2 = h\*f\_x\_y(x\_0 + h / 2, y\_0 + k1 / 2)

k3 = h\*f\_x\_y(x\_0 + h / 2, y\_0 + k2 / 2)

k4 = h\*f\_x\_y(x\_0 + h, y\_0 + k3)

return y\_0 + 1/6\*(k1 + 2\*k2 + 2\*k3 + k4)

def adams(equation\_number: int, x\_0: float, y\_0: float, h: float, x\_n: float, epsilon) -> List[float]:

f\_x\_y = lambdify([x, y], equations[equation\_number], 'numpy')

first\_four = solve\_single\_turn(runge\_kutt, equation\_number, x\_0, y\_0, h, x\_0 + 3\*h, 0.001, 4)

ys = first\_four[::]

last\_four\_xs = [x\_0 + i\*h for i in range(4)]

last\_four\_ys = first\_four[::]

last\_four\_funcs = [f\_x\_y(x\_i, y\_i) for x\_i, y\_i in zip(last\_four\_xs, last\_four\_ys)]

cur\_x = x\_0 + 3\*h

while abs(cur\_x + h - x\_n) < epsilon or cur\_x + h < x\_n:

delta = last\_four\_funcs[-1] - last\_four\_funcs[-2]

delta\_sq = last\_four\_funcs[-1] - 2 \* last\_four\_funcs[-2] + last\_four\_funcs[-3]

delta\_cube = last\_four\_funcs[-1] - 3 \* last\_four\_funcs[-2] + 3 \* last\_four\_funcs[-3] - last\_four\_funcs[-4]

cur\_y = last\_four\_ys[-1] + h \* last\_four\_funcs[-1] + h\*\*2/2 \* delta + 5/12\*h\*\*3\*delta\_sq + 3/8\*h\*\*4\*delta\_cube

last\_four\_funcs.pop(0)

last\_four\_xs.pop(0)

last\_four\_ys.pop(0)

cur\_x += h

ys.append(cur\_y)

last\_four\_xs.append(cur\_x)

last\_four\_ys.append(cur\_y)

last\_four\_funcs.append(f\_x\_y(cur\_x, cur\_y))

return ys

**Примеры и результат работы программы**

**Пример 1.**

Input:

1. y' = -2\*x\*y + exp(-x\*\*2)\*sin(x)

2. y' = x\*\*2\*exp(-x) - y

3. y' = -3\*x\*\*2\*y + exp(-x\*\*3)\*cos(x)

Введите номер уравнения: 1

Введите начальные условия x\_0, y\_0 через пробел: 2 2

Введите конец интервала дифференцирования: 4

Введите шаг: 0.1

Введите точность: 0.0001

Модифицированный метод Эйлера

x 2.0 2.1 2.2 2.3 2.4 2.5 2.6 2.7 2.8 2.9 3.0 3.1 3.2 3.3 3.4 3.5 3.6 3.7 3.8 3.9 4.0

y 2.0 1.3284 0.8649 0.5519 0.3452 0.2117 0.1272 0.0749 0.0433 0.0246 0.0137 0.0074 0.004 0.0021 0.0011 0.0006 0.0003 0.0001 0.0001 0.0 0.0

Метод Рунге-Кутта

x 2.0 2.1 2.2 2.3 2.4 2.5 2.6 2.7 2.8 2.9 3.0 3.1 3.2 3.3 3.4 3.5 3.6 3.7 3.8 3.9 4.0

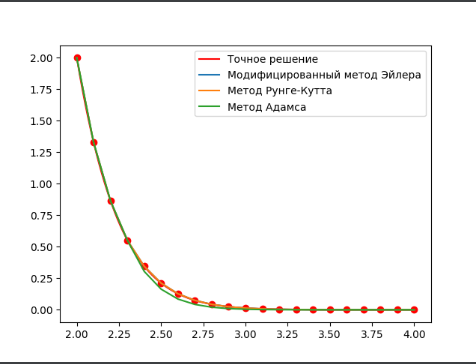
y 2.0 1.3284 0.8648 0.5518 0.3451 0.2116 0.1271 0.0748 0.0432 0.0244 0.0135 0.0074 0.0039 0.002 0.001 0.0005 0.0003 0.0001 0.0001 0.0 0.0

Метод Адамса

x 2.0 2.1 2.2 2.3 2.4 2.5 2.6 2.7 2.8 2.9 3.0 3.1 3.2 3.3 3.4 3.5 3.6 3.7 3.8 3.9 4.0

y 2.0 1.3284 0.8648 0.5518 0.3045 0.1638 0.0851 0.0427 0.0206 0.0096 0.0043 0.0019 0.0008 0.0003 0.0001 0.0 0.0 0.0 0.0 0.0 0.0

error 0.0477



**Пример 2.**

Input:

1. y' = -2\*x\*y + exp(-x\*\*2)\*sin(x)

2. y' = x\*\*2\*exp(-x) - y

3. y' = -3\*x\*\*2\*y + exp(-x\*\*3)\*cos(x)

Введите номер уравнения: 2

Введите начальные условия x\_0, y\_0 через пробел: 0 0

Введите конец интервала дифференцирования: 3

Введите шаг: 0.3

Введите точность: 0.001

Модифицированный метод Эйлера

x 0.0 0.3 0.6 0.9 1.2 1.5 1.8 2.1 2.4 2.7 3.0

y 0.0 0.007 0.041 0.1 0.174 0.251 0.321 0.378 0.417 0.44 0.447

Метод Рунге-Кутта

x 0.0 0.3 0.6 0.9 1.2 1.5 1.8 2.1 2.4 2.7 3.0

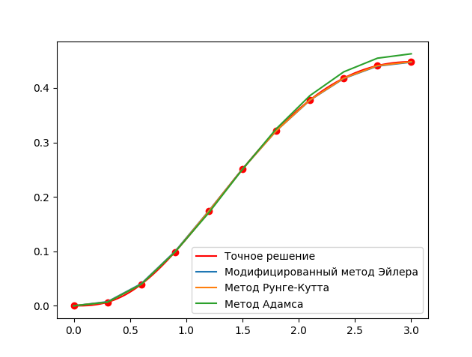
y 0.0 0.007 0.04 0.099 0.173 0.251 0.321 0.378 0.418 0.441 0.448

Метод Адамса

x 0.0 0.3 0.6 0.9 1.2 1.5 1.8 2.1 2.4 2.7 3.0

y 0.0 0.007 0.04 0.099 0.171 0.251 0.325 0.386 0.43 0.455 0.463

error 0.015



**Пример 3.**

Input:

1. y' = -2\*x\*y + exp(-x\*\*2)\*sin(x)

2. y' = x\*\*2\*exp(-x) - y

3. y' = -3\*x\*\*2\*y + exp(-x\*\*3)\*cos(x)

Введите номер уравнения: 3

Введите начальные условия x\_0, y\_0 через пробел: 0 0

Введите конец интервала дифференцирования: 2

Введите шаг: 0.1

Введите точность: 0.001

Модифицированный метод Эйлера

x 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0 1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9 2.0

y 0.0 0.1 0.197 0.287 0.365 0.422 0.454 0.456 0.429 0.377 0.309 0.235 0.166 0.108 0.064 0.035 0.017 0.008 0.003 0.001 0.0

Метод Рунге-Кутта

x 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0 1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9 2.0

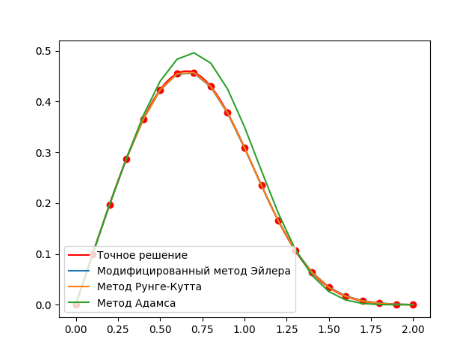
y 0.0 0.1 0.197 0.288 0.365 0.423 0.455 0.457 0.43 0.378 0.31 0.235 0.166 0.107 0.063 0.034 0.017 0.007 0.003 0.001 0.0

Метод Адамса

x 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0 1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9 2.0

y 0.0 0.1 0.197 0.288 0.372 0.44 0.483 0.496 0.476 0.425 0.35 0.265 0.181 0.11 0.058 0.026 0.009 0.003 0.0 0.0 -0.0

error 0.047



**Выводы.**

Реализация численных методов решения обыкновенных дифференциальных уравнений самая сложная задача, которую необходимо было решать в рамках данного курса, на мой взгляд. Основная сложность связана с необходимостью хардкодить большое количество данных по сравнению с предыдущими лабораторными, что усложняет модификацию и поддержку программы.