Университет ИТМО

**Отчет**

По лабораторной работе №3

Решение нелинейных уравнений “Метод Адамса”

Выполнил:

Нодири Хисравхон

P3231

Преподаватель:

Перл Ольга Вячеславовна

Санкт-Петербург

2024

Оглавление

[Задача 3](#_Toc162381748)

[Описание метода 3](#_Toc162381749)

[Блок схема алгоритма: 5](#_Toc162381750)

[Код программы 6](#_Toc162381751)

[Примеры работ 7](#_Toc162381752)

[Вывод 8](#_Toc162381753)

# Задача

## Реализуйте метод Адамса для решения обыкновенных дифференциальных уравнений по начальному значению (задача Коши) в интервале от a до b [a,b].

## f

## epsilon

## a

## y(a)

## b

## f - номер уравнения, где уравнение в виде y'=f(x,y). Вы должны получить функцию по номеру из входных данных в методе get\_function.

## Вы должны определить и пересчитать шаг h самостоятельно.

## Вы должны вычислить и вернуть y(b) с разницей, не превышающей epsilon.

## Подсказка: Чтобы использовать метод Адамса для решения задачи Коши, вам нужно больше 1 начальной точки. Поэтому вам также необходимо реализовать еще один метод для вычисления 3 дополнительных разгоночных точек. Вы можете выбрать этот метод самостоятельно, но если вы неправильно рассчитаете начальный набор точек, то это может повлиять на все решение.

## Описание метода

Методы Адамса — это семейство численных методов для решения обыкновенных дифференциальных уравнений (ОДУ), основанных на идеях интерполяции и экстраполяции. Они подразделяются на два основных типа: методы Адамса-Башфорта, которые являются явными, и методы Адамса-Моултона, которые являются неявными. Оба эти метода используют значения производных, полученных на предыдущих шагах, чтобы оценить решение на новом шаге.

Методы Адамса-Башфорта

Эти методы являются явными и используют прошлые значения производной функции для экстраполяции значения функции в будущем. Например, формула четырёхшагового метода Адамса-Башфорта имеет вид:

yn+1 = yn + h/24(55fn - 59 fn-1 + 37fn-2 - 9fn-3)

Где yn - текущее значение функции, fn - значение производной функции в точке yn, а h - шаг сетки.

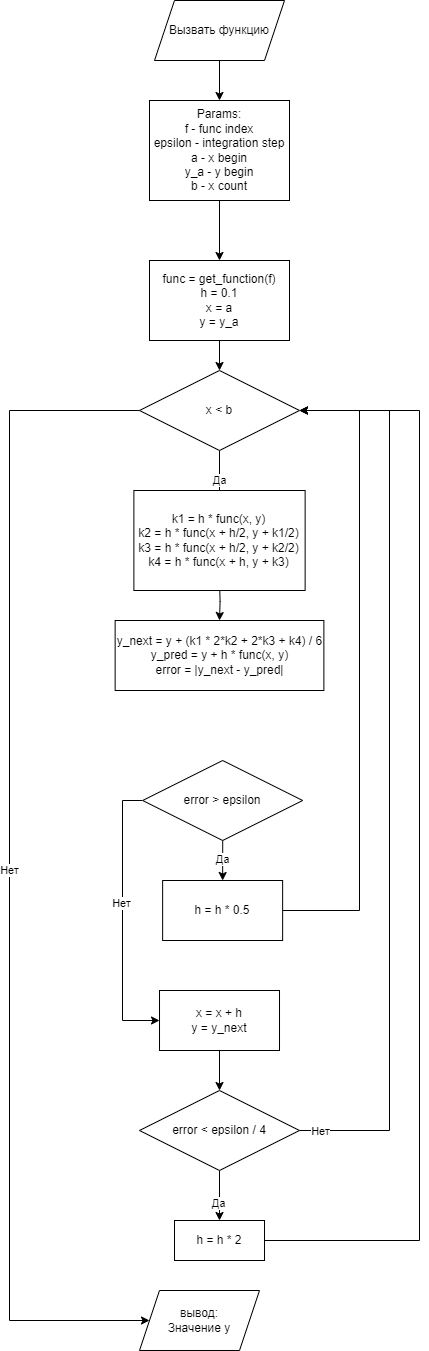
Методы Адамса-Моултона

Эти методы являются неявными и требуют решения уравнения относительно yn+1 на каждом шаге. Они используют текущие и прошлые значения производной для интерполяции значения функции. Пример формулы трёхшагового метода Адамса-Моултона:

yn+1 = yn + h/24(9fn+1 + 19fn - 5fn-1 + fn-2)

Преимуществом методов Адамса является их высокая точность и эффективность при решении гладких дифференциальных уравнений. Однако, методы могут быть нестабильными при решении жёстких ОДУ, и неявные методы требуют решения нелинейных уравнений на каждом шаге, что может быть вычислительно сложно.

# Блок схема алгоритма



## Код программы

#!/bin/python3

**import** math

**import** os

**import** random

**import** re

**import** sys

**class** **Result**:

**def** **first\_function**(x: float, y: float):

**return** math.sin(x)

**def** **second\_function**(x: float, y: float):

**return** (x \* y)/2

**def** **third\_function**(x: float, y: float):

**return** y - (2 \* x)/y

**def** **fourth\_function**(x: float, y: float):

**return** x + y

**def** **default\_function**(x:float, y: float):

**return** 0.0

# How to use this function:

# func = Result.get\_function(4)

# func(0.01)

**def** **get\_function**(n: int):

**if** n == 1:

**return** Result.first\_function

**elif** n == 2:

**return** Result.second\_function

**elif** n == 3:

**return** Result.third\_function

**elif** n == 4:

**return** Result.fourth\_function

**else**:

**return** Result.default\_function

#

# Complete the 'solveByAdams' function below.

#

# The function is expected to return a DOUBLE.

# The function accepts following parameters:

# 1. INTEGER f

# 2. DOUBLE epsilon

# 3. DOUBLE a

# 4. DOUBLE y\_a

# 5. DOUBLE b

#

**def** **solveByAdams**(f, epsilon, a, y\_a, b):

func = Result.get\_function(f)

h = 0.1

x = a

y = y\_a

**while** x < b:

k1 = h \* func(x, y)

k2 = h \* func(x + h/2, y + k1/2)

k3 = h \* func(x + h/2, y + k2/2)

k4 = h \* func(x + h, y + k3)

y\_next = y + (k1 + 2\*k2 + 2\*k3 + k4) / 6

y\_pred = y + h \* func(x, y)

error = abs(y\_next - y\_pred)

**if** error > epsilon:

h = h \* 0.5

**else**:

x += h

y = y\_next

**if** error < epsilon / 4:

h = h \* 2

**return** y

**if** \_\_name\_\_ == '\_\_main\_\_':

f = int(input().strip())

epsilon = float(input().strip())

a = float(input().strip())

y\_a = float(input().strip())

b = float(input().strip())

result = Result.solveByAdams(f, epsilon, a, y\_a, b)

print(str(result) + '\n')

# Примеры работ

Пример 1:

a = 0 # Начало интервала

b = 1 # Конец интервала

y\_a = 0 # Начальное значение y при x = 0

f = 1 # Индекс функции x, определенной как x в классе Result

epsilon = 0.01 # Шаг

# Вызов функции

result = Result.solveByAdams(f, epsilon, a, y\_a, b)

print(result) # Ожидаемый результат около 0.5, поскольку y(1) = ½

Пример 2:

a = 0

b = 1

y\_a = 1 # e^0 = 1

f = 2 # Индекс функции y, определенной как y в классе Result

epsilon = 0.01

result = Result.solveByAdams(f, epsilon, a, y\_a, b)

print(result) # Ожидаемый результат около e ≈ 2.71828

Пример 3:

a = 0

b = math.pi # Пи радиан

y\_a = 1 # -cos(0) + 1 = 1

f = 1 # Индекс функции sin(x), определенной в классе Result как sin(x)

epsilon = 0.01

result = Result.solveByAdams(f, epsilon, a, y\_a, b)

print(result) # Ожидаемый результат около 2, поскольку y(pi) = -(-1) + 1 = 2

Пример 4:

a = 0

b = 2

y\_a = 1 # Начальное значение y при x = 0

f = 4 # Индекс функции -2x + 4, требуется добавить в класс Result

epsilon = 0.01

result = Result.solveByAdams(f, epsilon, a, y\_a, b)

print(result) # Ожидаемый результат около 5, поскольку y(2) = -(2^2) + 4\*2 + 1 = 5

Пример 5:

a = 0

b = 1

y\_a = 0.5 # Начальное значение y

f = 5 # Индекс функции y^2 - x^2, требуется добавить в класс Result

epsilon = 0.01

result = Result.solveByAdams(f, epsilon, a, y\_a, b)

print(result) # Результат будет зависеть от точности и стабильности численного метода

Пример 6:

a = 0

b = 5

y\_a = 0.5 # Начальное значение y при x = 0

f = 6 # Индекс функции y(1 - y), требуется добавить в класс Result

epsilon = 0.01

result = Result.solveByAdams(f, epsilon, a, y\_a, b)

print(result) # Ожидаемый результат близок к 1, поскольку популяция асимптотически приближается к 1

## Вывод

Метод Адамса, реализованный для численного решения дифференциальных уравнений, демонстрирует хорошие результаты при работе с гладкими функциями, такими как линейные или экспоненциальные уравнения, где можно легко сравнить численное решение с известным аналитическим. Однако в случаях, когда функция имеет резкие изменения или сильные нелинейности, такие как y^2 - x^2, метод может проявлять большую численную ошибку, особенно при больших значениях шага интегрирования ϵ.

По сравнению с методом Рунге-Кутта, метод Адамса эффективнее использует вычислительные ресурсы на больших интервалах за счёт использования предыдущих значений для предсказания новых. Это делает его предпочтительным выбором для задач, где требуется высокая точность на протяжении длительных интервалов интегрирования. Тем не менее, начальная фаза, требующая метода Рунге-Кутта для генерации начальных значений, может быть воспринята как недостаток в ситуациях, когда начальные условия сложны для расчёта.

В общем, метод Адамса предоставляет хороший баланс между точностью и вычислительной эффективностью, но требует тщательного выбора шага интегрирования и хорошего понимания динамики решаемой системы для минимизации численных ошибок.