­­­Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

Факультет Программной Инженерии и Компьютерной Техники

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



Лабораторная работа №6 по дисциплине  
«Вычислительная математика»

"Численное дифференцирование"

Выполнил: Сафронов Егор Михайлович

Группа: P3213

Преподаватель: Малышева Татьяна Алексеевна

г. Санкт-Петербург

2022

Цель работы:

Решить задачу Коши численными методами. Использовать метод Эйлера и Адамса

Задание лабораторной работы:

1. Программная реализация задачи:  
   a) Исходные данные: ОДУ вида , начальные условия , интервал дифференцирования [*a, b*], шаг *h*, точность .  
   b) Составить таблицу приближенных значений интеграла дифференциального уравнения, удовлетворяющего начальным условиям. Для оценки точности использовать правило Рунге.  
   c) Построить графики точного решения и полученного численного решения (разными цветами).

Рабочие формулы:

Поскольку в моем варианте использовался простой метод Эйлера, а не модифицированный, то формула получения значения на каждом шаге, следующая:Diagram

Description automatically generated with medium confidence. Она использует значение, полученное на прошлом шаге, а также использует его для вычисления значения функции f(x, y).

Для определения точности использовалось правило Рунге:Diagram, text

Description automatically generated with medium confidence. Где y^h – это значение, полученное с шагом h, а y^2h – значение полученное с шагом h/2. Р в этой формуле – порядок точности метода: 2 для Эйлера и 4 для Адамса.

Поскольку чтобы запустить метод Адамса нам надо знать первые 4 значения, я их считаю методом Рунге-Кутта следующими формулам

Diagram

Description automatically generated

Text, letter

Description automatically generated

Сам метод Адамса немного сложнее Эйлера, он использует формулу Diagram, schematic

Description automatically generated, где Text, letter

Description automatically generated

и начинается только с 5-го узла, поскольку использует предыдущие 4 шага.

Примеры работы программы:

Пример 1: y' = y + (1+x)\*y^2

Text

Description automatically generated

Chart, line chart

Description automatically generatedTable

Description automatically generatedTable

Description automatically generated

Пример 2: y' = -y + (x + 1)^3

**A picture containing calendar

Description automatically generatedChart, line chart

Description automatically generated**

**Table

Description automatically generatedTable

Description automatically generated**

Реализация расчета:

func EulerMethod() \*m.Series {  
 n = int((b-a)/h) + 1  
 euler = make([][]float64, n+1)  
 for i := 0; i < n+1; i++ {  
 euler[i] = make([]float64, 2)  
 }  
 euler[0][0] = a  
 euler[0][1] = y0  
 for i := 1; i < n; i++ {  
 previousX := euler[i-1][0]  
 previousY := euler[i-1][1]  
 currentX := previousX + h  
 currentY := previousY  
 if index == 1 {  
 currentY += h \* function\_1(currentX, previousY)  
 } else if index == 2 {  
 currentY += h \* function\_2(currentX, previousY)  
 } else if index == 3 {  
 currentY += h \* function\_3(currentX, previousY)  
 }  
 euler[i][0] = currentX  
 euler[i][1] = currentY  
 }  
 EulerMethodHalf()  
 n = int((b-a)/h) + 1  
 finalSeries := m.NewSeries()  
  
 for i := 0; i < n; i++ {  
 finalSeries.Add(m.MakeValue(euler[i][0], euler[i][1]))  
 }  
  
 return finalSeries  
}

func AdamsMethod() \*m.Series {  
 n = int((b-a)/h) + 1  
 adams = make([][]float64, n+1)  
 for i := 0; i < n+1; i++ {  
 adams[i] = make([]float64, 2)  
 }  
 functionResults = make([]float64, n+1)  
  
 adams[0][0] = a  
 adams[0][1] = y0  
  
 var k1, k2, k3, k4 float64  
 for i := 1; i < 4; i++ {  
 previousX := adams[i-1][0]  
 previousY := adams[i-1][1]  
 if index == 1 {  
 k1 = h \* function\_1(previousX, previousY)  
 k2 = h \* function\_1(previousX+h/2, previousY+k1/2)  
 k3 = h \* function\_1(previousX+h/2, previousY+k2/2)  
 k4 = h \* function\_1(previousX+h, previousY+k3)  
 } else if index == 2 {  
 k1 = h \* function\_2(previousX, previousY)  
 k2 = h \* function\_2(previousX+h/2, previousY+k1/2)  
 k3 = h \* function\_2(previousX+h/2, previousY+k2/2)  
 k4 = h \* function\_2(previousX+h, previousY+k3)  
 } else if index == 3 {  
 k1 = h \* function\_3(previousX, previousY)  
 k2 = h \* function\_3(previousX+h/2, previousY+k1/2)  
 k3 = h \* function\_3(previousX+h/2, previousY+k2/2)  
 k4 = h \* function\_3(previousX+h, previousY+k3)  
 }  
 currentX := previousX + h  
 currentY := previousY + (k1+2\*k2+2\*k3+k4)/6  
  
 adams[i][0] = currentX  
 adams[i][1] = currentY  
 if index == 1 {  
 functionResults[i] = function\_1(currentX, currentY)  
 } else if index == 2 {  
 functionResults[i] = function\_2(currentX, currentY)  
 } else if index == 3 {  
 functionResults[i] = function\_3(currentX, currentY)  
 }  
 }  
 for i := 4; i < n; i++ {  
 currentX := adams[i-1][0] + h  
 currentY := 0.0  
 if index == 1 {  
 currentY = adams[i-1][1] +  
 h\*(function\_1(adams[i-1][0], adams[i-1][1])) +  
 (h\*h\*(function\_1(adams[i-1][0], adams[i-1][1])-  
 function\_1(adams[i-2][0], adams[i-2][1])))/2 +  
 (5\*h\*h\*h\*(function\_1(adams[i-1][0], adams[i-1][1])-  
 2\*function\_1(adams[i-2][0], adams[i-2][1])+  
 function\_1(adams[i-3][0], adams[i-3][1])))/12 +  
 (3\*h\*h\*h\*h\*(function\_1(adams[i-1][0], adams[i-1][1])-  
 3\*function\_1(adams[i-2][0], adams[i-2][1])+  
 3\*function\_1(adams[i-3][0], adams[i-3][1])-  
 function\_1(adams[i-4][0], adams[i-4][1])))/8  
 } else if index == 2 {  
 currentY = adams[i-1][1] +  
 h\*(function\_2(adams[i-1][0], adams[i-1][1])) +  
 (h\*h\*(function\_2(adams[i-1][0], adams[i-1][1])-  
 function\_2(adams[i-2][0], adams[i-2][1])))/2 +  
 (5\*h\*h\*h\*(function\_2(adams[i-1][0], adams[i-1][1])-  
 2\*function\_2(adams[i-2][0], adams[i-2][1])+  
 function\_2(adams[i-3][0], adams[i-3][1])))/12 +  
 (3\*h\*h\*h\*h\*(function\_2(adams[i-1][0], adams[i-1][1])-  
 3\*function\_2(adams[i-2][0], adams[i-2][1])+  
 3\*function\_2(adams[i-3][0], adams[i-3][1])-  
 function\_2(adams[i-4][0], adams[i-4][1])))/8  
 } else if index == 3 {  
 currentY = adams[i-1][1] +  
 h\*(function\_3(adams[i-1][0], adams[i-1][1])) +  
 (h\*h\*(function\_3(adams[i-1][0], adams[i-1][1])-  
 function\_3(adams[i-2][0], adams[i-2][1])))/2 +  
 (5\*h\*h\*h\*(function\_3(adams[i-1][0], adams[i-1][1])-  
 2\*function\_3(adams[i-2][0], adams[i-2][1])+  
 function\_3(adams[i-3][0], adams[i-3][1])))/12 +  
 (3\*h\*h\*h\*h\*(function\_3(adams[i-1][0], adams[i-1][1])-  
 3\*function\_3(adams[i-2][0], adams[i-2][1])+  
 3\*function\_3(adams[i-3][0], adams[i-3][1])-  
 function\_3(adams[i-4][0], adams[i-4][1])))/8  
 }  
  
 adams[i][0] = currentX  
 adams[i][1] = currentY  
 }  
 AdamsMethodHalf()  
 n = int((b-a)/h) + 1  
 finalSeries := m.NewSeries()  
  
 for i := 0; i < n; i++ {  
 finalSeries.Add(m.MakeValue(adams[i][0], adams[i][1]))  
 }  
  
 return finalSeries  
}

Выводы по работе:

Реализованы различные способы численного решения задачи Коши: методы Эйлера и Адамса. Стоит отметить, что метод Адамса дает куда более точные результаты, чем базовый метод Эйлера (сравнение в примерах работы программы). Хочу заключить что метод Эйлера стоит использовать только при очень небольшом шаге h и малой длине отрезка, поскольку тогда приращения будут минимальны, и ошибка не сильно накопится. Поэтому из двух опробованных методов лучше использовать метод Адамса вместе с методом Рунге-Кутта (для расчета первых значений).