­­­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 (для поиска первых 4 элементов – использовать метод Рунге-Кутта)

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.

Для определения точности использовалось правило Рунге:Diagram, text

Description automatically generated with medium confidence.

Для определения точности с целью остановки цикла при расчете по методу Адамса – использовалось стандартное правило e<=|y\_precise – y\_calculated|

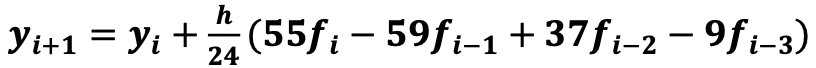
Поскольку чтобы запустить метод Адамса нам надо знать первые 4 значения, я их считаю методом Рунге-Кутта 4 порядка по формулам

Diagram

Description automatically generated

Text, letter

Description automatically generated

Сам метод Адамса несколько сложнее, он использует систему предиктор-корректор для достижения наибольшей точности. Предиктор, «основа» будущего значения считается как , а корректор, то с помощью чего достигается точность, рассчитывается по формуле .

Примеры работы программы:

Chart, 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)  
 }  
 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)  
 }  
 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)  
 }  
 }  
 for i := 4; i < n; i++ {  
 currentX := adams[i-1][0] + h  
 y\_predictor := adams[i-1][1] + h\*(55\*functionResults[i-1]-59\*functionResults[i-2]+37\*functionResults[i-3]-9\*functionResults[i-4])/24  
  
 if index == 1 {  
 functionResults[i] = function\_1(currentX, y\_predictor)  
 } else if index == 2 {  
 functionResults[i] = function\_2(currentX, y\_predictor)  
 }  
  
 y\_corrector := adams[i-1][1] + h\*(9\*functionResults[i]+19\*functionResults[i-1]-5\*functionResults[i-2]+functionResults[i-3])/24  
 for precision < math.Abs(y\_corrector-y\_predictor) {  
 y\_predictor = y\_corrector  
 if index == 1 {  
 functionResults[i] = function\_1(currentX, y\_predictor)  
 } else if index == 2 {  
 functionResults[i] = function\_2(currentX, y\_predictor)  
 }  
 y\_corrector = adams[i-1][1] + h\*(9\*functionResults[i]+19\*functionResults[i-1]-5\*functionResults[i-2]+functionResults[i-3])/24  
 }  
  
 adams[i][0] = currentX  
 adams[i][1] = y\_corrector  
 }  
 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 (подробное сравнения в примерах работы программы). Хочу заключить что метод Эйлера стоит использовать только при очень небольшом шаге h, поскольку тогда приращения будут минимальны и потери в методе будут меньше. В целом, лучше всего использовать метод Адамса вместе с методом Рунге-Кутта (для расчета первых значений).