Университет ИТМО

Мегафакультет компьютерных технологий и управления

Факультет программной инженерии и компьютерной техники

Изображение выглядит как Шрифт, текст, белый, Графика

Автоматически созданное описание

ЛАБОРАТОРНАЯ РАБОТА №6

ЧИСЛЕННОЕ РЕШЕНИЕ ОБЫКНОВЕННЫХ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ

Вариант №10

Группа: P3211

Студент: Орчиков Даниил Валерьевич

Преподаватель: Малышева Татьяна Алексеевна

г. Санкт-Петербург

2024

Оглавление

[Цель работы 3](#_Toc168266954)

[Программная реализация задачи 3](#_Toc168266955)

[Листинг программы 3](#_Toc168266956)

[Примеры и результаты работы программы 5](#_Toc168266957)

[Пример 1 5](#_Toc168266958)

[Пример 2 5](#_Toc168266959)

[Пример 3 7](#_Toc168266960)

[Вывод 7](#_Toc168266961)

Цель работы

Решить задачу Коши для обыкновенных дифференциальных уравнений численными методами.

# Программная реализация задачи

## Листинг программы

Представлен только код, непосредственно выполняющий вычисления

Весь код можно посмотреть [тут (GitHub)](https://github.com/DaniilOrchikov/Computational_mathematics/tree/master/laba6)

function Euler\_step(x0, y0, b, h) {  
 let xi = x0  
 let yi = y0  
 let res = []  
 res.push([xi, yi])  
 for (let i = 0; i < *Math*.floor((b - x0) / h); i++) {  
 yi = yi + h \* *f*(xi, yi)  
 xi += h  
 res.push([xi, yi])  
 }  
 return res  
}  
  
function Euler(x0, y0, b, epsilon) {  
 let v = 0  
 let current\_h = *h* let first\_step = Euler\_step(x0, y0, b, current\_h)  
 current\_h /= 2  
 let second\_step = Euler\_step(x0, y0, b, current\_h)  
 while (*Math*.abs(first\_step[first\_step.length - 1][1] - second\_step[second\_step.length - 1][1]) / (2 - 1) > epsilon && v < 10) {  
 first\_step = second\_step  
 current\_h /= 2  
 second\_step = Euler\_step(x0, y0, b, current\_h)  
 v ++  
 }  
 *console*.log("Эйлер: " + current\_h)  
 return second\_step  
}  
  
function Runge\_Kutta\_step(x0, y0, b, h) {  
 let xi = x0  
 let yi = y0  
 let res = []  
 res.push([xi, yi])  
 for (let i = 0; i < *Math*.floor((b - x0) / h + 0.1); i++) {  
 let k1 = h \* *f*(xi, yi)  
 let k2 = h \* *f*(xi + h / 2, yi + k1 / 2)  
 let k3 = h \* *f*(xi + h / 2, yi + k2 / 2)  
 let k4 = h \* *f*(xi + h, yi + k3)  
 yi = yi + 1 / 6 \* (k1 + 2 \* k2 + 2 \* k3 + k4)  
 xi += h  
 res.push([xi, yi])  
 }  
 return res  
}  
  
function Runge\_Kutta(x0, y0, b, epsilon){  
 let v = 0  
 let current\_h = *h* let first\_step = Runge\_Kutta\_step(x0, y0, b, current\_h)  
 current\_h /= 2  
 let second\_step = Runge\_Kutta\_step(x0, y0, b, current\_h)  
 while (*Math*.abs(first\_step[first\_step.length - 1][1] - second\_step[second\_step.length - 1][1]) / (2 \*\* 4 - 1) > epsilon && v < 10) {  
 first\_step = second\_step  
 current\_h /= 2  
 second\_step = Runge\_Kutta\_step(x0, y0, b, current\_h)  
 v ++  
 }  
 *console*.log("Рунге-Кутт: " + current\_h)  
 return second\_step  
}  
  
  
function Milne\_step(x0, y0, b, epsilon, h) {  
 let res = Runge\_Kutta\_step(x0, y0, x0 + h \* 4, h).slice(0, 4)  
 let xi = x0 + h \* 4  
 for (let i = 3; i < *Math*.floor((b - x0) / h); i++) {  
 let yi\_p = res[i - 3][1] + 4 \* h / 3 \* (2 \* *f*(res[i - 2][0], res[i - 2][1]) - *f*(res[i - 1][0], res[i - 1][1]) + 2 \* *f*(res[i][0], res[i][1]))  
 let yi\_c = res[i - 1][1] + h / 3 \* (*f*(res[i - 1][0], res[i - 1][1]) + 4 \* *f*(res[i][0], res[i][1]) + *f*(xi, yi\_p))  
 while (*Math*.abs(yi\_p - yi\_c) > epsilon) {  
 yi\_p = yi\_c  
 yi\_c = res[i - 1][1] + h / 3 \* (*f*(res[i - 1][0], res[i - 1][1]) + 4 \* *f*(res[i][0], res[i][1]) + *f*(xi, yi\_p))  
 }  
 res.push([xi, yi\_c])  
 xi += h  
 }  
 return res  
}  
function Milne(x0, y0, b, epsilon){  
 let v = 0  
 let current\_h = *h* let first\_step = Milne\_step(x0, y0, b,epsilon, current\_h)  
 current\_h /= 2  
 let second\_step = Milne\_step(x0, y0, b,epsilon, current\_h)  
 while (*Math*.max.apply(null, first\_step.map((x,i)=>{*Math*.abs(x - second\_step[i])})) > epsilon && v < 10) {  
 first\_step = second\_step  
 current\_h /= 2  
 second\_step = Milne\_step(x0, y0, b,epsilon, current\_h)  
 v++  
 }  
 *console*.log("Милн: " + current\_h)  
 return second\_step  
}

# Примеры и результаты работы программы

## Пример 1

Изображение выглядит как текст, График, линия, диаграмма

Автоматически созданное описание Изображение выглядит как текст, диаграмма, Параллельный, снимок экрана

Автоматически созданное описание Изображение выглядит как текст, снимок экрана, чек, Шрифт

Автоматически созданное описание

## Пример 2

Изображение выглядит как текст, снимок экрана, График, диаграмма

Автоматически созданное описание Изображение выглядит как текст, диаграмма, черно-белый, чек

Автоматически созданное описание

## Пример 3 Изображение выглядит как текст, График, диаграмма, линия Автоматически созданное описание Изображение выглядит как текст, снимок экрана, Шрифт, диаграмма Автоматически созданное описание

# Вывод

Во время выполнения данной лабораторной работы я познакомился с различными численными методами решения задачи Коши для обыкновенных дифф. уравнений и написал реализацию некоторых из них на языке JavaScript.