**Санкт-Петербургский Национальный Исследовательский Университет ИТМО**

**Лабораторная работа №5**

по дисциплине

*«Вычислительная математика»*

Выполнил: Анисимов М. Д.

Группа: Р3233

Преподаватель: Перл О. В.

Санкт-Петербург

2024 г.

**Цель работы**

Разработать программу, которая вычисляет частное решение дифференциального уравнения через улучшенный метод Эйлера. В качеств входных данных программа принимает номер функции, которую необходимо продифференцировать, границы интервала [a, b], приближение epsilon, и значение функции y(a). В качестве выходных данных программа вычисляет значение функции y(b), с разницей, не превышающей значение epsilon

**Описание метода**

Метод основан на классическом методе Эйлера. Отличие обычного метода от усовершенствованного заключается в точности результата. Функция делится на маленькие «ломанные» линии. Тем меньше шаг между линиями, тем более точное вычисление функции. В программе первым идёт вычисление классического метода Эйлера по формуле . Далее следует вычисление среднего значения производной функции по формуле . Это необходимо для более точного вычисления улучшенного метода Эйлера. Далее происходит вычисление самого улучшенного метода Эйлера по формуле . После необходимо вычислить следующий шаг h для дальнейших вычислений. Для этого нужно вычислить величину ошибки e по формуле между двумя методами Эйлера и подставить в формулу вычисления h:

, где – точность вычисления

.

**Код программы**

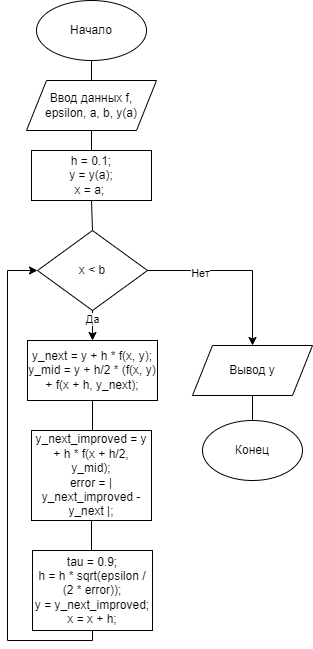
class Result {  
  
 private static double first\_function(double x, double y) {  
 return *sin*(x);  
 }  
  
 private static double second\_function(double x, double y) {  
 return (x \* y)/2;  
 }  
  
 private static double third\_function(double x, double y) {  
 return y - (2 \* x)/y;  
 }  
  
 private static double fourth\_function(double x, double y) {  
 return x + y;  
 }  
  
 private static double default\_function(double x, double y) {  
 return 0.0;  
 }  
  
 private static BiFunction<Double, Double, Double> get\_function(int n) {  
 switch (n) {  
 case (1):  
 return Result::*first\_function*;  
 case (2):  
 return Result::*second\_function*;  
 case (3):  
 return Result::*third\_function*;  
 case (4):  
 return Result::*fourth\_function*;  
 default:  
 return Result::*default\_function*;  
 }  
 }  
  
 public static double solveByEulerImproved(int f, double epsilon, double a, double y\_a, double b) {  
 double h = 0.1;  
 double y = y\_a;  
 double x = a;  
  
 BiFunction<Double, Double, Double> func = *get\_function*(f);  
  
 while (x < b) {  
 double y\_next = y + h \* func.apply(x, y);  
  
 double y\_mid = y + h/2 \* (func.apply(x, y) + func.apply(x + h, y\_next));  
 double y\_next\_improved = y + h \* func.apply(x + h/2, y\_mid);  
  
 double error = Math.*abs*(y\_next\_improved - y\_next);  
 double tau = 0.9;  
 h = h \* Math.*sqrt*(epsilon / (2 \* error));  
  
 y = y\_next\_improved;  
 x += h;  
 }  
 return y;  
 }  
}

**Пример работы программы**

|  |  |
| --- | --- |
| Пример №1 |  |
| Пример №2 |  |
| Пример №3 |  |
| Пример №4 |  |
| Пример №5 |  |
| Пример №6 |  |
| Примечания: | Программа вычисляет ответ в том случае, когда не возникает арифметических ошибок. В примере №4 программа выдала ответ NaN, так как произошло деление на ноль, и-за чего невозможно высчитать ответ. В случае, когда пользователь вводит номер уравнения, которого нет в списке по условию – программа выдаёт в качестве ответа значение y(a), так как остальные части уравнений при вычислении сводятся к нулю |

**Блок-схема программы**

**Вывод**

Программа работает в том случае, когда не возникает ошибок при вычислении метода Эйлера. Если, как пример, в вычислениях в знаменателе будет 0, то программа выдаст NaN. Точность самого вычисления зависит от параметра epsilon, который пользователь вводит сам.

В сравнении со стандартным методом Эйлера улучшенный метод является более точным. Что касается сравнения с методом Рунге-Кутты второго и четвёртого порядка, то Эйлера более прост в реализации, менее вычислительно затратный. С другой стороны Эйлера менее точен и менее стабилен по сравнению со своими аналогами, так же имеет не такой широкий диапазон применимости. Тоже самое касается сравнения улучшенного метода Эйлера с методом Хемминга – первый метод более прост в реализации, но менее точен, однако Хемминга не требует вычисления второй производной и имеет высокий порядок точности.

Соответственно метод применим в том случае, когда нам важна простота в реализации алгоритма, а точность вычислений или их стабильность отходит на второй план. Алгоритмическая сложность метода: O(1/2), где – заданная точность

Численная ошибка метода зависит от параметра эпсилон и от шага, на который мы производим вычисление. В моей программе численная ошибка зависит от того, какая разница в вычислениях между стандартным и улучшенным методом Эйлера, так как этот параметр и нужен для вычисления шага h