Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное  
учреждение высшего образования

Новосибирский государственный технический университет

Кафедра Автоматизированных систем управления

**Отчет по лабораторной работе № 6**

по дисциплине «Вычислительная математика»

Выполнили:

Букова Анна

Ланг Татьяна

Пустовских Дмитрий

Группа: АВТ-813

Преподаватель:

Лукоянычев

Алексей Викторович

Новосибирск

2020

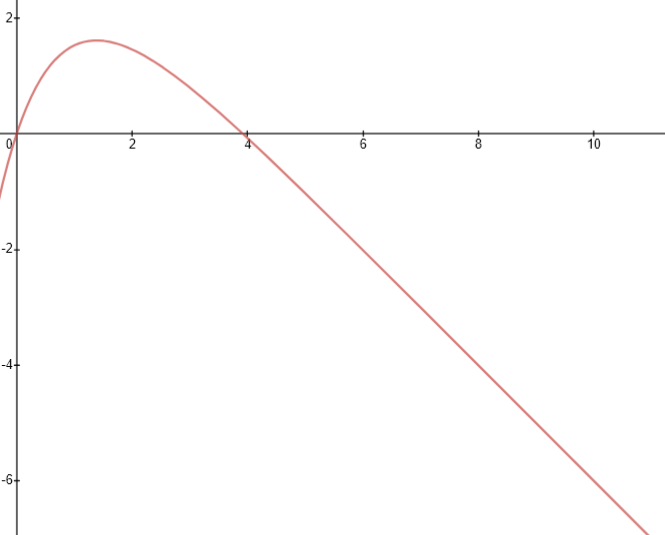
1. **Исходные данные**

Функция:

Начальные условия : (0;0)

Точное решение:

График:



1. **Результаты вычислений**

График, полученный методом Эйлера:

|  |  |
| --- | --- |
| h= 0,05 |  |
| h= 0,5 |  |
| h= 1 |  |

График, полученный методом Рунге–Кутты Мерсона:

|  |  |
| --- | --- |
| e = 0,01 |  |
| e = 0,001 |  |
| e = 0,1 |  |

График, полученный методом исправленного Эйлера:

|  |  |
| --- | --- |
| h= 0,05 |  |
| h= 0,5 |  |
| h= 1 |  |

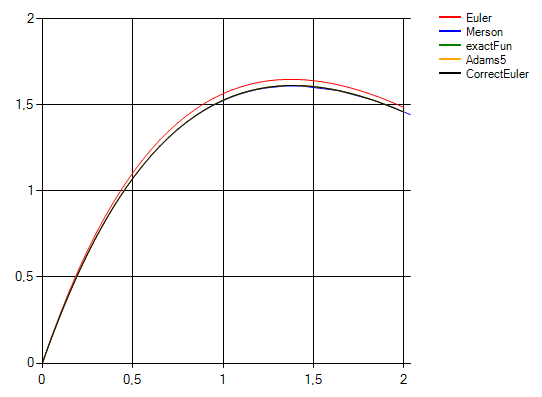
График, полученный методом Адамса 5-го порядка:

|  |  |
| --- | --- |
| h= 0,05 |  |
| h= 0,1 |  |
| h= 1 |  |

1. **Выводы**

В данной лабораторной работе мы познакомились с различными методами численного решения обыкновенных дифференциальных уравнений. Также практически реализовали методы на языке С#. При построении всех графиков вместе, можно увидеть, что только метод Эйлера значительно отличается от точного решения, а остальные методы отличаются от точного на очень маленькое значение, если принанять шаг h за очень маленькое значение.

При проведении работы можно заметить, что при увеличении шага в методе Адамса 5-го порядка, график начинает искажаться, можно предположить, что это заключается в том, что при увеличении шага методы выдают значения с ошибкой, а так как данный метод требует, чтобы перед началом решения были известны 5 значений данной дифуры. Учитывая то, что мы еще складываем значения то ошибка, увеличивается. Так же, если мы возьмём точные значения в качестве начальных, то при подсчете, новое значения тоже получится с ошибкой.



**Тексты программ**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace lab\_6

{

public class Differential

{

private Func<double, double, double> func;

private double x0;

private double y0;

public double exactFun(double x)

{

return 4 - x - 4 \* Math.Exp(-x);

}

public Differential(Func<double, double, double> foo, double x, double y)

{

func = new Func<double, double, double>(foo);

x0 = x;

y0 = y;

}

public double[][] Euler(double h, double xn)

{

int countPoin = Convert.ToInt32((xn - x0) / h) + 1;

double[][] result = new double[2][];

result[0] = new double[countPoin];

result[1] = new double[countPoin];

result[0][0] = x0;

result[1][0] = y0;

for (int i = 1; i < countPoin; ++i)

{

result[0][i] = result[0][i - 1] + h;

result[1][i] = result[1][i - 1] + h \* func(result[1][i - 1], result[0][i - 1]);

}

return result;

}

public double[][] CorrectedEuler(double h, double xn)

{

int countPoin = Convert.ToInt32((xn - x0) / h) + 1;

double[][] result = new double[2][];

result[0] = new double[countPoin];

result[1] = new double[countPoin];

result[0][0] = x0;

result[1][0] = y0;

for (int i = 1; i < countPoin; ++i)

{

double temp;

double temp2;

result[0][i] = result[0][i - 1] + h;

temp = func(result[1][i - 1], result[0][i - 1]);

temp2 = result[1][i - 1] + h \* temp;

result[1][i] = result[1][i - 1] + (h / 2) \* (temp + func(result[0][i], temp2));

}

return result;

}

public double[][] Merson(double h, double xn, double e)

{

double[][] result = new double[2][];

List<double> y = new List<double>();

List<double> x = new List<double>();

int ind = 0;

int iteration = 0;

x.Add(x0);

y.Add(y0);

while (x[ind] < xn)

{

double k1, k2, k3, k4, k5, d, yn;

while (true)

{

k1 = h \* func(x[ind], y[ind]);

k2 = h \* func(x[ind] + (h / 3), y[ind] + (k1 / 3));

k3 = h \* func(x[ind] + (h / 3), y[ind] + (k1 / 6) + (k2 / 6));

k4 = h \* func(x[ind] + (h / 2), y[ind] + (k1 / 8) + 3 \* (k3 / 8));

k5 = h \* func(x[ind] + h, y[ind] + (k1 / 2) - 3 \* (k3 / 2) + (3 \* k4));

d = (2 \* k1 - 9 \* k3 + 8 \* k4 - k5);

if (Math.Abs(d) < e \* Math.Abs(y[ind]) || iteration > 1/e)

{

break;

}

h /= 2;

iteration++;

}

yn = y[ind] + k1 / 6 + 2 \* (k4 / 3) + k5 / 6;

y.Add(yn);

x.Add(x[ind] + h);

if (Math.Abs(d) < (e / 32) \* Math.Abs(y[ind]))

{

h \*= 2;

}

iteration = 0;

ind++;

}

result[0] = x.ToArray();

result[1] = y.ToArray();

return result;

}

public double[][] Adams5(double h, double xn)

{

int countPoin = Convert.ToInt32((xn - x0) / h) + 1;

double[][] result = new double[2][];

double[][] beginPoint;

double[] Fn = new double[countPoin];

result[0] = new double[countPoin];

result[1] = new double[countPoin];

beginPoint = CorrectedEuler(h, x0 + h \* 4);

for(int i = 0; i < beginPoint[0].Length; ++i )

{

Fn[i] = func(beginPoint[0][i], beginPoint[1][i]);

result[0][i] = beginPoint[0][i];

result[1][i] = beginPoint[1][i];

}

for (int i = beginPoint[0].Length; i < countPoin; ++i)

{

result[0][i] = result[0][i - 1] + h;

result[1][i] = result[1][i - 1] + (h / 720) \* ((1901 \* Fn[i - 1]) - (2774 \* Fn[i - 2]) + (2616 \* Fn[i - 3]) - (1274 \* Fn[i - 4]) + (251 \* Fn[i - 5]));

Fn[i] = func(result[0][i], result[1][i]);

}

return result;

}

}

}