Министерство науки и высшего образования Российской Федерации

Новосибирский государственный технический университет

Кафедра автоматизированных систем управления



**Лабораторная работа №5**

**«Расчет переходной функции численными методами»**

|  |  |
| --- | --- |
| Группа: АВТ-813  Студент:  Чернаков Кирилл | Преподаватель:  Достовалов Дмитрий Николаевич,  к.т.н., заведующий кафедрой АСУ |

Новосибирск

2020 г.

1. **Передаточная функция**

**=**

1. **Cтруктурная схема в Matlab**

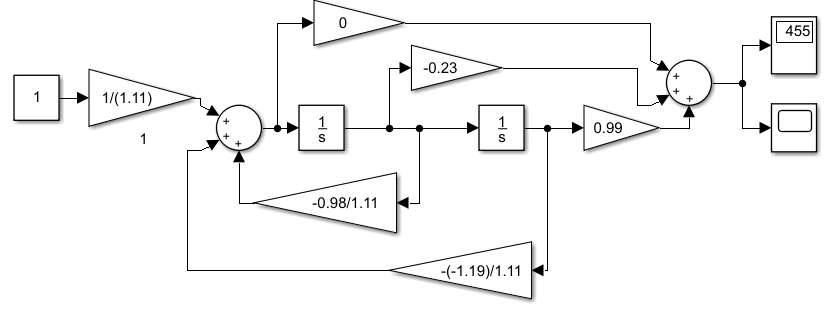
****

Рис. 1 – Структурная схема

1. **График переходной характеристики из Matlab**

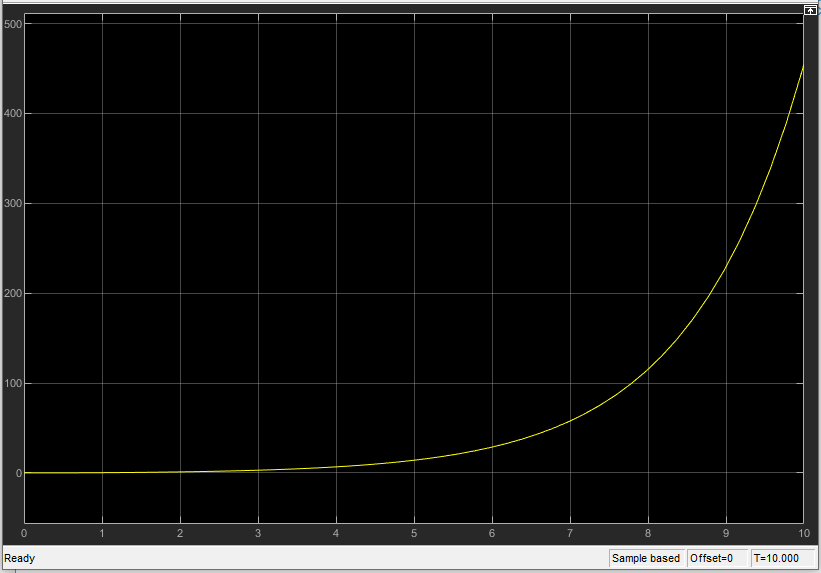
****

Рис. 2 – График переходной функции

1. **Значения y и, полученные в Matlab.**

Проверим эту систему на устойчивость, рассмотри характеристическое уравнение системы:

Найдем корни:

0.684144

Так как не все корни характеристического уравнения имеют отрицательную вещественную часть, следует что система не устойчивая и она никогда не придёт к установившемуся режиму.

Рассмотрим переходной процесс за время  **= 10**, за это время процесс придёт к значению **y = 455.97571918554**

1. **Система уравнений, используемая для расчета переходного процесса.**
2. **Значения y, полученные с помощью ваших программ.**

Полусонное значение с помощью программы(Метод Эйлера ): = -84,5074

Полусонное значение с помощью программы(Метод РунгеКутты): = -84,800

Значение полученное с помощью Matlab: **= 455.97571918554**

**Абсолютная погрешность для Эйлера:**

-

*-0,2755*

**Относительная погрешность для** **Эйлера:**

=0,3249%

**Абсолютная погрешность для Рунге-Кутты:**

-

*0,0171*

**Относительная погрешность для** **Рунге-Кутты:**

=0,02016%

**Выводы об оценке точности расчетов:**

В результатеотносительнаяпогрешность равняется 0,3249%, что является незначительной ошибкой, из этого следует правильность работы нашей программы и составленной системы уравнений. Как можно увидеть у метода Рунге-Кутты относительная погрешность меньше, чем у метода Эйлера, что говорит о высокой точности метода Рунге-Кутты.

1. **Скриншоты из программы**

****

Рис. 3 – Скриншот работы программы

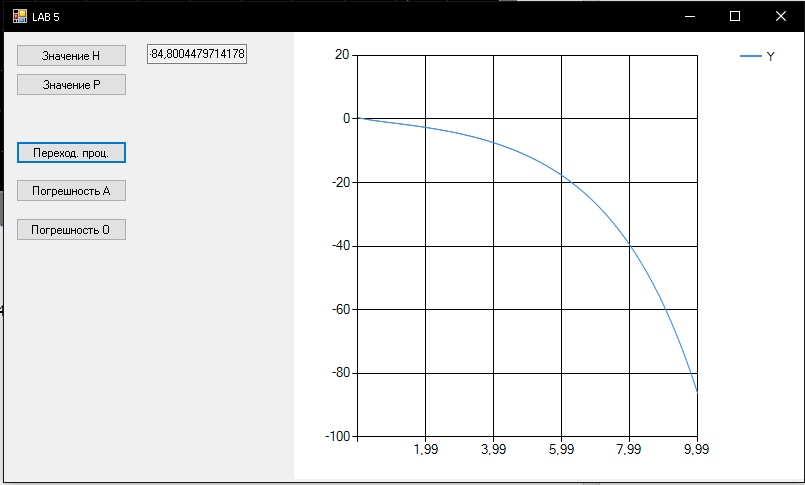


Рис. 4 – Скриншот работы программы

1. **Материалы по дополнительным заданиям**

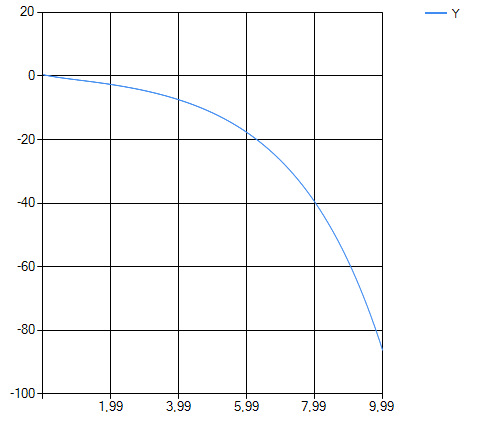
****

Рис. 5 – График переходной функции

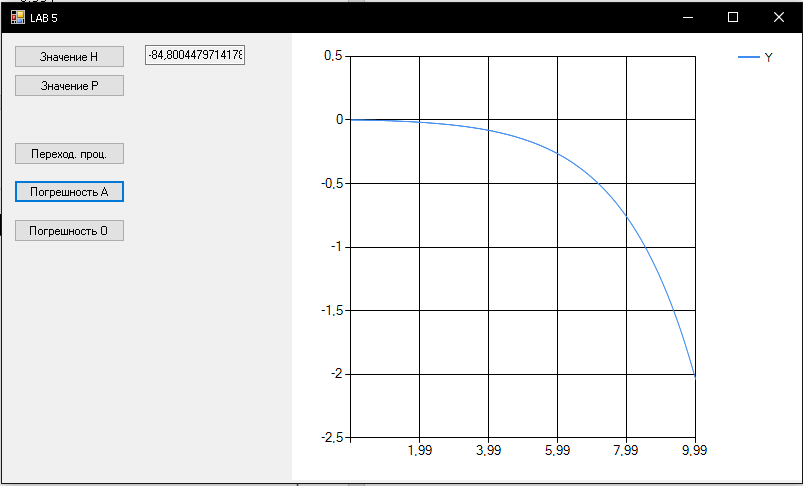
****

Рис. 6 – График абсолютной погрешности для метода Эйлера и Рунге-Кутты

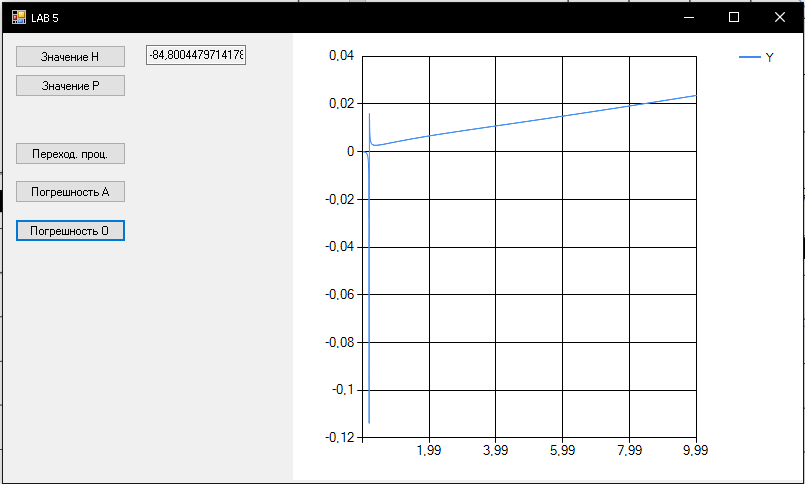
****

Рис. 7 – График относительной погрешности для метода Эйлера и Рунге-Кутты

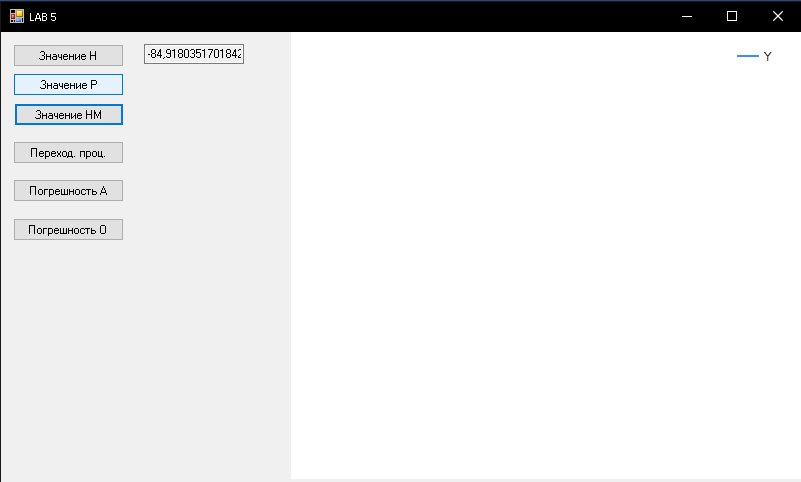
****

Рис. 8 – Скриншот работы программы

**Модифицированный метод Эйлера**

Полусонное значение с помощью программы(Метод Мод. Эйлера ): **= -84,9180**

Значение полученное с помощью Matlab: **= -84,7829**

**Абсолютная погрешность:**

-

*0,1351*

**Относительная погрешность:**

=0,1593%

**Выводы:** в результате мы добились уменьшение ошибки по сравнению с Эйлиром, путем изменения метода численного решения дифференциального уравнения.

1. **Приложение**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace OUT5lab

{

class PF

{

double a0;

double a1;

double a2;

double b0;

double b1;

double b2;

double g;

private double X3(double x1, double x2)

{

return (1 / b2) \* g - (b1 / b2) \* x2 - (b0 / b2) \* x1;

}

private double Y(double x1, double x2, double x3)

{

return a0 \* x1 + a1 \* x2 + a2 \* x3;

}

private double X2(double x3)

{

return x3;

}

private double X1(double x2)

{

return x2;

}

public PF(double a2, double a1, double a0, double b2, double b1, double b0, double g)

{

this.a0 = a0;

this.a1 = a1;

this.a2 = a2;

this.b0 = b0;

this.b1 = b1;

this.b2 = b2;

this.g = g;

}

public double getYNewton(double Xmax, double h, double x1, double x2)

{

double X\_1 = x1;

double X\_2 = x2;

double X\_3 = X3(X\_1, X\_2);

for (double i = 0; i < Xmax; i += h)

{

X\_3 = X3(X\_1, X\_2);

X\_1 = X\_1 + h \* X1(X\_2);

X\_2 = X\_2 + h \* X2(X\_3);

}

return Y(X\_1, X\_2, X\_3);

}

public double getYNodNewton(double Xmax, double h, double x1, double x2)

{

double X\_1 = x1;

double X\_2 = x2;

double X\_3 = X3(X\_1, X\_2);

for (double i = 0; i < Xmax; i += h)

{

X\_3 = X3(X\_1, X\_2);

X\_1 = X\_1 + h \* X1(X\_2 + (h / 2) \* X1(X\_2));

X\_2 = X\_2 + h \* X2(X\_3 + (h / 2) \* X2(X\_3));

}

return Y(X\_1, X\_2, X\_3);

}

public double[] getYArNewton(double Xmax, double h, double x1, double x2)

{

double X\_1 = x1;

double X\_2 = x2;

double X\_3 = X3(X\_1, X\_2);

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

for (double i = 0; i < Xmax; i += h)

{

X\_3 = X3(X\_1, X\_2);

list.Add(Y(X\_1, X\_2, X\_3));

X\_1 = X\_1 + h \* X1(X\_2);

X\_2 = X\_2 + h \* X2(X\_3);

}

return list.ToArray();

}

public double[] getYArRunge(double Xmax, double h, double x1, double x2)

{

double X\_1 = x1;

double X\_2 = x2;

double X\_3 = X3(X\_1, X\_2);

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

for (double i = 0; i < Xmax; i += h)

{

X\_3 = X3(X\_1, X\_2);

list.Add(Y(X\_1, X\_2, X\_3));

double k1, k2, k3, k4, k\_1, k\_2, k\_3, k\_4;

k1 = X1(X\_2);

k2 = X1(X\_2 + (h / 2) \* k1);

k3 = X1(X\_2 + (h / 2) \* k2);

k4 = X1(X\_2 + h \* k3);

k\_1 = X2(X\_3);

k\_2 = X2(X\_3 + (h / 2) \* k1);

k\_3 = X2(X\_3 + (h / 2) \* k2);

k\_4 = X2(X\_3 + h \* k3);

X\_1 = X\_1 + (h / 6) \* (k1 + 2 \* k2 + 2 \* k3 + k4);

X\_2 = X\_2 + (h / 6) \* (k\_1 + 2 \* k\_2 + 2 \* k\_3 + k\_4);

}

return list.ToArray();

}

public double getYRunge(double Xmax, double h, double x1, double x2)

{

double X\_1 = x1;

double X\_2 = x2;

double X\_3 = X3(X\_1, X\_2);

for (double i = 0; i < Xmax; i += h)

{

X\_3 = X3(X\_1, X\_2);

double k1, k2, k3, k4, k\_1, k\_2, k\_3, k\_4;

k1 = X1(X\_2);

k2 = X1(X\_2 + (h / 2) \* k1);

k3 = X1(X\_2 + (h / 2) \* k2);

k4 = X1(X\_2 + h \* k3);

X\_1 = X\_1 + (h / 6) \* (k1 + 2 \* k2 + 2 \* k3 + k4);

k\_1 = X2(X\_3);

k\_2 = X2(X\_3 + (h / 2) \* k1);

k\_3 = X2(X\_3 + (h / 2) \* k2);

k\_4 = X2(X\_3 + h \* k3);

X\_2 = X\_2 + (h / 6) \* (k\_1 + 2 \* k\_2 + 2 \* k\_3 + k\_4);

}

return Y(X\_1, X\_2, X\_3);

}

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace OUT5lab

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

PF du = new PF(-0.53,2.26,1.45,-1.32,-1.97,0.93,1);

this.textBox1.Text = Convert.ToString(du.getYNewton(10, 0.001 \* 10, 0, 0));

}

private void button2\_Click(object sender, EventArgs e)

{

PF du = new PF(-0.53, 2.26, 1.45, -1.32, -1.97, 0.93, 1);

double[] Y = du.getYArRunge(10, 0.001 \* 10, 0, 0);

this.chart1.Series["Y"].Points.Clear();

int j = 0;

for(double i = 0; i < 10; i += 0.01)

{

this.chart1.Series["Y"].Points.AddXY(i,Y[j]);

j++;

}

}

private void button3\_Click(object sender, EventArgs e)

{

PF du = new PF(-0.53, 2.26, 1.45, -1.32, -1.97, 0.93, 1);

double[] Y = du.getYArRunge(10, 0.001 \* 10, 0, 0);

double[] YN = du.getYArNewton(10, 0.001 \* 10, 0, 0);

this.chart1.Series["Y"].Points.Clear();

int j = 0;

for (double i = 0; i < 10; i += 0.01)

{

this.chart1.Series["Y"].Points.AddXY(i, Y[j]-YN[j]);

j++;

}

}

private void button4\_Click(object sender, EventArgs e)

{

PF du = new PF(-0.53, 2.26, 1.45, -1.32, -1.97, 0.93, 1);

double[] Y = du.getYArRunge(10, 0.001 \* 10, 0, 0);

double[] YN = du.getYArNewton(10, 0.001 \* 10, 0, 0);

this.chart1.Series["Y"].Points.Clear();

int j = 0;

for (double i = 0; i < 10; i += 0.01)

{

this.chart1.Series["Y"].Points.AddXY(i, ((Y[j] - YN[j])/ Y[j]) );

j++;

}

}

private void button5\_Click(object sender, EventArgs e)

{

PF du = new PF(-0.53, 2.26, 1.45, -1.32, -1.97, 0.93, 1);

this.textBox1.Text = Convert.ToString(du.getYRunge(10, 0.001, 0, 0));

}

}

}