

Национальный исследовательский университет ИТМО

Факультет инфокоммуникационных технологий

**Лабораторная работа 7**

Выполнил: Орел

Даниил Максимович

Группа № K3221

Проверил: Иванов С.Е.

Санкт-Петербург

2020

**Цель работы:**

* Реализовать на языке ООП С# алгоритм решения системы дифференциальных уравнений методом Рунге-Кутты четвертого порядка

**Ход работы:**

1. В программу был добавлен класс DifEquationSystem, содержащий в качестве свойства список выражений – правых частей системы уравнений вида , где :

using System.Collections.Generic;

using Expr = MathNet.Symbolics.SymbolicExpression;

namespace Lab\_07

{

class DifEquationSystem

{

public List<Expr> Functions { get; set; }

public DifEquationSystem(List<Expr> system)

{

Functions = system;

}

}

}

1. Класс RungeKutta в программе реализовывает алгоритм решения системы дифференциальных уравнений методом Рунге-Кутты четвертого порядка:

using MathNet.Symbolics;

using System;

using System.Collections.Generic;

using Expr = MathNet.Symbolics.SymbolicExpression;

namespace Lab\_07

{

class RungeKutta

{

double[] k1, k2, k3, k4;

FloatingPoint[] new\_val;

Dictionary<string, FloatingPoint> k2\_var, k3\_var, k4\_var;

Dictionary<string, FloatingPoint> currValues;

public DifEquationSystem Equation { get; set; }

public Dictionary<string, FloatingPoint> InitialСondition { get; set; }

public double[] Section { get; set; }

public int Divisions { get; set; }

public double Step { get; }

public RungeKutta

(DifEquationSystem equation, Dictionary<string, FloatingPoint> initialCondition, double[] section, int divisions)

{

Equation = equation;

InitialСondition = initialCondition;

Section = section;

Divisions = divisions;

Step = CalculateStep();

}

private double CalculateStep()

{

double step = (Section[1] - Section[0]) / Divisions;

return step;

}

public void Solve()

{

PrintInfo();

currValues = InitialСondition;

k1 = new double[Equation.Functions.Count];

k2 = new double[Equation.Functions.Count];

k3 = new double[Equation.Functions.Count];

k4 = new double[Equation.Functions.Count];

new\_val = new FloatingPoint[currValues.Count];

Console.Write("i\t");

foreach( var key in InitialСondition.Keys)

{

Console.Write($"{key}\t");

}

Console.WriteLine();

for (int div = 0; div <= Divisions; div++)

{

Console.Write($"{div}\t");

foreach (var val in currValues.Values)

Console.Write($"{val.RealValue:0.0000}\t");

//Нахождение k1 для каждого уравнения

for (int i = 0; i < Equation.Functions.Count; i++)

{

k1[i] = Step \* Equation.Functions[i].Evaluate(currValues).RealValue;

}

//Нахождение k2 для каждого уравнения

for (int i = 0; i < Equation.Functions.Count; i++)

{

int j = 0;

k2\_var = new Dictionary<string, FloatingPoint>();

foreach (var val in currValues)

{

if (val.Key == "x")

k2\_var.Add("x", Expr.Parse((val.Value.RealValue + Step / 2)

.ToString().Replace(",", ".")).RealNumberValue);

else

{

k2\_var.Add(val.Key, Expr.Parse((val.Value.RealValue + k1[j] / 2)

.ToString().Replace(",", ".")).RealNumberValue);

j++;

}

}

k2[i] = Step \* Equation.Functions[i].Evaluate(k2\_var).RealValue;

}

//Нахождение k3 для каждого уравнения

for (int i = 0; i < Equation.Functions.Count; i++)

{

int j = 0;

k3\_var = new Dictionary<string, FloatingPoint>();

foreach (var val in currValues)

{

if (val.Key == "x")

k3\_var.Add("x", Expr.Parse((val.Value.RealValue + Step / 2)

.ToString().Replace(",", ".")).RealNumberValue);

else

{

k3\_var.Add(val.Key, Expr.Parse((val.Value.RealValue+k2[j]/2)

.ToString().Replace(",", ".")).RealNumberValue);

j++;

}

}

k3[i] = Step \* Equation.Functions[i].Evaluate(k3\_var).RealValue;

}

//Нахождение k4 для каждого уравнения

for (int i = 0; i < Equation.Functions.Count; i++)

{

int j = 0;

k4\_var = new Dictionary<string, FloatingPoint>();

foreach (var val in currValues)

{

if (val.Key == "x")

k4\_var.Add("x", Expr.Parse((val.Value.RealValue + Step)

.ToString().Replace(",", ".")).RealNumberValue);

else

{

k4\_var.Add(val.Key, Expr.Parse((val.Value.RealValue + k3[j])

.ToString().Replace(",", ".")).RealNumberValue);

j++;

}

}

k4[i] = Step \* Equation.Functions[i].Evaluate(k4\_var).RealValue;

}

int k = 0;

foreach (var val in currValues.Values)

{

new\_val[k] = val;

k++;

}

//Обновление значений x, y1, y2, ...

k = 0;

currValues = new Dictionary<string, FloatingPoint>();

foreach(var val in InitialСondition)

{

if (val.Key == "x")

currValues.Add("x", Expr.Parse((new\_val[k].RealValue + Step)

.ToString().Replace(",", ".")).RealNumberValue);

else

{

currValues.Add(val.Key, Expr.Parse((new\_val[k].RealValue +

(k1[k-1] + 2\* k2[k-1] + 2 \* k3[k-1] + k4[k-1])/6)

.ToString().Replace(",", ".")).RealNumberValue);

}

k++;

}

Console.WriteLine();

}

Console.WriteLine("");

}

private void PrintInfo()

{

int i = 1;

foreach(var eq in Equation.Functions)

{

Console.WriteLine($"y{i}' = {eq.ToString()}");

i++;

}

Console.WriteLine($"Начальное условие: ");

foreach(var cond in InitialСondition)

{

if(cond.Key == "x")

Console.WriteLine($"\t{cond.Key} = {cond.Value.RealValue}");

else

Console.WriteLine($"\t{cond.Key}({InitialСondition["x"].RealValue}) = {cond.Value.RealValue}");

}

Console.WriteLine($"Рассматриваемый интервал: [{Section[0]}; {Section[1]}]");

Console.WriteLine($"Количество делений: {Divisions}");

Console.WriteLine($"Величина шага: {Step}");

Console.WriteLine("Решение методом Рунге-Кутты:\n");

}

}

}

1. В классе Program приведен пример системы уравнений для демонстрации работы программы, а также объявлен метод, который работает с известным точным решением системы уравнений и выводит числовые значения этого решения:

using System;

using System.Collections.Generic;

using MathNet.Symbolics;

using Expr = MathNet.Symbolics.SymbolicExpression;

namespace Lab\_07

{

class Program

{

static void Main(string[] args)

{

var x = Expr.Variable("x");

var y1 = Expr.Variable("y1");

var y2 = Expr.Variable("y2");

//Example 1

Dictionary<string, FloatingPoint> initialVal1 = new Dictionary<string, FloatingPoint>()

{

{ "x", 0 },

{ "y1", 3 },

{ "y2", 0 }

};

Expr func1 = (-2 \* y1 + 4 \* y2);

Expr func2 = (- y1 + 3 \* y2);

var functions = new List<Expr>();

functions.Add(func1);

functions.Add(func2);

DifEquationSystem equation1 =

new DifEquationSystem(functions);

double[] section1 = new double[] { 0, 1 };

RungeKutta rungeKutta1 =

new RungeKutta(equation1, initialVal1, section1, 10);

Func <double, double> answer1 = (4 \* (-x).Exp() - (2 \* x).Exp()).Compile("x");

Func <double, double> answer2= ((-x).Exp() - (2 \* x).Exp()).Compile("x");

var answers = new List<Func<double, double>>();

answers.Add(answer1);

answers.Add(answer2);

rungeKutta1.Solve();

PrintRealAnswer(answers, initialVal1, section1, 10);

Console.ReadKey();

}

private static void PrintRealAnswer(List<Func<double, double>> answers,

Dictionary<string, FloatingPoint> initVal, double[] section, int div)

{

double h = (section[1] - section[0]) / div;

double curr\_x = section[0];

Console.WriteLine("Точное решение:\n");

Console.Write("i\t");

foreach (var key in initVal.Keys)

{

Console.Write($"{key}\t");

}

Console.WriteLine();

for (int i = 0; i <= div; i++)

{

Console.Write($"{i}\t");

Console.Write($"{curr\_x:0.0000}\t");

foreach (var answer in answers)

Console.Write($"{answer(curr\_x):0.0000}\t");

Console.WriteLine();

curr\_x += h;

}

Console.WriteLine();

}

}

}

1. Результат работы программы:

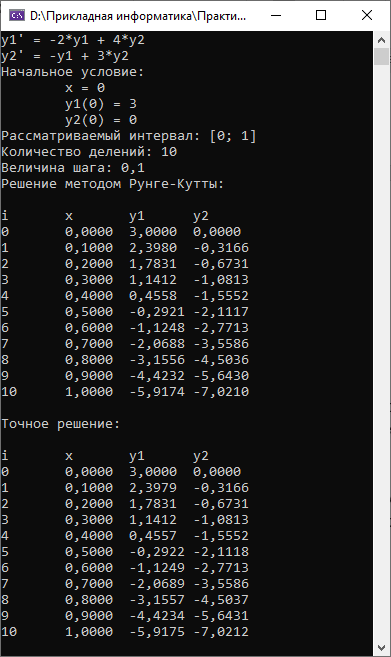


Рисунок 1 – Решение приведенной в качестве примера системы и вывод точного решения системы

**Вывод:**

В ходе выполнения данной практической работы был реализован алгоритм решения систем дифференциальных уравнений методом Рунге-Кутты четвертого порядка