Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное
учреждение высшего образования
Национальный исследовательский Нижегородский государственный
университет им. Н.И. Лобачевского

Институт информационных технологий, математики и механики

Отчет по лабораторной работе

«Численное решение задачи Коши для ОДУ»

Выполнил:
студент группы 381706-2
Зинков А. С.

Проверил:
Эгамов Альберт Исмаилович

Нижний Новгород
2020

Содержание

Введение

В лабораторной работе решается задача разработки программы поиска решения системы дифференциальных уравнений методам Рунге-Кутта.

Выбор метода решения системы дифференциальных уравнений объясняется тем, что метод Рунге-Кутта сочетает хорошую точность и высокую скорость.

Рассмотрим систему двух автономных обыкновенных дифференциальных уравнений общего вида $dx/dt = P(x, y)$; $dy/dt = Q(x, y)$ (1)

$P(x, y)$, $Q(x, y)$ – непрерывные функции, определенные в некоторой области G евклидовой плоскости и имеющие в этой области непрерывные производные порядка не ниже первого. Переменные x, y во времени изменяются в соответствие с системой уравнений, так что каждому состоянию системы соответствует пара значений переменных (x, y) Обратно, каждой паре переменных (x, y) соответствует определенное состояние системы.

Рассмотрим плоскость с осями координат, на которых отложены значения переменных x,y. Каждая точка M этой плоскости соответствует определенному состоянию системы. Такая плоскость носит название **фазовой плоскости** и изображает совокупность всех состояний системы.

Точка M(x,y) называется изображающей или представляющей точкой. Пусть в начальный момент времени t=t0 координаты изображающей точки M0(x(t0), y(t0)). В каждый следующий момент времени t изображающая точка будет смещаться в соответствии с изменениями значений переменных x(t), y(t). Совокупность точек M(x(t), y(t)) на фазовой плоскости, положение которых соответствует состояниям системы в процессе изменения во времени переменных x(t), y(t) согласно уравнениям системы, называется **фазовой траекторией.**

Совокупность фазовых траекторий при различных начальных значениях переменных дает легко обозримый "портрет" системы. Построение фазового портрета позволяет сделать выводы о характере изменений переменных x, y без знания аналитических решений исходной системы уравнений. Для изображения фазового портрета необходимо построить векторное поле направлений траекторий системы в каждой точке фазовой плоскости.

Метод Рунге-Кутты 4 порядка

Ме́тодыРу́нге — Ку́тты — большой класс численных методов решения задачи Коши для обыкновенных дифференциальных уравнений и их систем. Первые методы данного класса были предложены около 1900 года немецкими математиками К. Рунге и М. В. Куттой.

Итак, в работе был рассмотрен метод Рунге — Кутты 4го порядка, который является самым распространенным среди семейства методов, что его часто называют просто методом Рунге — Кутты.

$$\begin{cases} K_1 = f(x_k,\ y_k) \\ K_2 = f(x_k + \frac{h}{2},\ y_k + \frac{h}{2} K_1) \\ K_3 = f(x_k + \frac{h}{2},\ y_k + \frac{h}{2} K_2) \\ K_4 = f(x_k + h,\ y_k + h K_3) \\ y_{k+1} = y_k + \frac{h}{6} \cdot (K_1 + 2 K_2 + 2 K_3 + K_4) \end{cases}$$

Выбор уравнения

Для данной лабораторной работы мной была выбрано уравнение маятника с диссипацией $\ddot{x} + \delta\dot{x} + \sin x$.

При подстановке x’ = y получим систему дифференциальных уравнений :

$$\begin{cases} y = x' \\ y' = -(\sin(x) + \delta y) \end{cases}$$

Руководство пользователя

При запуске программы пользователю нужно ввести параметры ДУ, шаг и отрезок интегрирования. В поля уже введены базовые значения (рис.1).

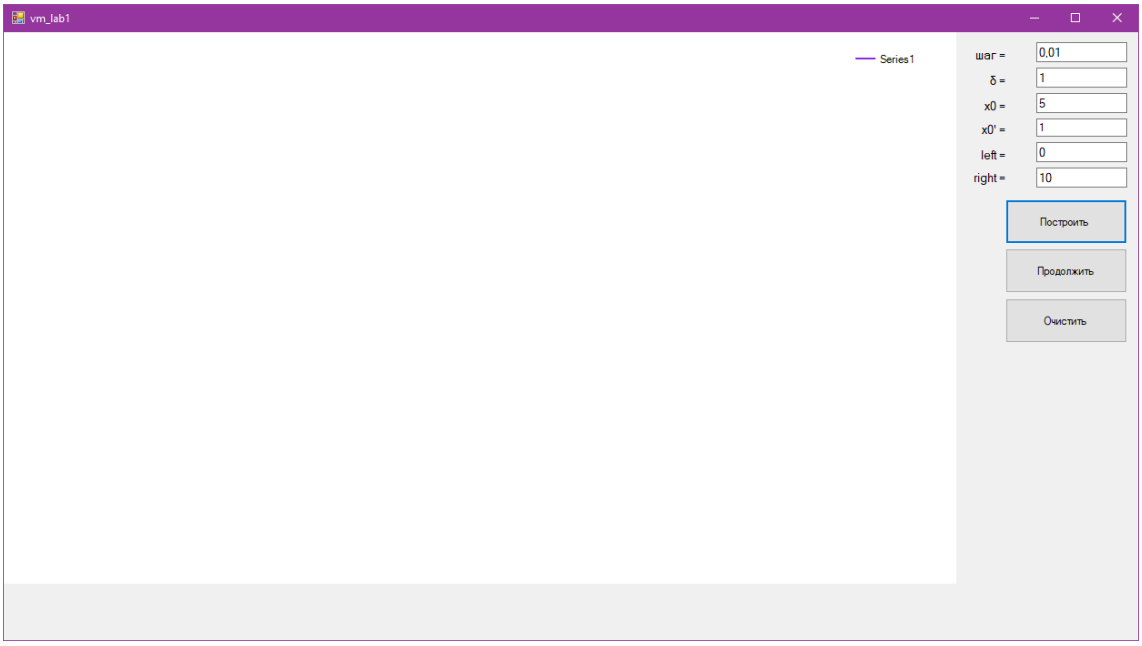

Рис.1.

Для того чтобы построить фазовую траекторию для данных начальных условий нажмите на кнопку построить (рис.2).

Рис.2.

Для того чтобы построить новую фазовую траекторию измените начальные условия и нажмите кнопку построить. Каждая новая фазовая траектория изображается другим цветом(рис.3).

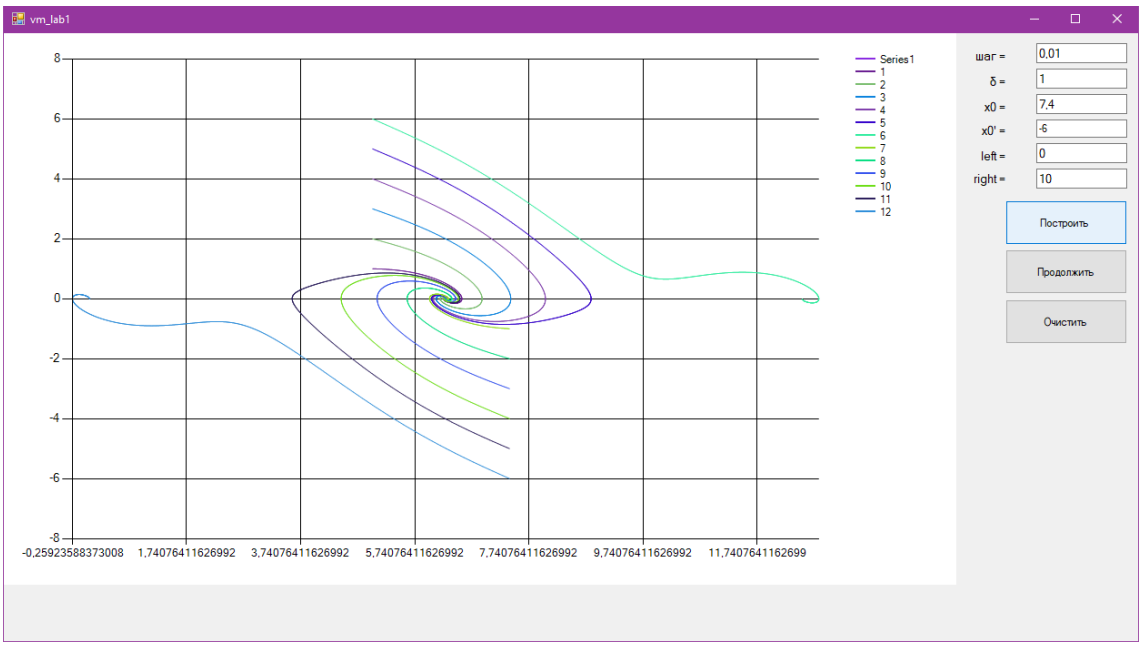

Рис.3.

Массивы значений для x и y сохраняются в отдельный файл (рис.4).

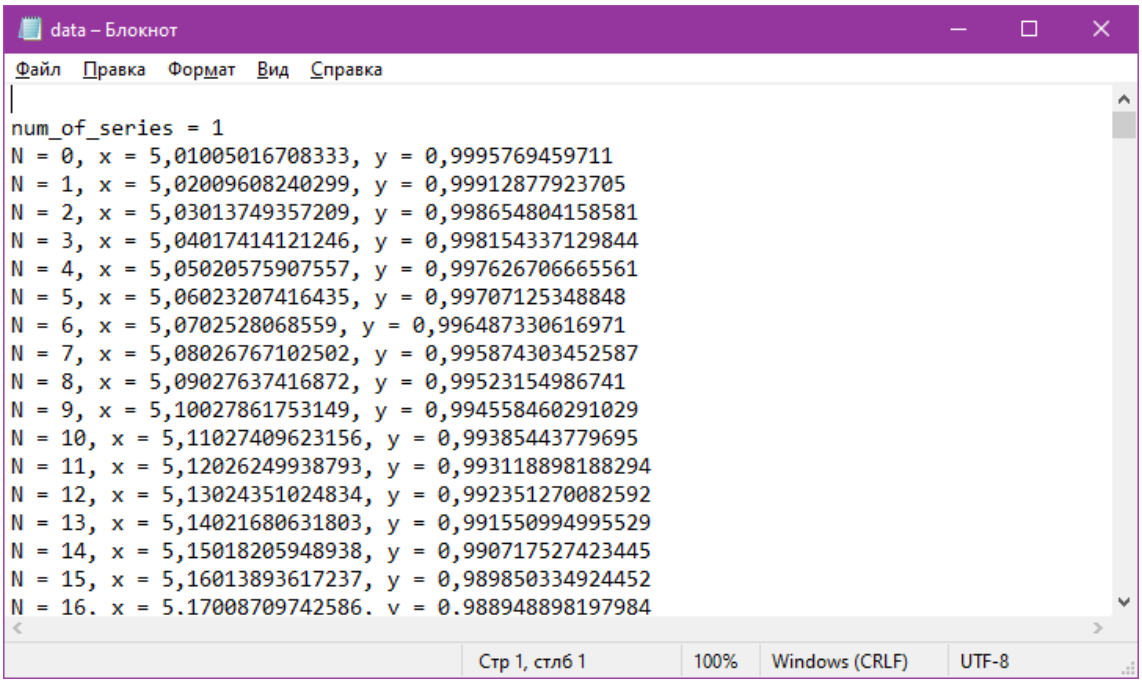

Рис.4.

Для того чтобы очистить график нажмите кнопку очистить (рис.5).

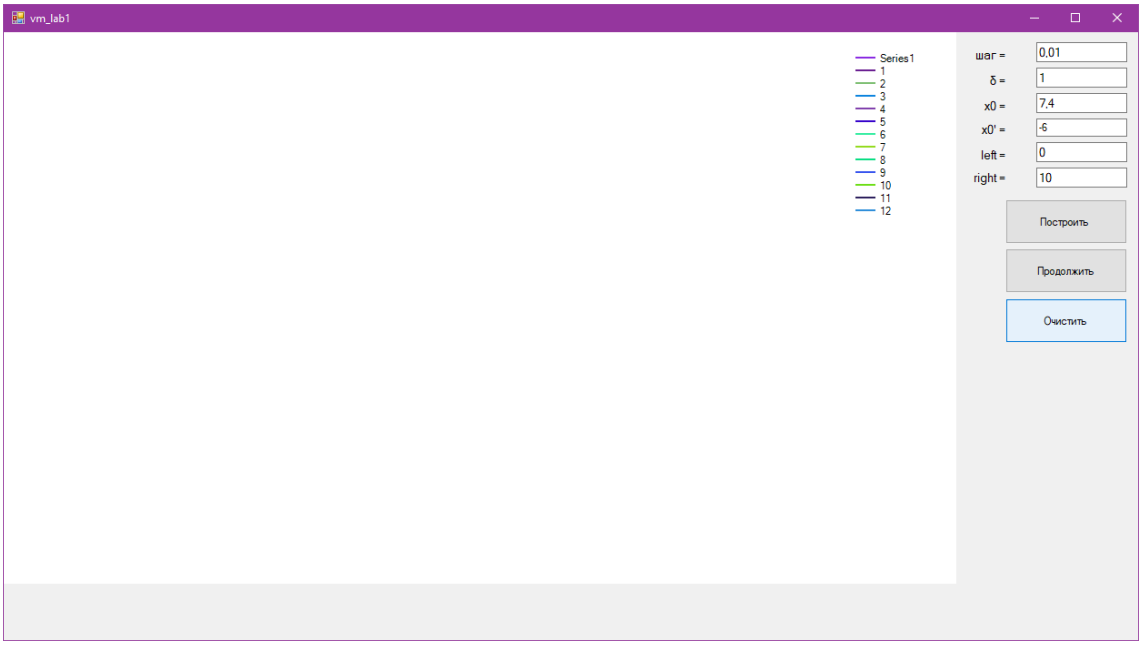

Рис.5.

Если вам нужно продолжить фазовую траекторию, нажмите кнопку продолжить (рис.6,7).

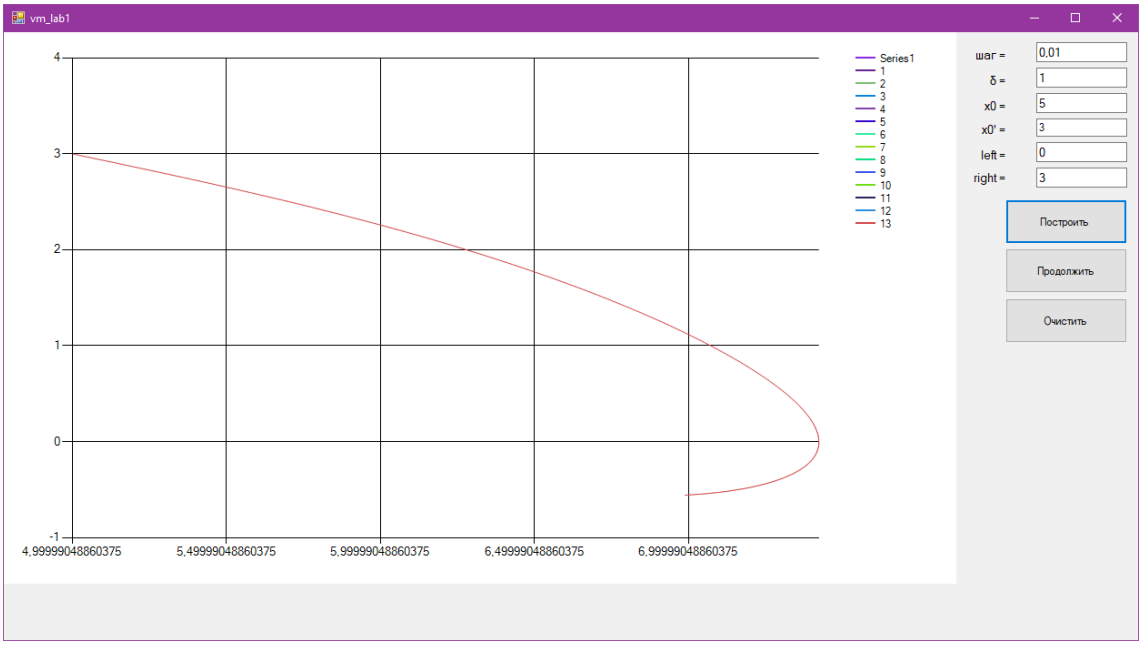

Рис.6.

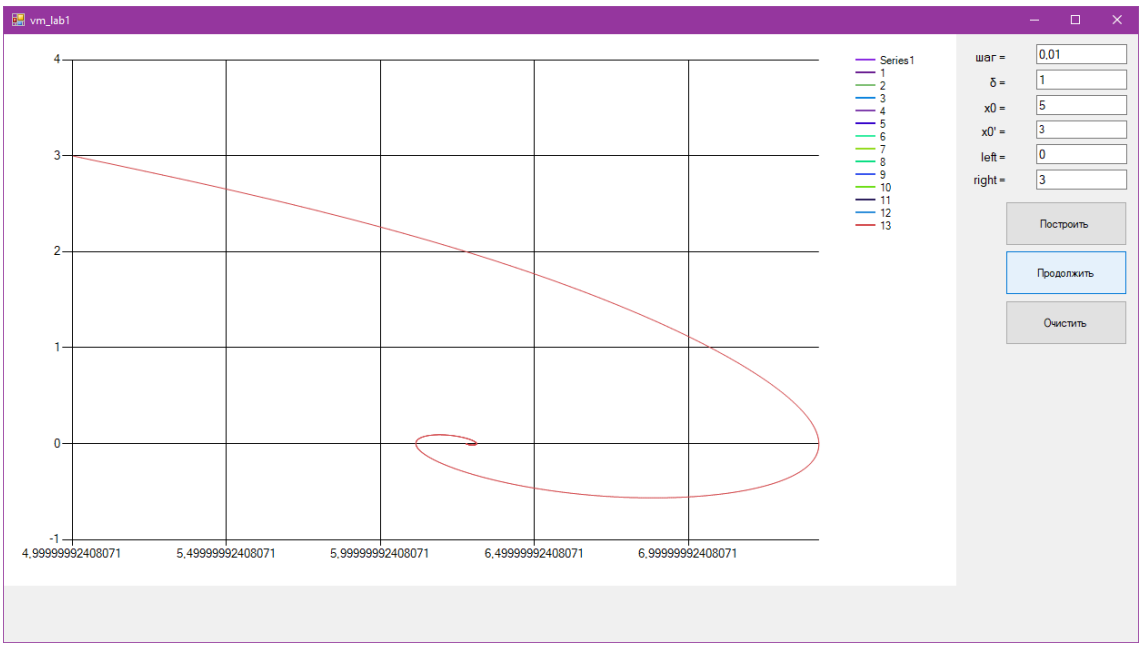
vm_lab1
4
3
2
1
0
-1
4,99999992408071
5,49999992408071
5,99999992408071
6,49999992408071
6,99999992408071
Series1
1
2
3
4
5
6
7
8
9
10
11
12
13
шаг =
0,01
δ =
1
x0 =
5
x0' =
3
left =
0
right =
3
Построить
Продолжить
Очистить

Рис.7.

Руководство программиста

Программа написана на языке C# с помощью windows forms.

В файле Page1.cs реализованы три метода:

В методе build реализовано построение фазовой траектории, а так же запись значений в отдельный файл.

```
public static void build(double h, double x0, double y0, double delta, int right, List<double>
res_x, List<double> res_y, StreamWriter sw)

       {
         res_x.Add(x0);
         res_y.Add(y0);
         int i = 0;
         double k = 0;
         while (k < right)
         {
               double k1_x = h * res_y[i];
               double k1_y = h * func(res_x[i], res_y[i], delta);
               double k2_x = h * (res_y[i] + k1_x / 2);
               double k2_y = h * func(res_x[i] + k1_x / 2, res_y[i] + k1_y / 2, delta);
               double k3_x = h * (res_y[i] + k2_x / 2);
               double k3_y = h * func(res_x[i] + k2_x / 2, res_y[i] + k2_y / 2, delta);
               double k4_x = h * (res_y[i] + k3_x);
               double k4_y = h * func(res_x[i] + k3_x, res_y[i] + k3_y, delta);

               double xt = res_x[i] + (k1_x + 2 * k2_x + 2 * k3_x + k4_x) / 6;
               res_x.Add(xt);

               double yt = res_y[i] + (k1_y + 2 * k2_y + 2 * k3_y + k4_y) / 6;
               res_y.Add(yt);

               sw.WriteLine($"N = {i}, x = {xt}, y = {yt}");
               ++i;
               k += h;
         }
       }
```

В методе extend реализовано продолжение построения фазовой траектории, а так же запись значений в отдельный файл.

```
       public static void extend(double h, double delta, int n, List<double> res_x, List<double>
res_y, StreamWriter sw) {
         int i = res_x.Count - 1;
         double k = 0;
         while (k < 10) {
               double k1_x = h * res_y[i];
               double k1_y = h * func(res_x[i], res_y[i], delta);
               double k2_x = h * (res_y[i] + k1_x / 2);
               double k2_y = h * func(res_x[i] + k1_x / 2, res_y[i] + k1_y / 2, delta);
               double k3_x = h * (res_y[i] + k2_x / 2);
               double k3_y = h * func(res_x[i] + k2_x / 2, res_y[i] + k2_y / 2, delta);
               double k4_x = h * (res_y[i] + k3_x);
               double k4_y = h * func(res_x[i] + k3_x, res_y[i] + k3_y, delta);

               double xt = res_x[i] + (k1_x + 2 * k2_x + 2 * k3_x + k4_x) / 6;
               res_x.Add(xt);

               double yt = res_y[i] + (k1_y + 2 * k2_y + 2 * k3_y + k4_y) / 6;
               res_y.Add(yt);

               sw.WriteLine($"N = {i}, x = {xt}, y = {yt}");
               ++i;
               k += h;
         }
       }
```

Метод func – вспомогательный.

```
static double func(double x, double y, double delta) {
                return -delta * y - Math.Sin(x);
        }
```

Файл Form1.cs содержит три метода:

Метод button1_click срабатывает при нажатии на кнопку “Построить”. Вызывает функцию для построения фазовой траектории.

```
private void button1_Click(object sender, EventArgs e)
{
  double h = Convert.ToDouble(this.h.Text);
  int right = Convert.ToInt32(this.right.Text);
  int left = Convert.ToInt32(this.left.Text);
  double delta = Convert.ToDouble(this.delta.Text);
  double x0 = Convert.ToDouble(this.x0.Text);
  double dx = Convert.ToDouble(this.dx.Text);

  Random random = new Random();
  this.chart1.Series.Add(Convert.ToString(++num_of_series));
  this.chart1.Series[num_of_series].ChartType =
       System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Line;
  this.chart1.Series[num_of_series].Color = Color.FromArgb(random.Next(255),
       random.Next(255), random.Next(255));
  if (num_of_series == 1)
  {
       res_x = new List<double>();
       res_y = new List<double>();
  }
  else
  {
       res_x.Clear();
       res_y.Clear();
  }

  sw.WriteLine($" ");
  sw.WriteLine($"num_of_series = {num_of_series}");
  vm_metod.build(h, x0, dx, delta, right, res_x, res_y, sw);
  int size = (int)(left / h);
  for (int i = size; i < res_x.Count; i++)
  {
       this.chart1.Series[num_of_series].Points.AddXY(res_x[i], res_y[i]);
  }
}
```

Метод button2_click срабатывает при нажатии на кнопку “Очистить”. Очищает график.

```
private void button2_Click(object sender, EventArgs e)
{
  for (int i = 0; i < num_of_series + 1; ++i)
  {
       this.chart1.Series[i].Points.Clear();
  }
  sw.Close();
  sw = new StreamWriter("data.txt", false, System.Text.Encoding.Default);
}
```

Метод button3_click срабатывает при нажатии на кнопку “Продолжить”. Вызывает функцию для продолжения построения фазовой траектории.

```
private void button3_Click(object sender, EventArgs e)
{
  double h = Convert.ToDouble(this.h.Text);
  int n = Convert.ToInt32(this.left.Text);
  double delta = Convert.ToDouble(this.delta.Text);
  int size = res_x.Count - 1;
  vm_metod.extend(h, delta, n, res_x, res_y, sw);

  for (int i = size; i < res_x.Count; i++)
  {
       this.chart1.Series[num_of_series].Points.AddXY(res_x[i], res_y[i]);
  }
}
```

Заключение

В результате работы была написана программа на языке C# с графическим интерфейсом, которая позволила построить фазовую траекторию для уравнения второго порядка - маятника с диссипацией $\ddot{x} + \delta\dot{x} + \sin x$.

Программа позволяет строить фазовую траекторию с заданными параметрами ДУ, шагом и отрезком интегрирования.

Литература

- Самарский А. А., Гулин А. В. Численные методы: Учеб.пособие для вузов. – М.: Наука. Главная редакция физикоматематической литературы.
- ЯцекГаловиц C++. Стандартная библиотека шаблонов – Питер.2018. – 432 с.
- http://mathprofi.ru/metody_eilera_i_runge_kutty.html