Федеральное государственное автономное образовательное
учреждение высшего образования
Национальный исследовательский
университет ИТМО

Факультет программной инженерии и компьютерной техники

Направление подготовки:

Системное и Прикладное Программное Обеспечение

(09.03.04 Програмная инженерия)

Дисциплина «Вычислительная математика»

Отчет

По лабораторной работе №6

Вариант №10

Студент

Карташев Владимир Сергеевич,
группа P3215

Преподаватель / Практик

Малышева Татьяна Алексеевна

г. Санкт-Петербург, 2024 г.

Цель работы

Решить задачу Коши для обыкновенных дифференциальных уравнений численными методами.

Описание метода, расчетные формулы

Метод Эйлера

Метод Э й л е р а (1707–1783) основан на разложении искомой функции $y(x)$в ряд Тейлора в окрестностях узлов $x = x_i$ $(i = 0, 1, \dots)$, в котором отбрасываются все члены, содержащие производные второго и более высоких порядков:

$$Y(x_i + h) = Y(x_i) + Y'(x_i) \cdot h + O(h^2)$$

Полагаем: $Y'(x_i) = f(x_i, Y(x_i)) = f(x_i, y_i) = \frac{y_{i+1} - y_i}{h}$

Введем последовательность равноотстоящих точек $x_0, x_1, \dots, x_n$ (узлов), выбрав малый шаг $h = x_{i+1} - x_i = const$. Тогда получаем **формулу Эйлера**:

$$\boldsymbol{y_{i+1} = y_i + hf(x_i, y_i)} \qquad (7)$$

При i=0 находим значение сеточной функции y_1 при $x = x_1$:

$$y_1 = y_0 + hf(x_0, y_0)$$

Значение y_0 задано начальным условием:

$$\boldsymbol{y_0 = Y_0} \qquad (8)$$

Аналогично могут быть найдены значения сеточной функции в других узлах:

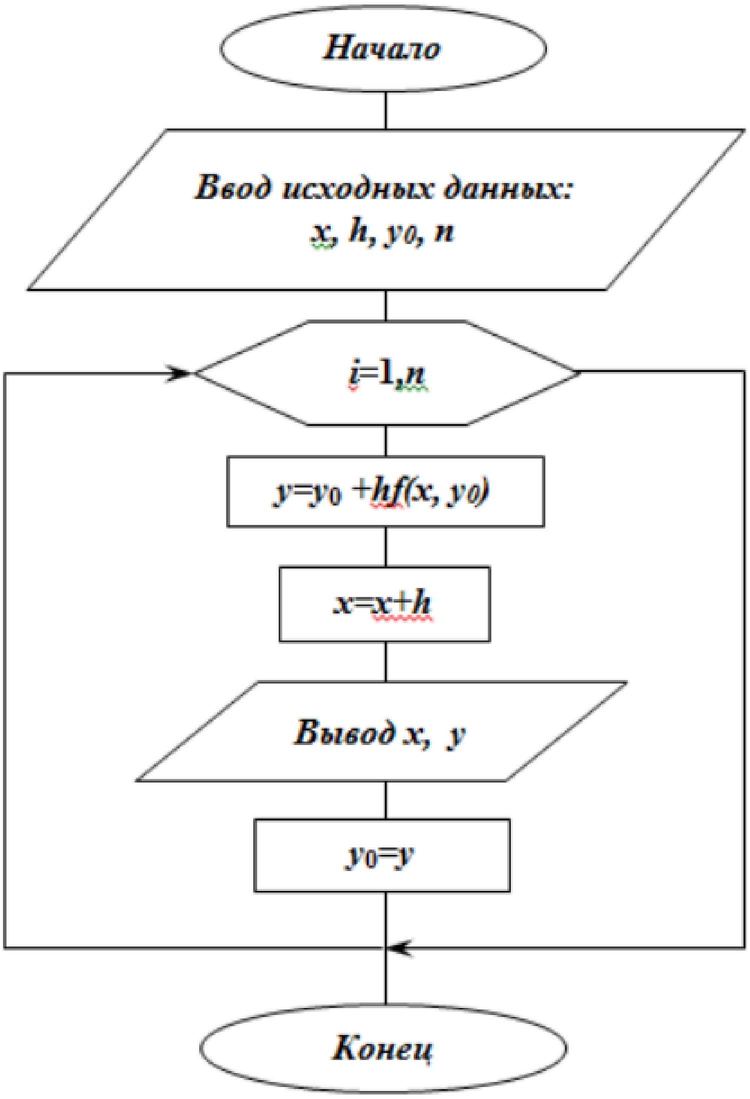
$$y_2 = y_1 + hf(x_1, y_1)$$

.....

$$y_n = y_{n-1} + hf(x_{n-1}, y_{n-1})$$

Построенный алгоритм называется методом Эйлера. Разностная схема этого метода представлена соотношениями (7), (8). Они имеют вид рекуррентных формул, с помощью которых значение сеточной функции y_{i+1} в любом узле x_{i+1} вычисляется по ее значению y_i в предыдущем узле x_i. Поэтому метод Эйлера относится к ***одношаговым*** методам.

Блок-схема метода Эйлера



Метод Рунге-Кутта 4-го порядка

$$y_{i+1} = y_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4),$$

$$k_1 = h \cdot f(x_i, y_i)$$

$$k_2 = h \cdot f(x_i + \frac{h}{2}, y_i + \frac{k_1}{2})$$

$$k_3 = h \cdot f(x_i + \frac{h}{2}, y_i + \frac{k_2}{2})$$

$$k_4 = h \cdot f(x_i + h, y_i + k_3)$$

Метод милна

Вычислительные формулы:

а) этап прогноза:

$$y_i^{\text{прогн}} = y_{i-4} + \frac{4h}{3}(2f_{i-3} - f_{i-2} + 2f_{i-1})$$

б) этап коррекции:

$$y_i^{\text{корр}} = y_{i-2} + \frac{h}{3}(f_{i-2} + 4f_{i-1} + f_i^{\text{прогн}})$$

$$f_i^{\text{прогн}} = f(x_i, y_i^{\text{прогн}})$$

Для начала счета требуется задать решения в трех первых точках, которые можно получить одношаговыми методами (например, методом Рунге-Кутта).

Листинг программы

Одноступенчатые методы:

```go
const MAX_ITERS = 200

func rungeRule(x, y, x2, y2 []float64, p int, eps float64) bool {
   for i := range x {
      errorValue := math.Abs(y[i]-y2[2*i]) / (math.Pow(2, float64(p)) - 1)
      if errorValue > eps {
         return false
      }
   }
   return true
}

func OneStepMethodSolve(method func(func(float64, float64) float64, float64, float64,
float64, int) ([]float64, []float64),
   f func(float64, float64) float64, x0, y0, hInitial float64, n int, eps float64, p int)
([]float64, []float64) {

   iter := 0
   for {
      if iter >= MAX_ITERS {
         x, y := method(f, x0, y0, hInitial, n)
         fmt.Println("Warning! Превышен лимит итераций:", MAX_ITERS)
         return x, y
      }

      x, y := method(f, x0, y0, hInitial, n)
      h2 := hInitial / 2
      n2 := n * 2
      x2, y2 := method(f, x0, y0, h2, n2)

      if iter <= 1 {
         fmt.Println("y[n]:", y[len(y)-1])
      }

      if rungeRule(x, y, x2, y2, p, eps) {
         return x, y
      }

      hInitial = h2
      n = n2
      iter++
   }
}
```

Метод Эйлера (обычный):

```go
func EulerMethod(f func(float64, float64) float64, x0, y0, h float64, n int) ([]float64,
[]float64) {
   x := make([]float64, n+1)
   y := make([]float64, n+1)
   x[0], y[0] = utils.Rounding(x0), utils.Rounding(y0)

   for i := 1; i <= n; i++ {
      y[i] = utils.Rounding(y[i-1] + h*f(x[i-1], y[i-1]))
      x[i] = utils.Rounding(x[i-1] + h)
   }
   return x, y
}
```

Метод Рунге-Кутта:

```
func RungeKutta(f func(float64, float64) float64, x0, y0, h float64, n int) ([]float64,
[]float64) {
    x := make([]float64, n+1)
    y := make([]float64, n+1)
    x[0], y[0] = utils.Rounding(x0), utils.Rounding(y0)

    for i := 1; i <= n; i++ {
        k1 := h * f(x[i-1], y[i-1])
        k2 := h * f(x[i-1]+h/2, y[i-1]+k1/2)
        k3 := h * f(x[i-1]+h/2, y[i-1]+k2/2)
        k4 := h * f(x[i-1]+h, y[i-1]+k3)
        y[i] = utils.Rounding(y[i-1] + (k1+2*k2+2*k3+k4)/6)
        x[i] = utils.Rounding(x[i-1] + h)
    }
    return x, y
}
```

Многоступенчатый метод Милна:

```
func MilneMethod(f func(float64, float64) float64, x0, y0, h float64, n int, eps float64,
yPrecise func(float64, float64) float64) ([]float64, []float64) {
    x, y := one_step_methods.RungeKutta(f, x0, y0, h, n)

    for i := range x {
        fmt.Println(x[i], y[i])
    }
    fmt.Println()
    fmt.Println()

    x = x[:4]
    y = y[:4]

    h2 := h / 2
    n2 := 2 * n
    for i := 3; i <= n; i++ {
        yPredict := y[i-3] + 4*h/3*(2*f(x[i-2], y[i-2])-f(x[i-1], y[i-1])+2*f(x[i], y[i]))
        xNext := x[i] + h
        y = append(y, utils.Rounding(y[i-1]+h/3*(f(x[i-1], y[i-1])+4*f(x[i], y[i])+f(xNext,
yPredict))))
        x = append(x, utils.Rounding(xNext))
    }
    epsActual := math.Abs(yPrecise(x[n], y[n]) - y[n])
    if epsActual > eps {
        x1, y1 := MilneMethod(f, x0, y0, h2, n2, eps, yPrecise)
        return x1, y1
    }
    return x, y
}
```

Примеры и результаты работы программы

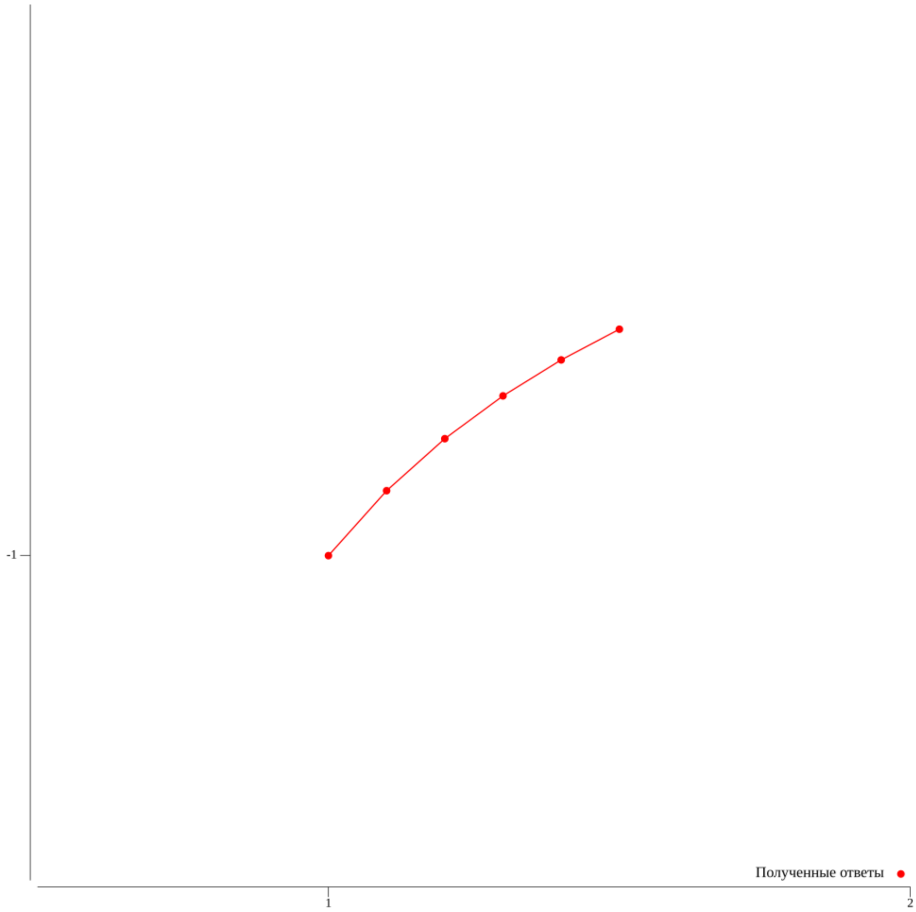
Метод Эйлера:

```
Выберите метод:
1. Метод Эйлера
2. Метод Рунге-Кутта 4- го порядка
3. Метод Милна
Выберите тип ввода (1-3):
1
Выбран метод - Метод Эйлера


Выберите функцию:
1. y' = y + (1 + x) * y^2
2. y' = x * y^2
3. y' = y - sinx
Выберите тип ввода (1-3):
1
Выбрана функция: y' = y + (1 + x) * y^2
Введите x0 и y0 разделенные пробелом: 1 -1
Введите n и h разделенные пробелом: 5 0.1
введите e: 0.01
x0 = -1
y0 = 1
n = 5
h = 0.1
e = 0.01
y(x0) = 1
y[n]: -0.65136
y[n]: -0.65937

Таблица решения задачи Коши обычным методом Эйлера:
+---+-----+----------+--------------+---------+----------+--------------+----------------+
| # |   X |        Y | ПОГРЕШНОСТЬ  | X AT H/2 | Y AT H/2 | ПОГРЕШНОСТЬ  | ТОЧНОЕ РЕШЕНИЕ |
+---+-----+----------+--------------+---------+----------+--------------+----------------+
| 0 |   1 |       -1 |            0 |       1 |       -1 |            0 |             -1 |
| 1 | 1.1 |     -0.9 |      0.00909 |     1.1 | -0.90499 |       0.0041 |       -0.90909 |
| 2 | 1.2 |  -0.8199 |      0.01343 |     1.2 | -0.82716 |      0.00617 |       -0.83333 |
| 3 | 1.3 |   -0.754 |      0.01523 |     1.3 | -0.76213 |       0.0071 |       -0.76923 |
| 4 | 1.4 | -0.69864 |      0.01565 |     1.4 |  -0.7069 |      0.00739 |       -0.71429 |
| 5 | 1.5 | -0.65136 |      0.01531 |     1.5 | -0.65937 |       0.0073 |       -0.66667 |
+---+-----+----------+--------------+---------+----------+--------------+----------------+

График построен: out/simple_euler.png
```



Метод Рунге-Кутта:

Выберите метод:
1. Метод Эйлера
2. Метод Рунге-Кутта 4- го порядка
3. Метод Милна
Выберите тип ввода (1-3):
2
Выбран метод - Метод Рунге-Кутта 4- го порядка

Выберите функцию:
1. y' = y + (1 + x) * y^2
2. y' = x * y^2
3. y' = y - sinx

Выберите тип ввода (1-3):
1

Выбрана функция: y' = y + (1 + x) * y^2

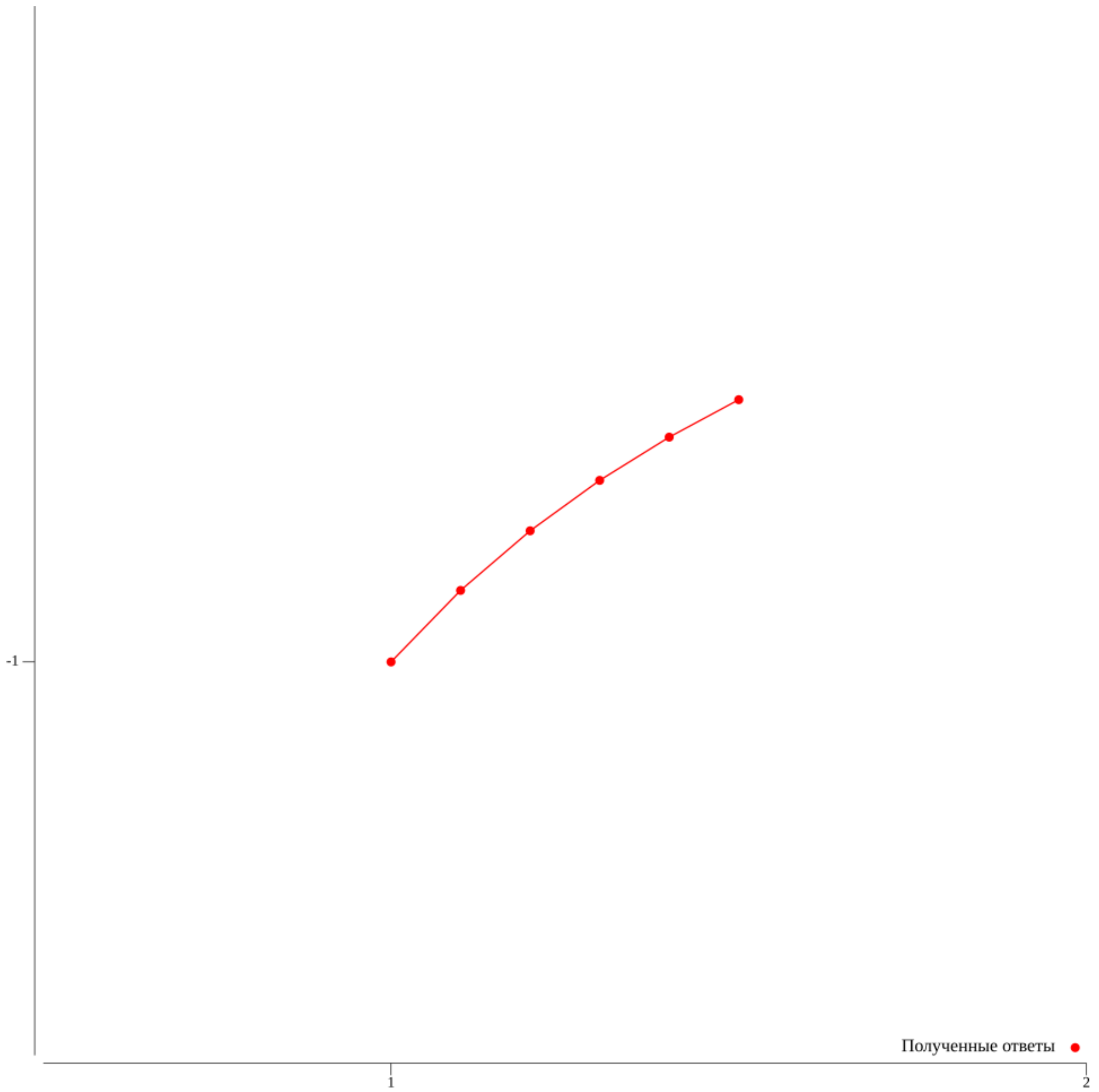
Введите x0 и y0 разделенные пробелом: 1 -1
Введите n и h разделенные пробелом: 5 0.1
введите e: 0.01
x0 = -1
y0 = 1
n = 5
h = 0.1
e = 0.01

y(x0) = 1
y[n]: -0.66667
y[n]: -0.66667

Таблица решения задачи Коши методом Рунге-Кутта:

#	X	Y	ПОГРЕШНОСТЬ	X AT H/2	Y AT H/2	ПОГРЕШНОСТЬ	ТОЧНОЕ РЕШЕНИЕ
0	1	-1	0	1	-1	0	-1
1	1.1	-0.90909	0	1.1	-0.90909	0	-0.90909
2	1.2	-0.83333	0	1.2	-0.83333	0	-0.83333
3	1.3	-0.76923	0	1.3	-0.76923	0	-0.76923
4	1.4	-0.71429	0	1.4	-0.71429	0	-0.71429
5	1.5	-0.66667	0	1.5	-0.66667	0	-0.66667

График построен: out/runge_kutta.png



Метод Милна:

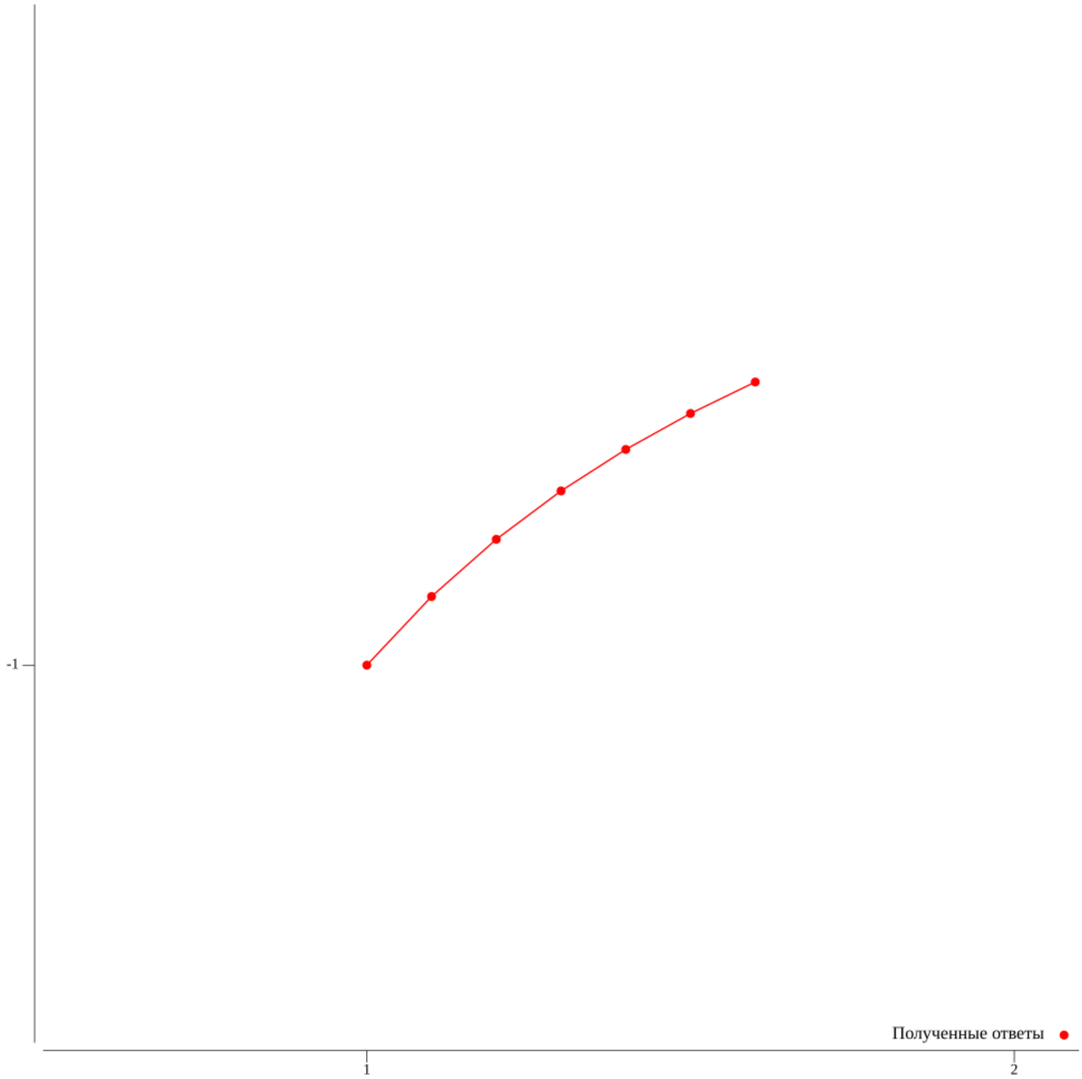
```
Выберите метод:
1. Метод Эйлера
2. Метод Рунге-Кутта 4- го порядка
3. Метод Милна
Выберите тип ввода (1-3):
3
Выбран метод - Метод Милна


Выберите функцию:
1. y' = y + (1 + x) * y^2
2. y' = x * y^2
3. y' = y - sinx
Выберите тип ввода (1-3):
1
Выбрана функция: y' = y + (1 + x) * y^2
Введите x0 и y0 разделенные пробелом: 1 -1
Введите n и h разделенные пробелом: 5 0.1
введите e: 0.01
x0 = -1
y0 = 1
n = 5
h = 0.1
e = 0.01
y(x0) = 1
1 -1
1.1 -0.90909
1.2 -0.83333
1.3 -0.76923
1.4 -0.71429
1.5 -0.66667


1.1 1.1
1.2000000000000002 1.2
1.3000000000000003 1.3
1.4000000000000004 1.4
1.5000000000000004 1.5
1.6000000000000005 1.6

Таблица решения задачи Коши методом Милна:
+---+-----+----------+------------------+----------------+
| # |   X |        Y | ТОЧНОЕ РЕШЕНИЕ | ПОГРЕШНОСТЬ |
+---+-----+----------+------------------+----------------+
| 0 |   1 |       -1 |               -1 |              0 |
| 1 | 1.1 | -0.90909 |         -0.90909 |              0 |
| 2 | 1.2 | -0.83333 |         -0.83333 |              0 |
| 3 | 1.3 | -0.76923 |         -0.76923 |              0 |
| 4 | 1.4 | -0.71427 |         -0.71429 |          2e-05 |
| 5 | 1.5 | -0.66666 |         -0.66667 |          1e-05 |
| 6 | 1.6 | -0.62498 |           -0.625 |          2e-05 |
+---+-----+----------+------------------+----------------+

График построен: out/milna.png
```



Выводы

В ходе выполнения данной лабораторной работы было определено, что представляет собой задача Коши. Были изучены численные методы решения ОДУ, включая одношаговые и многошаговые подходы. Эти методы были реализованы в программе, которая решает задачу Коши для различных дифференциальных уравнений.