Университет ИТМО

Факультет программной инженерии и компьютерной техники

Лабораторная работа №6 по "Вычислительной математике"

Численное дифференцирование

Вариант 4

Выполнил: Дьяконов Михаил Павлович
Группа: P3211
Преподаватель: Малышева Татьяна Алексеевна

Университет ИТМО, 2021

Цель работы

Решить задачу Коши численными методами. Для исследования использовать одношаговые и многошаговые методы.

Рабочие формулы

Усовершенствованный метод Эйлера:

$$y_{i+1} = y_i + \frac{h}{2}\left[f(x_i, y_i) + f\left(x_{i+1}, y_i + hf(x_i, y_i)\right)\right],\ i = 0, 1 \ldots$$

Метод Адамса:

$$\Delta f_i = f_i - f_{i-1}$$
$$\Delta^2 f_i = f_i - 2f_{i-1} + f_{i-2}$$
$$\Delta^3 f_i = f_i - 3f_{i-1} + 3f_{i-2} - f_{i-3}$$

$$y_{i+1} = y_i + hf_i + \frac{h^2}{2}\Delta f_i + \frac{5h^3}{12}\Delta^2 f_i + \frac{3h^4}{8}\Delta^3 f_i$$

Правило Рунге:

$$R = \frac{y^h - y^{2h}}{2^p - 1},$$

Листинг программы

AdamsMethod.kt:

```
class AdamsMethod : DiffMethod {

    override fun calculate(input: Input): MethodResult {
        val xValues = DoubleArray(input.n) {input.x0 + it*input.h}
        val exactYValues: DoubleArray = calculateExactValues(input.func, xValues,
            input.func.const(input.x0, input.y0))
        val firstYValues = getFirstYValues(input)
        val yValues = DoubleArray(input.n) {if(it < FIRST_ELEMENTS_COUNT) firstYValues[it] else 0.0}
        val derivativeValues = DoubleArray(input.n) {if(it < FIRST_ELEMENTS_COUNT)
            input.func.derivative(xValues[it], yValues[it]) else 0.0}
        val indexes = IntArray(input.n) {it}

        for(i in FIRST_ELEMENTS_COUNT until input.n) {
            derivativeValues[i-1] = input.func.derivative(xValues[i-1], yValues[i-1])
            val delta1F: Double = derivativeValues[i-1] - derivativeValues[i-2]
            val delta2F: Double = derivativeValues[i-1] - 2*derivativeValues[i-2] +
                derivativeValues[i-3]
            val delta3F: Double = derivativeValues[i-1] - 3*derivativeValues[i-2] +
                3*derivativeValues[i-3] - derivativeValues[i-4]
            yValues[i] = yValues[i-1] + input.h*derivativeValues[i-1] + input.h.pow(2)/2*delta1F +
                    5/12 * input.h.pow(3)*delta2F + 3/8 * input.h.pow(4)*delta3F
        }
        derivativeValues[input.n-1] = input.func.derivative(xValues[input.n-1], yValues[input.n-1])

        return MethodResult(indexes, xValues, yValues, exactYValues, derivativeValues)
    }

    private fun getFirstYValues(input: Input): DoubleArray {
        return RungeKuttaMethod().calculate(Input(input.func, input.method, input.x0, input.y0,
            input.x0+input.h*3, input.h, FIRST_ELEMENTS_COUNT, input.accuracy)).yValues
    }

    override fun getAccuracyOrder(): Int {
        return 4
```

```
    }
}
```

AdvancedEulerMethod.kt:

```
class AdvancedEulerMethod : DiffMethod {

    override fun calculate(input: Input): MethodResult {
        val xValues = DoubleArray(input.n) {input.x0 + it*input.h}
        val exactYValues: DoubleArray = calculateExactValues(input.func, xValues,
            input.func.const(input.x0, input.y0))
        val yValues = DoubleArray(input.n) {input.y0}
        val derivativeValues = DoubleArray(input.n)
        val indexes = IntArray(input.n) {it}

        for(i in 1 until input.n) {
            derivativeValues[i-1] = input.func.derivative(xValues[i-1], yValues[i-1])
            yValues[i] = yValues[i-1] + input.h/2 * (derivativeValues[i-1]
                    + input.func.derivative(xValues[i], yValues[i-1]+input.h*derivativeValues[i-1]))
        }
        derivativeValues[input.n-1] = input.func.derivative(xValues[input.n-1], yValues[input.n-1])

        return MethodResult(indexes, xValues, yValues, exactYValues, derivativeValues)
    }

    override fun getAccuracyOrder(): Int {
        return 2
    }
}
```

RungeKuttaMethod.kt:

```
class RungeKuttaMethod : DiffMethod {

    override fun calculate(input: Input): MethodResult {
        val xValues = DoubleArray(input.n) {input.x0 + it*input.h}
        val exactYValues: DoubleArray = calculateExactValues(input.func, xValues,
            input.func.const(input.x0, input.y0))
        val yValues = DoubleArray(input.n) {input.y0}
        val indexes = IntArray(input.n) {it}

        for(i in 1 until input.n) {
            val k1 = input.h * input.func.derivative(xValues[i-1], yValues[i-1])
            val k2 = input.h * input.func.derivative(xValues[i-1]+input.h/2, yValues[i-1]+k1/2)
            val k3 = input.h * input.func.derivative(xValues[i-1]+input.h/2, yValues[i-1]+k2/2)
            val k4 = input.h * input.func.derivative(xValues[i-1]+input.h, yValues[i-1]+k3)
            yValues[i] = yValues[i-1] + (k1 + 2*k2 + 2*k3 + k4)/6
        }

        return MethodResult(indexes, yValues, xValues, exactYValues)
    }

    override fun getAccuracyOrder(): Int {
        return 4
    }
}
```

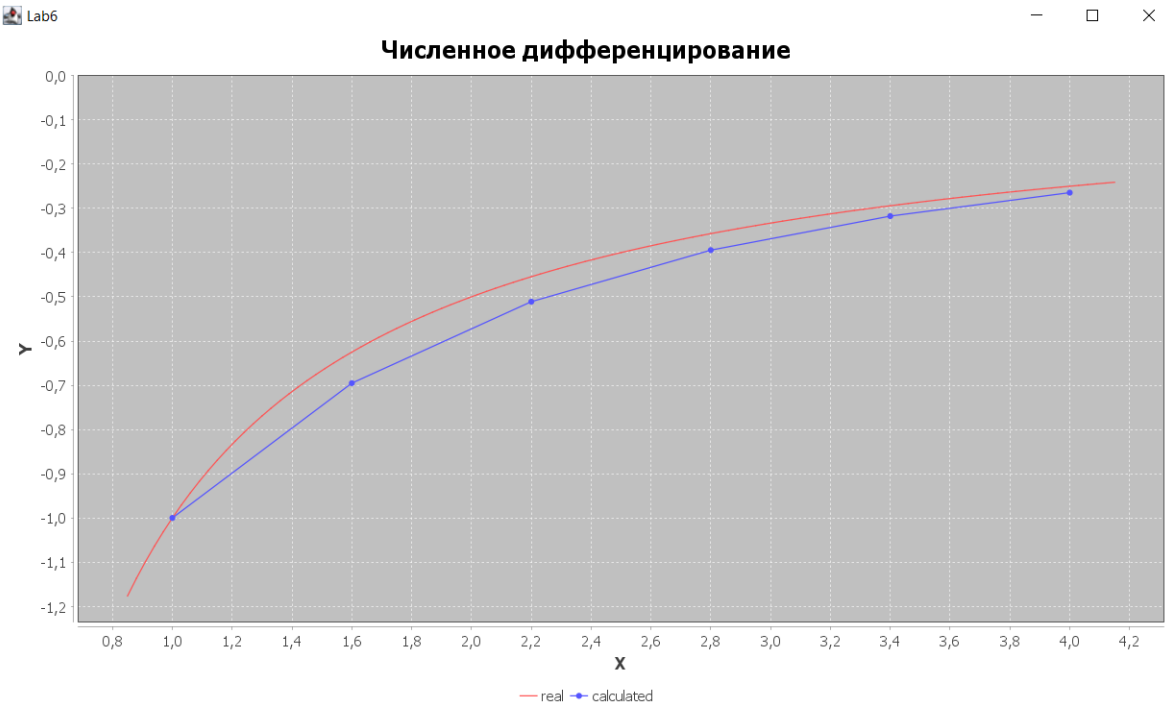
RungeCalculator.kt:

```
class RungeCalculator {

    fun calculateR(input: Input): Double {
```

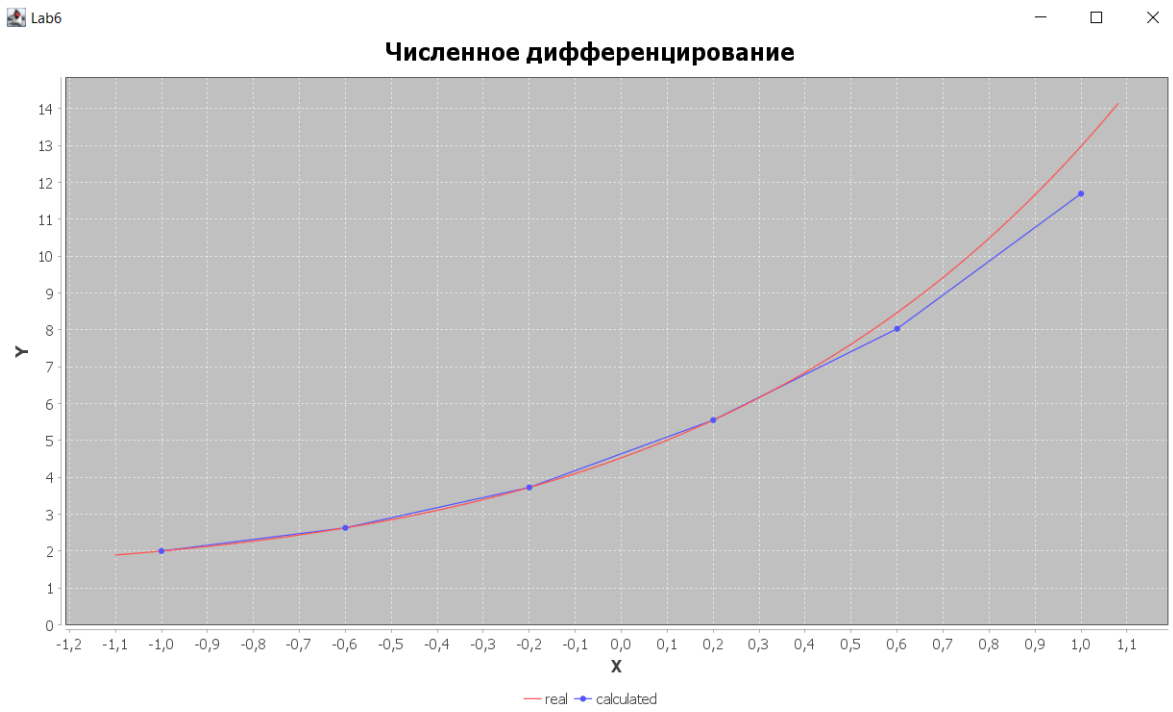
```
        val method: DiffMethod = if(input.method == 1) AdvancedEulerMethod() else AdamsMethod()
        val yh = method.calculate(input).yValues[2]
        val y2h = method.calculate(Input(input.func, input.method, input.x0, input.y0,
            input.xn, input.h * 2, input.n, input.accuracy)).yValues[1]

        return (yh - y2h) / (2.0.pow(method.getAccuracyOrder()) - 1)
    }
}
```

Примеры

```
Выберите ОДУ:
1 - y' = y + (x + 1)*y^2
2 - y' = e^(2x) + y
3 - y' = sin(x) + y
1
Выберите метод:
1 - Усовершенствованный метод Эйлера
2 - Метод Адамса
1
Введите x0 и y0:
1 -1
Введите конец интервала(x_n):
4
Введите интервал(h):
0.6
Введите точность([1.0E-6 - 0.1]):
0.001
========================
i               x_i             y_i             f(x_i, y_i)     Точное
0               1.0             -1.0            1.0             -1.0
1               1.6             -0.695          0.561           -0.625
2               2.2             -0.511          0.325           -0.455
3               2.8             -0.394          0.197           -0.357
4               3.4             -0.318          0.126           -0.294
5               4.0             -0.265          0.085           -0.25
Погрешность по правилу Рунге: -0.1026012661
```



```
Выберите ОДУ:
1 - y' = y + (x + 1)*y^2
2 - y' = e^(2x) + y
3 - y' = sin(x) + y
3
Выберите метод:
1 - Усовершенствованный метод Эйлера
2 - Метод Адамса
2
Введите x0 и y0:
-1 2
Введите конец интервала(x_n):
1
Введите интервал(h):
0.4
Введите точность([1.0E-6 - 0.1]):
0.001
=========================
i               x_i             y_i             f(x_i, y_i)     Точное
0               -1.0            2.0             1.159           2.0
1               -0.6            2.629           2.064           2.629
2               -0.2            3.725           3.526           3.725
3               0.2             5.55            5.749           5.551
4               0.6             8.027           8.592           8.465
5               1.0             11.691          12.533          12.975
Погрешность по правилу Рунге: 2.934013E-4
```



Вывод

В результате выполнения данной лабораторной работы я узнал, какие существуют методы численного дифференцирования, реализовал одношаговый и многошаговый методы для дифференцирования ОДУ первого порядка.