Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»
КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Лабораторная работа № 1

Тема Программная реализация приближенного аналитического метода и численных алгоритмов первого и второго порядков точности при решении задачи Коши для ОДУ

Студент Якуба Д. В.

Группа ИУ7-63Б

Оценка (баллы) _____

Преподаватель Градов В. М.

Москва.
2021 г.

Лабораторная работа по теме «Программная реализация приближенного аналитического метода и численных алгоритмов первого и второго порядков точности при решении задачи Коши для ОДУ»

Тема:
Программная реализация приближенного аналитического метода и численных алгоритмов первого и второго порядков точности при решении задачи Коши для ОДУ.

Цель работы:
Получение навыков решения задачи Коши для ОДУ методами Пикара и явными методами первого порядка точности (Эйлера) второго порядка точности (Рунге-Кутта).

Задание:
Задано ОДУ, не имеющее аналитического решения:

$$\begin{cases} u(x) = x^2 + u^2 \\ u(0) = 0 \end{cases}$$

Требуется построить таблицу, содержащую значения аргумента с заданным шагом в интервале [0, x_{max}] и результаты расчета функции $u(x)$ в приближениях Пикара (от 1-го до 4-го), а также численными методами. Границу интервала x_{max} выбирать максимально возможной из условия, чтобы численные методы обеспечивали точность вычисления решения уравнения $u(x)$ до второго знака после запятой.

Входные данные:
Конечное значение x_{max}, шаг.

Выходные данные:
Таблица, содержащая значения аргумента с заданным шагом в интервале [0, x_{max}] и результаты расчета функции $u(x)$ в приближениях Пикара (от 1-го до 4-го), а также численными методами.

Описание

Обыкновенными дифференциальными уравнениями (ОДУ) называются уравнения с одной независимой переменной. Если независимых переменных больше, чем одна, то уравнение называется дифференциальным уравнением с частными производными.

Задача Коши

Общее решения ДУ n-го порядка зависит от констант общего решения. Для выделения частного решения требуется задать n условий.

В задаче Коши все дополнительные условия задаются в одной точке:

$$u(x) = \varphi(x, C_1, C_2, \dots, C_n)$$

$$u_k(\xi) = \eta_k, k = 1, \dots, n$$

Можно выделить три метода решения обыкновенных дифференциальных уравнений в задаче Коши: аналитические, аналитические приближенные и численные.

Метод Пикара

Данный метод является представителем приближенных методов решения рассматриваемого класса задач. Идея метода чрезвычайно проста и сводится к процедуре последовательных приближений для решения интегрального уравнения, к которому приводится исходное дифференциальное уравнение. Пусть поставлена задача Коши:

$$\nu'(x) = \varphi\big(x, \nu(x)\big), x_0 \le x \le x_1, \nu(x_0) = \nu_0$$

Проинтегрируем выписанное уравнение:

$$\nu(x) = \nu + \int_{x_0}^{x} \varphi\big(t, \nu(t)\big)dt$$

Процедура последовательных приближений метода Пикара реализуется согласно следующей схеме:

$$y_n(x) = \nu_0 + \int_{x_0}^{x} \varphi\big(t, y_{n-1}(t)\big)dt,$$

причём $y_0(t) = \nu_0, (i - \text{номер итерации})$.

Метод Эйлера

Метод Эйлера — простейший численный метод решения систем обыкновенных дифференциальных уравнений.

Метод Эйлера в явном виде представлен формулой:

$$y_{n+1} = y_n + h\varphi(x_n, y_n)$$

где

$$\varphi(x, y) = x^2 + y^2$$

Метод Рунге-Кутта 2-го порядка

Методы Рунге-Кутта являются численными. На практике применяются методы Рунге-Кутта, обеспечивающие построение разностных схем

различного порядка точности. Наиболее употребительны схемы второго и четвертого порядков.

Используя формулу Тейлора, решение дифференциального уравнения можно представить в виде:

$$\nu_{n+1} = \nu_n + h_n \nu'_n + \frac{1}{2} h_n^2 \nu''_n + \cdots (1),$$

где обозначено $\nu_n = \nu(x_n), \nu'_n = \nu'(x_n), h_n = x_{n+1} - x_n$.

Согласно $\nu'_n = \varphi(x_n, \nu_n), \nu''_n = \varphi'_x(x_n, \nu_n) + \varphi'_\nu(x_n, \nu_n)\varphi(x_n, \nu_n)$. Далее удерживаем только выписанные члены ряда. Представим вторую производную следующим образом:

$$\nu''_n = (\nu'_n)' = \frac{\varphi(a, b) - \varphi(x_n, y_n)}{\Delta x}$$

Пусть $a = x_n + \gamma h, b = \nu_n + \delta h$.

Обозначим приближенное значение решения в узле с номером n через y_n.

Имеем:

$$y_{n+1} = y_n + h_n \varphi(x_n, y_n) + \frac{1}{2} h_n^2 \left[\frac{\varphi(x_n + \gamma h_n, y_n + \delta h_n) - \varphi(x_n, y_n)}{\Delta x} \right]$$
$$= y_n + h_n [\beta \varphi(x_n, y_n) + \alpha \varphi(x_n + \gamma h_n, y_n + \delta h_n)]$$

Введенные здесь параметры $\alpha, \beta, \gamma, \delta$ подлежат определению. Разлагая правую часть в ряд Тейлора до линейных членом и приводя подобные члены, получим последовательно:

$$y_{n+1} = y_n + h_n \{\beta \varphi(x_n, y_n \alpha [\varphi(x_n, y_n) + \varphi'_x(x_n, y_n)\gamma h_n + \varphi'_y(x_n, y_n)\delta h_n]\}$$
$$= y_n + (\alpha + \beta) h_n \varphi(x_n, y_n) + \alpha h_n^2 [\gamma \varphi'_x(x_n, y_n) + \delta \varphi'_y(x_n, y_n)] \quad (2)$$

Условием выбора параметров $\alpha, \beta, \gamma, \delta$ поставим близость выражения (2) ряду (1), тогда:

$$\alpha + \beta = 1, \alpha\gamma = \frac{1}{2}, \alpha\delta = \frac{1}{2}\varphi(x_n, y_n)$$

Один параметр остается свободным. Пусть это будет α, тогда:

$$\beta = 1 - \alpha, \gamma = \frac{1}{2\alpha}, \delta = \frac{1}{2\alpha}\varphi(x_n, y_n)$$

Таким образом получим:

$$y_{n+1} = y_n + h_n \left\{ (1 - \alpha)\varphi(x_n, y_n) + \alpha \varphi \left(x_n + \frac{1}{2\alpha} h_n, y_n + \frac{h_n}{2\alpha} \varphi(x_n, y_n) \right) \right\}$$

Результат

Вычисления производились с шагом $h = 10^{-5}$, для демонстрации выводится каждый 1000-ый результат.

```
"C:\Program Files\Java\jdk-15.0.1\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2020.1.4\lib\idea_rt.jar=50554:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2020.
```

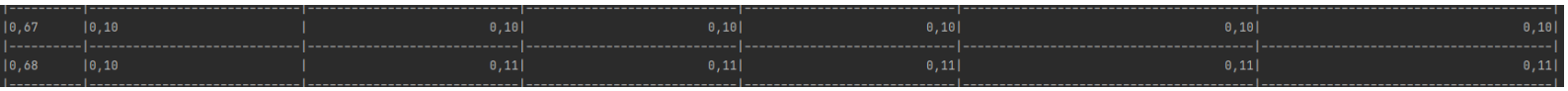
x	Метод Пикара				Метод Эйлера	Метод Рунге-Кутта 2-го порядка точности
	Приближение 1	Приближение 2	Приближение 3	Приближение 4		
0,000000	0,000000	0,000000	0,000000	0,000000	0,000000	0,000000
0,100000	0,000333	0,000333	0,000333	0,000333	0,000333	0,000333
0,200000	0,002667	0,002667	0,002667	0,002667	0,002665	0,002667
0,300000	0,009000	0,009003	0,009003	0,009003	0,008999	0,009003
0,400000	0,021333	0,021359	0,021359	0,021359	0,021351	0,021359
0,500000	0,041667	0,041791	0,041791	0,041791	0,041778	0,041791
0,600000	0,072000	0,072444	0,072448	0,072448	0,072429	0,072448
0,700000	0,114333	0,115641	0,115660	0,115660	0,115634	0,115660
0,800000	0,170667	0,173995	0,174079	0,174080	0,174045	0,174080
0,900000	0,243000	0,250592	0,250897	0,250906	0,250860	0,250907
1,000000	0,333333	0,349206	0,350185	0,350230	0,350169	0,350232
1,100000	0,443667	0,474599	0,477414	0,477606	0,477533	0,477617
1,200000	0,576000	0,632876	0,640282	0,641016	0,640960	0,641077
1,300000	0,732333	0,831934	0,850035	0,852572	0,852715	0,852880
1,400000	0,914667	1,081990	1,123560	1,131680	1,132866	1,133113
1,500000	1,125000	1,396205	1,486771	1,511146	1,517052	1,517448
1,600000	1,365333	1,791421	1,980024	2,049464	2,075721	2,076423
1,700000	1,637667	2,288998	2,666774	2,856462	2,971335	2,972797
1,800000	1,944000	2,915778	3,647363	4,148638	4,684098	4,688130
1,900000	2,286333	3,705177	5,080810	6,372211	9,545668	9,566992
2,000000	2,666667	4,698413	7,218990	10,483923	270,068406	317,566479
2,100000	3,087000	5,945871	10,459783	18,605273	Infinity	Infinity
2,200000	3,549333	7,508632	15,428932	35,570143	Infinity	Infinity
2,300000	4,055667	9,460152	23,104821	72,620544	Infinity	Infinity

Контрольные вопросы

1. Укажите интервалы значений аргумента, в которых можно считать решением заданного уравнения каждое из первых 4-х приближений Пикара. Точность результата оценивать до второй цифры после запятой. Объяснить свой ответ.

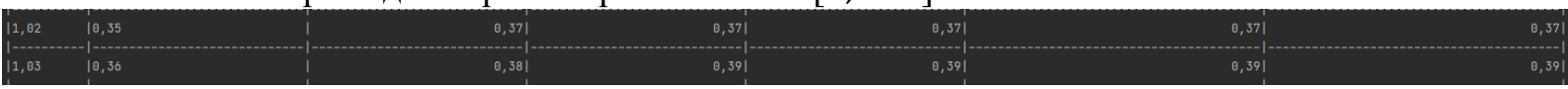
Ответ: для нахождения интервалов значений аргумента, в которых можно считать решение заданного уравнения каждое из первых 4-х приближений Пикара потребуется уменьшать шаг и проводить сравнительный анализ для разных приближений. Интервал для некоторого приближения будет включать в себя все значения аргумента, при которых полученные решения методом Пикара будут совпадать со значениями (до второй цифры, как это указано в вопросе) более высоких порядков приближения и рассмотренных численных методов при шаге меньшим или равным 10^{-5}.

Имеем:



0,67	0,10	0,10	0,10	0,10	0,10	0,10
0,68	0,10	0,11	0,11	0,11	0,11	0,11

Интервал для первого приближения: [0,0.67]

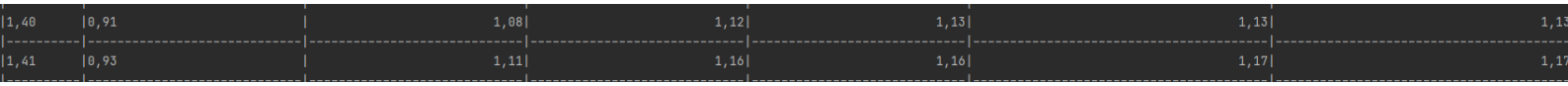


1,02	0,35	0,37	0,37	0,37	0,37	0,37
1,03	0,36	0,38	0,39	0,39	0,39	0,39

Интервал для второго приближения: [0,1.02]

1,27	0,68	0,77	0,78	0,78	0,78	0,78
1,28	0,70	0,79	0,80	0,81	0,81	0,81

Интервал для третьего приближения: [0,1.27]



1,40	0,91	1,08	1,12	1,13	1,13	1,13
1,41	0,93	1,11	1,16	1,16	1,17	1,17

Интервал для четвёртого приближения: [0,1.40]

2. Пояснить, каким образом можно доказать правильность полученного результата при фиксированном значении аргумента в численных методах.

Ответ: доказать правильность полученного результата при фиксированном значении аргумента в численных методах можно посредством постепенного уменьшения шага. Если при уменьшении шага полученный результат изменится незначительно (относительно предыдущих изменений), полученный результат можно считать правильным.

3. Каково значение функции при x=2, то есть привести значение u(2).

Ответ: опираясь на вопрос №2, привести значение $u(2)$ можно посредством уменьшения шага и анализом изменения полученного численным методом значения.

2,000000	2,666667	4,698413	7,218990	10,483923	28,392534	143,913416

Для $h = 10^{-3}$ $u(2) = 143{,}913416$;

2,000000	2,666667	4,698413	7,218990	10,483923	126,596561	305,208027

Для $h = 10^{-4}$ $u(2) = 305{,}208027$;

Разница с предыдущим значением: 161,294611.

2,000000	2,666667	4,698413	7,218990	10,483923	270,068406	317,566479

Для $h = 10^{-5}$ $u(2) = 317{,}566479$;

Разница с предыдущим значением: 12,358452.

2,000000	2,666667	4,698413	7,218990	10,483923	312,061276	317,720876

Для $h = 10^{-6}$ $u(2) = 317{,}720876$;

Разница с предыдущим значением: 0,154397.

2,000000	2,666667	4,698413	7,218990	10,483923	317,145350	317,722444

Для $h = 10^{-7}$ $u(2) = 317{,}722444$;

Разница с предыдущим значением: 0,001568.

Таким образом: $u(2) \approx 317.72$

Код программы

```
import kotlin.math.abs
import kotlin.math.ceil
import kotlin.math.pow

const val xF = 10
const val picF = 120
const val picFfo = picF / 4 - 1
const val eulF = 40
const val rkF = 40

fun firstApprox(x: Double) : Double
{
    return x * x * x / 3
}

fun secondApprox(x: Double) : Double
{
    return firstApprox(x) + x.pow(7) / 63
}

fun thirdApprox(x: Double) : Double
{
    return secondApprox(x) + x.pow(11) * 2 / 2079 + x.pow(15) / 59535
}

fun fourthApprox(x: Double) : Double
{
    return secondApprox(x) + 2 * x.pow(11) / 2079 + 13 * x.pow(15) / 218295 + 82 *
x.pow(19) / 37328445 +
            662 * x.pow(23) / 10438212015 + 4 * x.pow(27) / 3341878155 +
            x.pow(31) / 109876902975
}

fun picardMet(xStart: Double, step: Double, numOfIters: Int) :
MutableList<MutableList<Double>>
{
    val outList: MutableList<MutableList<Double>> = mutableListOf()

    var curX = xStart

    outList.add(mutableListOf(curX, firstApprox(curX), secondApprox(curX),
thirdApprox(curX), fourthApprox(curX)))

    for (i in 1..numOfIters)
```

```
    {
        curX += step
        outList.add(mutableListOf(curX, firstApprox(curX), secondApprox(curX),
thirdApprox(curX), fourthApprox(curX)))
    }

    return outList
}

fun curMathFun(x: Double, y: Double) : Double { return x * x + y * y}

fun eulerMet(xStart: Double, yStart: Double, step: Double, numOfIters: Int) :
MutableList<Double>
{
    val outList: MutableList<Double> = mutableListOf()

    var curX = xStart
    var curY = yStart

    outList.add(curY)
    for (i in 1..numOfIters)
    {
        curY += step * curMathFun(curX, curY)

        outList.add(curY)

        curX += step
    }

    return outList
}

fun rungeKutMet(xStart: Double, yStart: Double, alpha: Double, step: Double,
numOfIters: Int) : MutableList<Double>
{
    val outList: MutableList<Double> = mutableListOf()

    var curX = xStart
    var curY = yStart

    outList.add(curY)
    for (i in 1..numOfIters)
    {
        curY += step * ((1 - alpha) * curMathFun(curX, curY) +
                alpha * curMathFun(curX + step / (2 * alpha),curY + step *
curMathFun(curX, curY) / (2 * alpha)))

        outList.add(curY)

        curX += step
    }

    return outList
}

fun printHead()
{
    print(String.format("|%-${xF}s|%-${picF - 1}s|%-${eulF}s|%-${rkF}s|\n", "", "",
"", "").replace(' ', '-'))
    print(String.format("|%-${xF}s|%-${picF - 1}s|%-${eulF}s|%-${rkF}s|\n", "x",
"Метод Пикара", "Метод Эйлера", "Метод Рунге-Кутта 2-го порядка точности"))
```

```
    print(String.format("|%${xF}s|", " "))
    for (i in 1..4)
        print(String.format("%-${picFfo}s|", "").replace(' ', '-'))
    println(String.format("%${eulF}s|%${rkF}s|", "", ""))

    print(String.format("|%${xF}s|", " "))
    for (i in 1..4)
        print(String.format("%-${picFfo}s|", "Приближение $i"))
    print(String.format("%${eulF}s|%${rkF}s|\n", "", ""))

    print(String.format("|%${xF}s|", " ").replace(' ', '-'))
    for (i in 1..4)
        print(String.format("%-${picFfo}s|", "").replace(' ', '-'))
    println(String.format("%${eulF}s|%${rkF}s|", "", "").replace(' ', '-'))
}

fun printAnswers(picAnsw: MutableList<MutableList<Double>>, eulAnsw:
MutableList<Double>, runAnsw: MutableList<Double>)
{
    for (i in 0 until picAnsw.size step 100)
    {
        val curPic = picAnsw[i]
        println(
            String.format(
                "|%-${xF}f|%-
${picFfo}f|%${picFfo}f|%${picFfo}f|%${picFfo}f|%${eulF}f|%${rkF}f|",
                curPic[0], curPic[1], curPic[2], curPic[3], curPic[4], eulAnsw[i],
runAnsw[i]
            )
        )
        print(String.format("|%${xF}s|", " ").replace(' ', '-'))
        for (i in 1..4)
            print(String.format("%-${picFfo}s|", "").replace(' ', '-'))
        println(String.format("%${eulF}s|%${rkF}s|", "", "").replace(' ', '-'))
    }
}

fun main()
{
    val xStart = 0.0
    val xStop = 2.0
    val yStart = 0.0
    val step = 10e-5

    val iters = ceil(abs(xStop - xStart) / step).toInt()

    val picAnsw = picardMet(xStart, step, iters)
    val eulAnsw = eulerMet(xStart, yStart, step, iters)
    val runAnsw = rungeKutMet(xStart, yStart, 0.5, step, iters)

    printHead()
    printAnswers(picAnsw, eulAnsw, runAnsw)
}
```