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»

Отчет

к лабораторной работе №1
по курсу «Моделирование»

Студент
ИУ7-64Б Дикобаева А.Л.

Преподаватель
Градов В.М.

2021г.

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) и второго порядка точности (Рунге-Кутта).

Исходные данные.

1. ОДУ, не имеющее аналитического решения

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

Результаты работы программы

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. Границу интервала x_{max} выбирать максимально возможной из условия, чтобы численные методы обеспечивали точность вычисления решения уравнения $u(x)$ до второго знака после запятой.

Вопросы при защите лабораторной работы

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.

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

3. Каково значение функции при $x = 2$, т.е. привести значение $u(2)$.

1 Метод Пикара

Метод Пикара является представителем приближенных методов решения рассматриваемого класса задач. Идея метода сводится к процедуре последовательных приближений для решения интегрального уравнения, к которому приводится исходное дифференциальное уравнение.

Поставлена задача Коши:

$$\begin{cases} u'(x) = f(x, u(x)), \\ u(x_0) = u_0 \end{cases}$$

Проинтегрируем выписанное уравнение

$$u(x) = u_0 + \int_{x_0}^{x} f(t, u(t))\, dt$$

Процедура последовательных приближений метода Пикара реализуется согласно следующей схеме

$$y_s(x) = u_0 + \int_{x_0}^{x} f(t, y_{s-1}(t))\, dt\ ,$$

причем $y_0(t) = v_0$, (i – номер итерации).

Заданная в лабораторной работе ОДУ, не имеющее аналитического решения

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

Правая часть непрерывна и удовлетворяет условию Липшица. Значит, решение существует, а метод Пикара сойдется. По схеме Пикара рассчитаем первые четыре приближения для заданного ОДУ.

$$y_1(x) = 0 + \int_0^x t^2\, dt\ = \frac{x^3}{3}$$

$$y_2(x) = 0 + \int_0^x (t^2 + (\frac{t^3}{3})^2)\,dt \; = \frac{x^3}{3} + \frac{x^7}{63}$$

$$y_3(x) = 0 + \int_0^x (t^2 + (\frac{t^3}{3} + \frac{t^7}{63})^2)\,dt \; = \frac{x^3}{3} + \frac{x^7}{63} + \frac{2x^{11}}{2079} + \frac{x^{15}}{59535}$$

$$y_4(x) = 0 + \int_0^x (t^2 + (\frac{t^3}{3} + \frac{t^7}{63} + \frac{2t^{11}}{2079} + \frac{t^{15}}{59535})^2)\,dt \; =$$

$$= \frac{x^3}{3} + \frac{x^7}{63} + \frac{2x^{11}}{2079} + \frac{13x^{15}}{218295} + \frac{82x^{19}}{37328445} + \frac{662x^{23}}{10438212015} + \frac{4x^{27}}{3341878155} + \frac{x^{31}}{109876902975}$$

2 Метод Эйлера

Также задача может быть решена с помощью численных методов.

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

$$f(x_n, y_n) = y_n^2 + x_n^2$$

3 Метод Рунге-Кутта 2-ого порядка точности

$$y_{n+1} = y_n + h[(1-\alpha)k_1 + \alpha k_2], \tag{1}$$

где

$$k_1 = f(x_n, y_n),$$

$$k_2 = f(x_n + \frac{h}{2\alpha}, y_n + \frac{h}{2\alpha}k_1),$$

В практике расчетов используют формулу (1) при значениях $\alpha = 1$, $\alpha = \frac{1}{2}$. В лабораторной работе приняли $\alpha = \frac{1}{2}$

4 Листинг программы

```
import java.lang.Math.pow
import kotlin.math.abs

fun f(x: Double, y: Double): Double {
    return pow(x, 2.0) + pow(y, 2.0);
}

val picardOne: (x: Double) -> Double = {
        x -> pow(x, 3.0) / 3
}
val picardTwo: (x: Double) -> Double = {
        x -> picardOne(x) +
        pow(x, 7.0) / 63
}
val picardThree: (x: Double) -> Double = {
        x -> picardTwo(x) +
        2 * pow(x, 11.0) / 2079 +
        pow(x, 15.0) / 59535
}
val picardFour: (x: Double) -> Double = {
        x -> picardThree(x) +
        4 * pow(x, 15.0) / 93555 +
        82 * pow(x, 19.0) / 37328445 +
        662 * pow(x, 23.0) / 10438212015 +
        4 * pow(x, 27.0) / 3341878155 +
        pow(x, 31.0) / 109876902975
}
```

```
28
29 fun picard(columnX: ArrayList<Double>,
30             funcPickard: (x: Double)-> Double): ArrayList<Double> {
31     var columnY = arrayListOf<Double>()
32     for (x in columnX) {
33         val y = funcPickard(x)
34         columnY.add(y)
35     }
36     return  columnY
37 }
38
39 fun euler(columnX: ArrayList<Double>, h:Double): ArrayList<Double> {
40     var columnY = arrayListOf<Double>()
41     var y = 0.0
42     for (x in columnX) {
43         y += h * f(x, y)
44         columnY.add(y)
45     }
46     return  columnY
47 }
48
49 fun runge(columnX: ArrayList<Double>, h:Double): ArrayList<Double> {
50     var columnY = arrayListOf<Double>()
51     var y = 0.0
52     val alpha = 0.5
53
54     for (x in columnX) {
55         val k1 = f(x, y)
56         val k2 = f(x + h / 2 / alpha, y + k1 * h / 2 / alpha)
57         y += h * ((1 - alpha) * k1 + alpha * k2)
58         columnY.add(y)
59     }
60     return  columnY
61 }
62
63 fun printColumns(table: ArrayList<ArrayList<Double>>, eps:Double) {
64     var i = 0
65     while (abs(table[6][i] - table[5][i]) <= eps) {
66         print("| %4.3f    ".format(table[0][i]))
67
68         for (j in 1..(table.size - 1)) {
69             print("| %7.6f     ".format(table[j][i]))
70         }
71         println("| %4.3f        |".format(abs(table[6][i] - table[5][i])))
72         i++
73     }
74     println("-".repeat(109))
75 }
76
77 fun printHead() {
78     var line = "-".repeat(109)
79     println(line)
80     println("|          |                                Метод Пикара                          " +
81             "|  Метод       |     Метод     |                |")
82     println("|    x     |       1       |       2       |       3       |       4       " +
83             "|  Эйлера      | РунгеКутта- |  Точность    |")
84     println(line)
85
86 }
87
88 fun printTable(table: ArrayList<ArrayList<Double>>, eps: Double) {
89     printHead()
90     printColumns(table, eps)
91
92 }
93
94 fun getX(firstX: Double, maxX: Double, h: Double): ArrayList<Double> {
95     var columnX = arrayListOf<Double>()
96     var x = firstX
97     while(x <= maxX) {
```

```
            columnX.add(x)
            x += h
        }
        return columnX
}

fun main() {
        val firstX = 0.0
        val maxX = 2.0
        val h = 0.01
        val eps = 0.01

        println("Начальный x = " + firstX)
        println("Максимальный x = " + maxX)
        println("Шаг для численных методов = " + h)
        println("Точность Eps = " + eps)

        var table = arrayListOf<ArrayList<Double>>()
        val columnX = getX(firstX, maxX, h)
        table.add(columnX)
        table.add(picard(columnX, picardOne))
        table.add(picard(columnX, picardTwo))
        table.add(picard(columnX, picardThree))
        table.add(picard(columnX, picardFour))
        table.add(euler(columnX, h))
        table.add(runge(columnX, h))
        printTable(table, eps)
}
```

5 Результаты работы программы

На рисунках представлены результаты работы программы. Вывод таблицы осуществляется, пока численные методы обеспечивают точность вычисления решения уравнения $u(x)$ до второго знака после запятой.

```
Начальный x = 0.0
Максимальный x = 2.0
Шаг для численных методов = 0.01
Точность Eps = 0.01
```

	Метод Пикара				Метод	Метод	
x	1	2	3	4	Эйлера	Рунге-Кутта	Точность
0,000	0,000000	0,000000	0,000000	0,000000	0,000000	0,000001	0,000
0,010	0,000000	0,000000	0,000000	0,000000	0,000001	0,000003	0,000
0,020	0,000003	0,000003	0,000003	0,000003	0,000005	0,000010	0,000
0,030	0,000009	0,000009	0,000009	0,000009	0,000014	0,000022	0,000
0,040	0,000021	0,000021	0,000021	0,000021	0,000030	0,000043	0,000
0,050	0,000042	0,000042	0,000042	0,000042	0,000055	0,000073	0,000
0,060	0,000072	0,000072	0,000072	0,000072	0,000091	0,000116	0,000
0,070	0,000114	0,000114	0,000114	0,000114	0,000140	0,000172	0,000
0,080	0,000171	0,000171	0,000171	0,000171	0,000204	0,000245	0,000
0,090	0,000243	0,000243	0,000243	0,000243	0,000285	0,000335	0,000
0,100	0,000333	0,000333	0,000333	0,000333	0,000385	0,000446	0,000
0,110	0,000444	0,000444	0,000444	0,000444	0,000506	0,000578	0,000
0,120	0,000576	0,000576	0,000576	0,000576	0,000650	0,000735	0,000
0,130	0,000732	0,000732	0,000732	0,000732	0,000819	0,000917	0,000
0,140	0,000915	0,000915	0,000915	0,000915	0,001015	0,001128	0,000
0,150	0,001125	0,001125	0,001125	0,001125	0,001240	0,001368	0,000
0,160	0,001365	0,001365	0,001365	0,001365	0,001496	0,001641	0,000
0,170	0,001638	0,001638	0,001638	0,001638	0,001785	0,001947	0,000
0,180	0,001944	0,001944	0,001944	0,001944	0,002109	0,002290	0,000
0,190	0,002286	0,002286	0,002286	0,002286	0,002470	0,002670	0,000
0,200	0,002667	0,002667	0,002667	0,002667	0,002870	0,003091	0,000
0,210	0,003087	0,003087	0,003087	0,003087	0,003311	0,003553	0,000
0,220	0,003549	0,003550	0,003550	0,003550	0,003795	0,004060	0,000
0,230	0,004056	0,004056	0,004056	0,004056	0,004325	0,004613	0,000
0,240	0,004608	0,004609	0,004609	0,004609	0,004901	0,005213	0,000
0,250	0,005208	0,005209	0,005209	0,005209	0,005526	0,005864	0,000
0,260	0,005859	0,005860	0,005860	0,005860	0,006202	0,006567	0,000

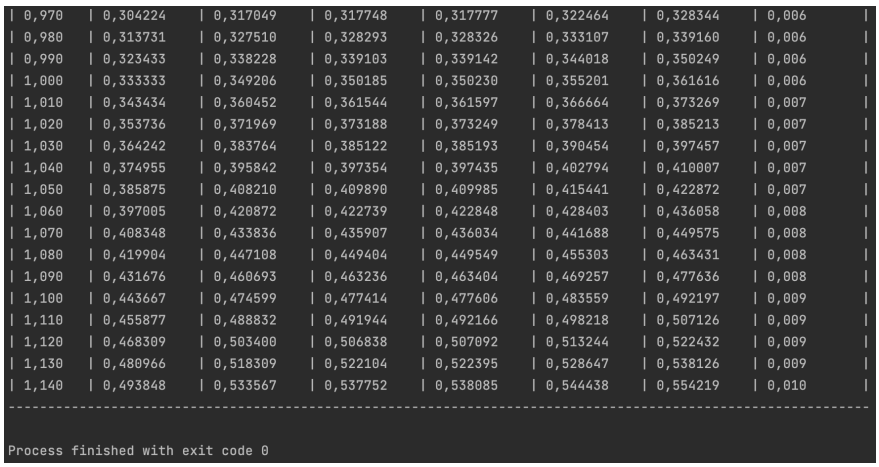
Рис. 1: Результаты работы программы

0,270	0,006561	0,006563	0,006563	0,006563	0,006932	0,007324	0,000
0,280	0,007317	0,007319	0,007319	0,007319	0,007716	0,008137	0,000
0,290	0,008130	0,008132	0,008132	0,008132	0,008558	0,009008	0,000
0,300	0,009000	0,009003	0,009003	0,009003	0,009458	0,009940	0,000
0,310	0,009930	0,009935	0,009935	0,009935	0,010420	0,010933	0,001
0,320	0,010923	0,010928	0,010928	0,010928	0,011445	0,011991	0,001
0,330	0,011979	0,011986	0,011986	0,011986	0,012536	0,013115	0,001
0,340	0,013101	0,013110	0,013110	0,013110	0,013693	0,014308	0,001
0,350	0,014292	0,014302	0,014302	0,014302	0,014920	0,015570	0,001
0,360	0,015552	0,015564	0,015564	0,015564	0,016218	0,016906	0,001
0,370	0,016884	0,016899	0,016899	0,016899	0,017590	0,018315	0,001
0,380	0,018291	0,018309	0,018309	0,018309	0,019037	0,019801	0,001
0,390	0,019773	0,019795	0,019795	0,019795	0,020562	0,021366	0,001
0,400	0,021333	0,021359	0,021359	0,021359	0,022166	0,023011	0,001
0,410	0,022974	0,023005	0,023005	0,023005	0,023852	0,024740	0,001
0,420	0,024696	0,024733	0,024733	0,024733	0,025622	0,026553	0,001
0,430	0,026502	0,026545	0,026546	0,026546	0,027477	0,028453	0,001
0,440	0,028395	0,028445	0,028445	0,028445	0,029421	0,030442	0,001
0,450	0,030375	0,030434	0,030434	0,030434	0,031454	0,032522	0,001
0,460	0,032445	0,032515	0,032515	0,032515	0,033580	0,034696	0,001
0,470	0,034608	0,034688	0,034688	0,034688	0,035801	0,036966	0,001
0,480	0,036864	0,036957	0,036957	0,036957	0,038117	0,039333	0,001
0,490	0,039216	0,039324	0,039324	0,039324	0,040533	0,041800	0,001
0,500	0,041667	0,041791	0,041791	0,041791	0,043049	0,044369	0,001
0,510	0,044217	0,044359	0,044360	0,044360	0,045669	0,047042	0,001
0,520	0,046869	0,047033	0,047033	0,047033	0,048394	0,049822	0,001
0,530	0,049626	0,049812	0,049813	0,049813	0,051226	0,052711	0,001
0,540	0,052488	0,052701	0,052702	0,052702	0,054168	0,055711	0,002
0,550	0,055458	0,055700	0,055701	0,055701	0,057223	0,058824	0,002
0,560	0,058539	0,058813	0,058814	0,058814	0,060391	0,062053	0,002
0,570	0,061731	0,062041	0,062043	0,062043	0,063677	0,065400	0,002
0,580	0,065037	0,065388	0,065390	0,065390	0,067081	0,068868	0,002
0,590	0,068460	0,068855	0,068858	0,068858	0,070607	0,072458	0,002
0,600	0,072000	0,072444	0,072448	0,072448	0,074257	0,076174	0,002
0,610	0,075660	0,076159	0,076163	0,076163	0,078033	0,080017	0,002

Рис. 2: Результаты работы программы (продолжение)

0,620	0,079443	0,080002	0,080007	0,080007	0,081938	0,083991	0,002
0,630	0,083349	0,083974	0,083980	0,083980	0,085975	0,088097	0,002
0,640	0,087381	0,088079	0,088087	0,088087	0,090144	0,092339	0,002
0,650	0,091542	0,092320	0,092328	0,092328	0,094451	0,096719	0,002
0,660	0,095832	0,096698	0,096708	0,096708	0,098896	0,101240	0,002
0,670	0,100254	0,101216	0,101228	0,101228	0,103483	0,105903	0,002
0,680	0,104811	0,105878	0,105892	0,105892	0,108214	0,110713	0,002
0,690	0,109503	0,110685	0,110701	0,110701	0,113092	0,115672	0,003
0,700	0,114333	0,115641	0,115660	0,115660	0,118120	0,120782	0,003
0,710	0,119304	0,120747	0,120770	0,120770	0,123300	0,126047	0,003
0,720	0,124416	0,126008	0,126034	0,126035	0,128636	0,131469	0,003
0,730	0,129672	0,131426	0,131456	0,131457	0,134131	0,137052	0,003
0,740	0,135075	0,137003	0,137039	0,137039	0,139787	0,142798	0,003
0,750	0,140625	0,142744	0,142785	0,142785	0,145607	0,148711	0,003
0,760	0,146325	0,148650	0,148697	0,148698	0,151595	0,154794	0,003
0,770	0,152178	0,154725	0,154780	0,154781	0,157754	0,161049	0,003
0,780	0,158184	0,160972	0,161035	0,161036	0,164087	0,167482	0,003
0,790	0,164346	0,167395	0,167467	0,167468	0,170597	0,174094	0,003
0,800	0,170667	0,173995	0,174079	0,174080	0,177288	0,180889	0,004
0,810	0,177147	0,180778	0,180874	0,180876	0,184163	0,187872	0,004
0,820	0,183789	0,187746	0,187856	0,187858	0,191227	0,195045	0,004
0,830	0,190596	0,194903	0,195028	0,195031	0,198481	0,202412	0,004
0,840	0,197568	0,202252	0,202395	0,202398	0,205931	0,209978	0,004
0,850	0,204708	0,209797	0,209959	0,209963	0,213580	0,217746	0,004
0,860	0,212019	0,217541	0,217726	0,217731	0,221432	0,225720	0,004
0,870	0,219501	0,225489	0,225699	0,225705	0,229492	0,233904	0,004
0,880	0,227157	0,233644	0,233882	0,233889	0,237762	0,242303	0,005
0,890	0,234990	0,242011	0,242280	0,242288	0,246249	0,250922	0,005
0,900	0,243000	0,250592	0,250897	0,250906	0,254955	0,259764	0,005
0,910	0,251190	0,259393	0,259738	0,259749	0,263886	0,268835	0,005
0,920	0,259563	0,268417	0,268807	0,268819	0,273046	0,278140	0,005
0,930	0,268119	0,277670	0,278108	0,278123	0,282441	0,287683	0,005
0,940	0,276861	0,287155	0,287648	0,287666	0,292075	0,297469	0,005
0,950	0,285792	0,296876	0,297431	0,297452	0,301953	0,307504	0,006
0,960	0,294912	0,306840	0,307463	0,307487	0,312081	0,317794	0,006

Рис. 3: Результаты работы программы (продолжение)



0,970	0,304224	0,317049	0,317748	0,317777	0,322464	0,328344	0,006
0,980	0,313731	0,327510	0,328293	0,328326	0,333107	0,339160	0,006
0,990	0,323433	0,338228	0,339103	0,339142	0,344018	0,350249	0,006
1,000	0,333333	0,349206	0,350185	0,350230	0,355201	0,361616	0,006
1,010	0,343434	0,360452	0,361544	0,361597	0,366664	0,373269	0,007
1,020	0,353736	0,371969	0,373188	0,373249	0,378413	0,385213	0,007
1,030	0,364242	0,383764	0,385122	0,385193	0,390454	0,397457	0,007
1,040	0,374955	0,395842	0,397354	0,397435	0,402794	0,410007	0,007
1,050	0,385875	0,408210	0,409890	0,409985	0,415441	0,422872	0,007
1,060	0,397005	0,420872	0,422739	0,422848	0,428403	0,436058	0,008
1,070	0,408348	0,433836	0,435907	0,436034	0,441688	0,449575	0,008
1,080	0,419904	0,447108	0,449404	0,449549	0,455303	0,463431	0,008
1,090	0,431676	0,460693	0,463236	0,463404	0,469257	0,477636	0,008
1,100	0,443667	0,474599	0,477414	0,477606	0,483559	0,492197	0,009
1,110	0,455877	0,488832	0,491944	0,492166	0,498218	0,507126	0,009
1,120	0,468309	0,503400	0,506838	0,507092	0,513244	0,522432	0,009
1,130	0,480966	0,518309	0,522104	0,522395	0,528647	0,538126	0,009
1,140	0,493848	0,533567	0,537752	0,538085	0,544438	0,554219	0,010

Process finished with exit code 0

Рис. 4: Результаты работы программы (продолжение)

6 Ответы на 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.**

   Каждое следующее приближение Пикара точнее предыдущего. n-ое приближение Пикара можно считать решением уравнения, если выполняется условие

   $$|y_n(x) - y_{n+1}(x)| < \epsilon,$$

   Для вычисления интервала 4-ого приближения нужно рассчитать 5-ое приближение.
   Листинг программы:

```
fun main() {
    val firstX = 0.0
    val maxX = 2.0
    val h = 0.001
    val eps = 0.01

    var table = arrayListOf<ArrayList<Double>>()
    val columnX = getX(firstX, maxX, h)
    table.add(columnX)
    table.add(picard(columnX, picardOne))
    table.add(picard(columnX, picardTwo))
    table.add(picard(columnX, picardThree))
    table.add(picard(columnX, picardFour))

    var i = 0
    while(abs(table[2][i] - table[1][i]) <= eps) i++
    println("Для 1- ого приближения x принадлежит [0, %4.3f".format(table[0][i]) + "]")
    while(abs(table[3][i] - table[2][i]) <= eps) i++
    println("Для 2- ого приближения x принадлежит [0, %4.3f".format(table[0][i]) + "]")
    while(abs(table[4][i] - table[3][i]) <= eps) i++
    println("Для 3- его приближения x принадлежит [0, %4.3f".format(table[0][i]) + "]")
}

```

   Результаты работы программы: При заданной точности $\epsilon = 0.01$ n-е приближение Пикара является решением, когда x принадлежит интервалам:

```
Для 1-ого приближения x принадлежит [0, 0,937]
Для 2-ого приближения x принадлежит [0, 1,233]
Для 3-его приближения x принадлежит [0, 1,419]
```

Рис. 5: Результаты работы программы (задание 1)

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

   Точность численных методов зависит от шага, поэтому чтобы доказать правильность полученного результата при фиксированном значении аргумента в численных методах, нужно сравнить полученное значение со значением, полученном при меньшем шаге.

3. **Каково значение функции при $x = 2$, т.е. привести значение $u(2)$.**

   По схеме ответа на вопрос №2 рассчитаем значение $u(2)$

   $u(2) = 317.72$

```
fun getEuler(firstX:Double, maxX:Double, firstY: Double, h:Double):Double {
    var x = firstX
    var y = firstY
    while (x <= maxX) {
        y += h * f(x, y)
        x += h
    }
    return  y
}

fun main(){
    val firstX = 0.0
    val maxX = 2.0
    var h = 0.1
    val eps = 0.01
    val firstY = 0.0

    var curY = 0.0
    var nextY = getEuler(firstX, maxX, firstY, h)
    println("     h       |       u(2)      |  точность")
    do {
        curY = nextY
        h *= 0.1
        nextY = getEuler(firstX, maxX, firstY, h)
        println("%1.9f | %13.9f | %1.3f".format(h*10, curY, abs(nextY - curY)))
    } while(abs(curY - nextY) > eps)

    println("u(2) = " + curY)
}

```

```
     h      |      u(2)     |  точность
0,100000000 |   5,852099612 | 22,540
0,010000000 |  28,392533753 | 114,235
0,001000000 | 142,627249698 | 134,735
0,000100000 | 277,362500130 | 34,699
0,000010000 | 312,061276403 | 5,185
0,000001000 | 317,245934954 | 0,419
0,000000100 | 317,664629635 | 0,053
0,000000010 | 317,717741760 | 0,002
u(2) = 317.7177417599254
```

Рис. 6: Результаты работы программы (задание 3)