



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

по Лабораторной работе

по курсу «Моделирование»

на тему: «Программная реализация приближенного аналитического метода и
численных алгоритмов первого и второго порядков точности при решении
задачи Коши для ОДУ»

Студент ИУ7-63Б
(Группа)

(Подпись, дата)

Миронов Г. А.
(И. О. Фамилия)

Преподаватель

(Подпись, дата)

Градов В. М.
(И. О. Фамилия)

2022 г.

СОДЕРЖАНИЕ

Введение	3
1 Теоритическая часть	4
1.1 Постановка задачи Коши	4
1.2 Описание численных методов решения задачи Коши	4
1.2.1 Метод Пикара	4
1.2.2 Метод Эйлера	5
1.2.3 Неявный метод Эйлера	5
1.2.4 Метод Рунге-Кутта	5
2 Технологическая часть	6
2.1 Выбор средств реализации	6
2.2 Требования к программному обеспечению	6
2.3 Сведения о модулях программы	6
2.4 Реализация алгоритмов	7
2.5 Вывод	12
3 Результаты работы программного обеспечения	13
3.1 Пример работы	13
3.2 Результаты работы	13
3.3 Вывод	17
4 Контрольные вопросы	18
4.1 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.	18
4.2 Пояснить, каким образом можно доказать правильность полученного результата при фиксированном значении аргумента в численных методах.	18
4.3 Каково значение функции при $x=2$, т.е. привести значение $u(2)$	19
Заключение	20

Введение

Моделирование — специфический метод (методология) познания, основанный на замене реального объекта (системы, процесса, ситуации, явления) некоторой моделью и исследовании в дальнейшем созданной модели.

Целью данной лабораторной работы является получение навыков решения задачи Коши при помощи методов Пикара, Эйлера и Рунге–Кутты 2-го порядка.

Для достижения данной цели необходимо решить следующие задачи:

- описать постановку задачи Коши;
- привести конкретный пример задачи Коши, рассматриваемый в рамках лабораторной работы;
- описать соответствующие методы решения задачи:
 - метод Пикара;
 - явный метод Эйлера;
 - неявный метод Эйлера;
 - метод Рунге–Кутты 2-го порядка;
- описать структуру разрабатываемого ПО;
- определить средства программной реализации;
- реализовать соответствующее ПО;
- привести результаты работы разработанного ПО.

1 Теоритическая часть

В данном разделе рассматриваются существующие методы решения задачи Коши.

1.1 Постановка задачи Коши

Прежде чем описывать методы решения задачи, требуется провести постановку соответствующей задачи.

Имеется ОДУ, у которого отсутствует аналитическое решение:

$$\begin{cases} u'(x) = f(x, u) \\ u(\xi) = \eta \end{cases} \quad (1.1)$$

Требуется найти решение данного ОДУ.

В рамках данной лабораторной работы будет решаться следующая задача:

$$\begin{cases} u'(x) = x^2 + u^2 \\ u(0) = 0 \end{cases} \quad (1.2)$$

1.2 Описание численных методов решения задачи Коши

1.2.1 Метод Пикара

Из условия задачи Коши получаем:

$$u(x) = \eta + \int_{\xi}^x \phi(t, u(t)) dt \quad (1.3)$$

Построим ряд функций:

$$y^{(s)} = \eta + \int_{\xi}^x \phi(t, y^{(s-1)}(t)) dt, \quad y^{(0)} = \eta \quad (1.4)$$

Построим первые 4 приближения для уравнения 1.3:

$$y^{(1)}(x) = 0 + \int_0^x (t^2 + 0^2) dt = \frac{x^3}{3} \quad (1.5)$$

$$y^{(2)}(x) = 0 + \int_0^x (t^2 + [\frac{t^3}{3}]^2) dt = \frac{x^3}{3} + \frac{x^7}{63} \quad (1.6)$$

$$y^{(3)}(x) = 0 + \int_0^x (t^2 + [\frac{t^3}{3} + \frac{t^7}{63}]^2) dt = \frac{x^3}{3} + \frac{x^7}{63} + \frac{2x^{11}}{2079} + \frac{x^{15}}{59535} \quad (1.7)$$

$$\begin{aligned} y^{(4)}(x) &= 0 + \int_0^x (t^2 + [\frac{t^3}{3} + \frac{t^7}{63} + \frac{2t^{11}}{2079} + \frac{t^{15}}{59535}]^2) dt = \\ &= \frac{x^3}{3} + \frac{x^7}{63} + \frac{2x^{11}}{2079} + \frac{13x^{15}}{218295} + \frac{82x^{19}}{37328445} + \\ &+ \frac{662x^{23}}{10438212015} + \frac{4x^{27}}{3341878155} + \frac{x^{31}}{109876902975} \end{aligned} \quad (1.8)$$

1.2.2 Метод Эйлера

$$y^{(n+1)}(x) = y^{(n)}(x) + h \cdot f(x_n, y^{(n)}) \quad (1.9)$$

Порядок точности: $O(h)$.

1.2.3 Неявный метод Эйлера

$$y^{(n+1)}(x) = y^{(n)}(x) + h \cdot f(x_{n+1}, y^{(n+1)}) \quad (1.10)$$

1.2.4 Метод Рунге-Кутта

$$y^{n+1}(x) = y^n(x) + h((1 - \alpha)R_1 + \alpha R_2) \quad (1.11)$$

где $R_1 = f(x_n, y^n)$, $R_2 = f(x_n + \frac{h}{2\alpha}, y^n + \frac{h}{2\alpha}R_1)$, $\alpha = \frac{1}{2}$ или 1

Порядок точности: $O(h^2)$.

2 Технологическая часть

В данном разделе будут представлены средства разработки программного обеспечения и детали реализации разрабатываемого программного обеспечения.

2.1 Выбор средств реализации

В качестве языка программирования для разработки программного обеспечения был выбран язык **Golang**. Данный выбор обусловлен тем, что данный язык предоставляет весь функционал требуемый для решения поставленной задачи, кроме того, я имею опыт разработки на данном языке.

2.2 Требования к программному обеспечению

К программе предъявляется ряд требований:

- На вход подаётся интервал для проведения вычислений, а так же - шаг вычисления.
- На выходе — таблица с результатами выполнения каждого из вышеуказанных алгоритмов.

2.3 Сведения о модулях программы

Данная программа разбита на следующие модули:

- `main.go` – Файл, содержащий точку входа в программу. В нем происходит общение с пользователем и вызов алгоритмов.
- `log.go` – Файл содержит функции для вывода информации.
- `diff_equation.go` – Файл содержит реализацию ОДУ, указанного в задаче.
- `picard.go` – Файл содержит реализацию метода Пикарда.
- `euler.go` – Файл содержит реализацию методов Эйлера.
- `runge_kutt.go` – Файл содержит реализацию метода Рунге-Кутты.
- `misc.go` – Файл содержит различные функции для вычислений.

2.4 Реализация алгоритмов

В листингах 2.1 – 2.8 представлены исходные коды реализуемого программного обеспечения.

Листинг 2.1 – Файл `main.go`. Реализация точки входа в программу

```
1 func process(f Flags) [][]float64 {
2     n := int(math.Ceil(math.Abs(*f.to-*f.from) / *f.h))
3
4     result := make([][]float64, 0)
5     result = append(result, solvers.InitX(*f.from, *f.h, n))
6
7     result = append(result,
8         solvers.NewPicard().Solution(*f.from, *f.h, n)...)
9     result = append(result,
10        solvers.NewEuler().Solution(*f.from, *f.y0, *f.h, n))
11    result = append(result,
12        solvers.NewEuler().ImplSolution(*f.from, *f.y0, *f.h, n))
13    result = append(result,
14        solvers.NewRK().Solution(*f.from, *f.y0, 0.5, *f.h, n))
15
16    return result
17 }
18
19 func main() {
20     f := inputArgs()
21
22     result := process(f)
23
24     logger.Log(solvers.RotateMtrx(result), *f.each)
25     if *f.genOutput {
26         logger.LogToCsv(solvers.RotateMtrx(result), *f.
27             outputFilename)
28     }
29 }
```

Листинг 2.2 – Файл `picard.go`. Реализация метода Пикара

```

1 type Picard struct {
2 }
3
4 func NewPicard() Picard {
5     return Picard{}
6 }
7
8 func (p Picard) Solution(x0, h float64, n int) [][]float64 {
9     r := make([][]float64, 4)
10    for i := 0; i < 4; i++ {
11        r[i] = make([]float64, 0)
12    }
13
14    for i := 0; i <= n; i++ {
15        r[0] = append(r[0], p.approx1(x0))
16        r[1] = append(r[1], p.approx2(x0))
17        r[2] = append(r[2], p.approx3(x0))
18        r[3] = append(r[3], p.approx4(x0))
19
20        x0 += h
21    }
22
23    return r
24 }
25
26 func (p Picard) approx1(x float64) float64 {
27     return x * x * x / 3
28 }
29
30 func (p Picard) approx2(x float64) float64 {
31     return p.approx1(x) + math.Pow(x, 7)/63
32 }
33
34 func (p Picard) approx3(x float64) float64 {
35     return p.approx2(x) + 2*math.Pow(x, 11)/2079 + math.Pow(x,
36         15)/59535
37 }
38
39 func (p Picard) approx4(x float64) float64 {
40     return p.approx2(x) + 2*math.Pow(x, 11)/2079 + 13*math.Pow(x,
41         15)/218295 +
42         82*math.Pow(x, 19)/37328445 + 662*math.Pow(x, 23)
43         /10438212015 +
44         4*math.Pow(x, 27)/3341878155 + math.Pow(x, 31)
45         /109876902975
46 }

```


Листинг 2.3 – Файл `diff_equation.go`. Реализация ОДУ из условия

```
1 func equation(x, u float64) float64 {  
2     return x*x + u*u  
3 }
```

Листинг 2.4 – Файл `euler.go`. Реализация методов Эйлера

```
1 type Euler struct {  
2 }  
3  
4 func NewEuler() Euler {  
5     return Euler{}  
6 }  
7  
8 func (e Euler) Solution(x0, y0, h float64, n int) []float64 {  
9     r := make([]float64, 0)  
10  
11     r = append(r, y0)  
12  
13     for i := 0; i < n; i++ {  
14         y0 += h * equation(x0, y0)  
15         x0 += h  
16  
17         r = append(r, y0)  
18     }  
19  
20     return r  
21 }  
22  
23 func (e Euler) ImplSolution(x0, y0, h float64, n int) []float64 {  
24     r := make([]float64, 0)  
25  
26     r = append(r, y0)  
27     for i := 0; i < n; i++ {  
28         D := 1 - 4*h*(y0+h*(x0+h)*(x0+h))  
29  
30         y0 = (1 - math.Sqrt(D)) / 2 / h  
31         x0 += h  
32  
33         r = append(r, y0)  
34     }  
35  
36     return r  
37 }
```

Листинг 2.5 – Файл `runge_kutta.go`. Реализация метода Рунге–Кутты

```
1 func (rk RungeKutta) Solution(x0, y0, a, h float64, n int) []  
   float64 {  
2     r := make([]float64, 0)  
3  
4     for i := 0; i <= n; i++ {  
5       r = append(r, y0)  
6       y0 += h * ((1-a)*equation(x0, y0) + a*equation(x0+h/2/a,  
7         y0+h*equation(x0, y0)/2/a))  
8       x0 += h  
9     }  
10    return r  
11 }
```

Листинг 2.6 – Файл `misc.go`. Реализация различных функций

```
1 func InitX(x0, h float64, n int) []float64 {  
2   x := make([]float64, 0)  
3   for i := 0; i <= n; i++ {  
4     x = append(x, x0)  
5     x0 += h  
6   }  
7   return x  
8 }  
9  
10 func RotateMtrx(matrix [][]float64) [][]interface{} {  
11   result := make([][]interface{}, len(matrix[0]))  
12   for i := 0; i < len(result); i++ {  
13     result[i] = make([]interface{}, 0)  
14   }  
15  
16   for _, row := range matrix {  
17     for j, v := range row {  
18       result[j] = append(result[j], fmt.Sprintf("%.6g", v))  
19     }  
20   }  
21  
22   return result  
23 }
```

Листинг 2.7 – Файл log.go. Реализация функций для вывода информации.
Часть 1

```
1 var (
2     headers = []interface{}{
3         "X",
4         "Picard, 1st approx",
5         "Picard, 2nd approx",
6         "Picard, 3rd approx",
7         "Picard, 4th approx",
8         "Euler",
9         "Implicit Euler",
10        "Runge-Kutta",
11    }
12 )
13
14 func Log(matrix [][]interface{}, each int64) {
15     headerFmt := color.New(color.FgGreen, color.Underline)
16     columnFmt := color.New(color.FgYellow)
17
18     tbl := table.New(headers...)
19     tbl.WithHeaderFormatter(headerFmt.SprintfFunc())
20     tbl.WithFirstColumnFormatter(columnFmt.SprintfFunc())
21
22     for i, row := range matrix {
23         if i%int(each) != 0 {
24             continue
25         }
26
27         tbl.AddRow(row...)
28     }
29
30     tbl.Print()
31 }
32
33 func LogToCsv(matrix [][]interface{}, filename string) {
34     toString := func(slice []interface{}) []string {
35         strArray := make([]string, len(slice))
36         for i, arg := range slice {
37             strArray[i] = arg.(string)
38         }
39         return strArray
40     }
41     file, err := os.Create(filename)
42     checkError("Cannot create file", err)
43     defer file.Close()
```

Листинг 2.8 – Файл log.go. Реализация функций для вывода информации.
Часть 2

```
1
2     writer := csv.NewWriter(file)
3     defer writer.Flush()
4
5     matrix = append([][] interface{}{headers}, matrix...)
6     for _, row := range matrix {
7         err := writer.Write(toString(row))
8         checkError("Cannot write to file", err)
9     }
10 }
11
12 func checkError(message string, err error) {
13     if err != nil {
14         log.Fatal(message, err)
15     }
```

2.5 Вывод

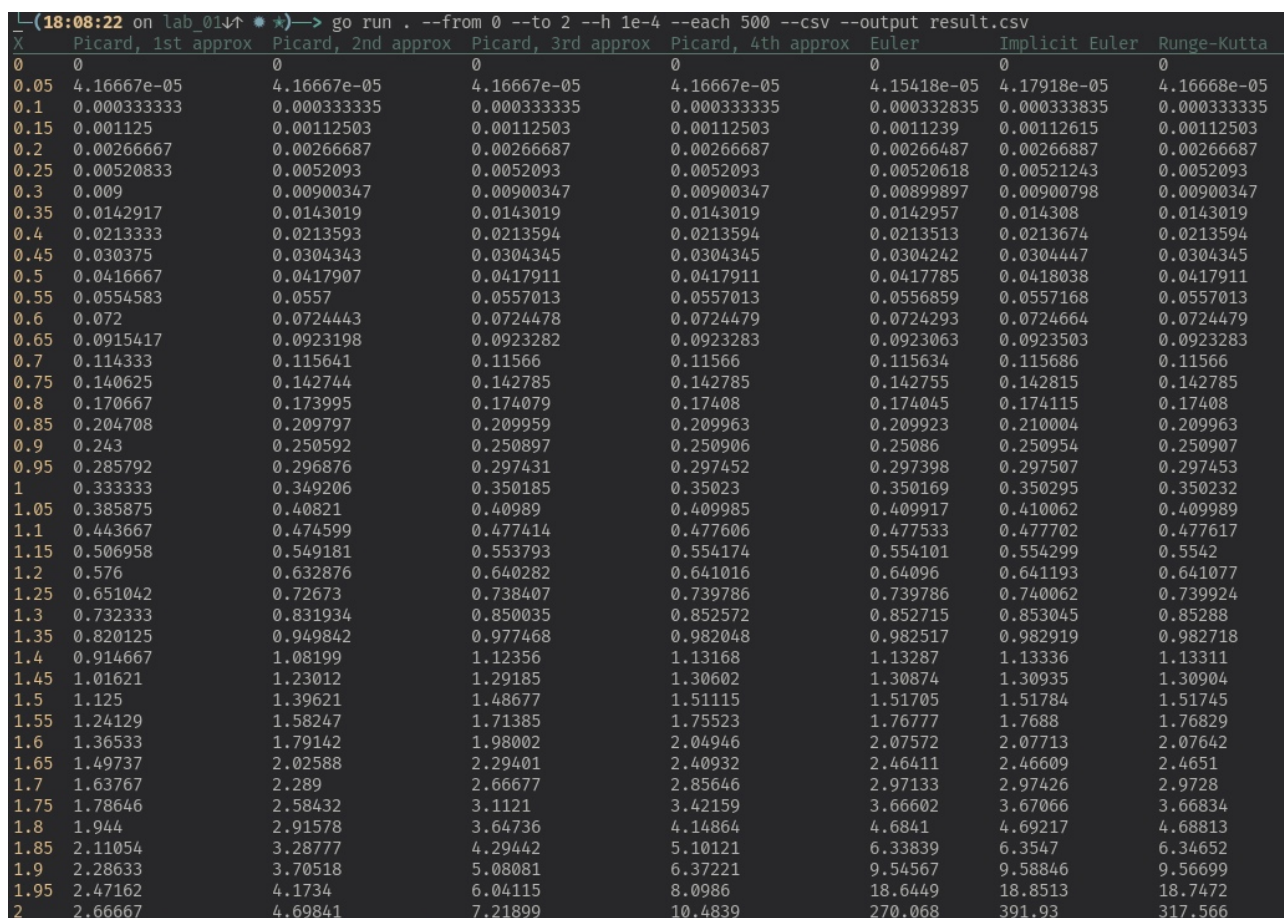
В данном разделе были представлены средства разработки программного обеспечения и детали реализации разрабатываемого программного обеспечения.

3 Результаты работы программного обеспечения

В данном разделе будут представлены результаты работы разработанного программного обеспечения.

3.1 Пример работы

На Рисунке 3.1 представлен результат работы реализованного программного обеспечения. Значения вычислены для шага 10^{-4} , выведено каждое 500 значение.



X	Picard, 1st approx	Picard, 2nd approx	Picard, 3rd approx	Picard, 4th approx	Euler	Implicit Euler	Runge-Kutta
0	0	0	0	0	0	0	0
0.05	4.16667e-05	4.16667e-05	4.16667e-05	4.16667e-05	4.15418e-05	4.17918e-05	4.16668e-05
0.1	0.000333333	0.000333335	0.000333335	0.000333335	0.000332835	0.000333835	0.000333335
0.15	0.001125	0.00112503	0.00112503	0.00112503	0.0011239	0.00112615	0.00112503
0.2	0.00266667	0.00266687	0.00266687	0.00266687	0.00266487	0.00266887	0.00266687
0.25	0.00520833	0.0052093	0.0052093	0.0052093	0.00520618	0.00521243	0.0052093
0.3	0.009	0.00900347	0.00900347	0.00900347	0.00899897	0.00900798	0.00900347
0.35	0.0142917	0.0143019	0.0143019	0.0143019	0.0142957	0.014308	0.0143019
0.4	0.0213333	0.0213593	0.0213594	0.0213594	0.0213513	0.0213674	0.0213594
0.45	0.030375	0.0304343	0.0304345	0.0304345	0.0304242	0.0304447	0.0304345
0.5	0.0416667	0.0417907	0.0417911	0.0417911	0.0417785	0.0418038	0.0417911
0.55	0.0554583	0.0557	0.0557013	0.0557013	0.0556859	0.0557168	0.0557013
0.6	0.072	0.0724443	0.0724478	0.0724479	0.0724293	0.0724664	0.0724479
0.65	0.0915417	0.0923198	0.0923282	0.0923283	0.0923063	0.0923503	0.0923283
0.7	0.114333	0.115641	0.11566	0.11566	0.115634	0.115686	0.11566
0.75	0.140625	0.142744	0.142785	0.142785	0.142755	0.142815	0.142785
0.8	0.170667	0.173995	0.174079	0.17408	0.174045	0.174115	0.17408
0.85	0.204708	0.209797	0.209959	0.209963	0.209923	0.210004	0.209963
0.9	0.243	0.250592	0.250897	0.250906	0.25086	0.250954	0.250907
0.95	0.285792	0.296876	0.297431	0.297452	0.297398	0.297507	0.297453
1	0.333333	0.349206	0.350185	0.35023	0.350169	0.350295	0.350232
1.05	0.385875	0.40821	0.40989	0.409985	0.409917	0.410062	0.409989
1.1	0.443667	0.474599	0.477414	0.477606	0.477533	0.477702	0.477617
1.15	0.506958	0.549181	0.553793	0.554174	0.554101	0.554299	0.5542
1.2	0.576	0.632876	0.640282	0.641016	0.64096	0.641193	0.641077
1.25	0.651042	0.72673	0.738407	0.739786	0.739786	0.740062	0.739924
1.3	0.732333	0.831934	0.850035	0.852572	0.852715	0.853045	0.85288
1.35	0.820125	0.949842	0.977468	0.982048	0.982517	0.982919	0.982718
1.4	0.914667	1.08199	1.12356	1.13168	1.13287	1.13336	1.13311
1.45	1.01621	1.23012	1.29185	1.30602	1.30874	1.30935	1.30904
1.5	1.125	1.39621	1.48677	1.51115	1.51705	1.51784	1.51745
1.55	1.24129	1.58247	1.71385	1.75523	1.76777	1.7688	1.76829
1.6	1.36533	1.79142	1.98002	2.04946	2.07572	2.07713	2.07642
1.65	1.49737	2.02588	2.29401	2.40932	2.46411	2.46609	2.4651
1.7	1.63767	2.289	2.66677	2.85646	2.97133	2.97426	2.9728
1.75	1.78646	2.58432	3.1121	3.42159	3.66602	3.67066	3.66834
1.8	1.944	2.91578	3.64736	4.14864	4.6841	4.69217	4.68813
1.85	2.11054	3.28777	4.29442	5.10121	6.33839	6.3547	6.34652
1.9	2.28633	3.70518	5.08081	6.37221	9.54567	9.58846	9.56699
1.95	2.47162	4.1734	6.04115	8.0986	18.6449	18.8513	18.7472
2	2.66667	4.69841	7.21899	10.4839	270.068	391.93	317.566

Рисунок 3.1 – Пример работы программы

3.2 Результаты работы

В таблице 3.1 приведены значения работы реализованного программного обеспечения. Значения вычислены для шага 10^{-4} , выведено каждое 500 значение.

Таблица 3.1 – Вычисленные значения для различных методов.

X	Методы решения									
	Пикарда (приближение)				Эйлера		Рунге-Кутта			
	1	2	3	4	Явный	Неявный				
0	0	0	0	0	0	0	0	0		
0.05	4.16667e-05	4.16667e-05	4.16667e-05	4.16667e-05	4.15418e-05	4.17918e-05	4.16668e-05			
0.1	0.000333333	0.000333335	0.000333335	0.000333335	0.000332835	0.000333835	0.000333335			
0.15	0.001125	0.00112503	0.00112503	0.00112503	0.0011239	0.00112615	0.00112503			
0.2	0.002666667	0.002666687	0.002666687	0.002666687	0.00266487	0.00266887	0.002666687			
0.25	0.00520833	0.0052093	0.0052093	0.0052093	0.00520618	0.00521243	0.0052093			
0.3	0.009	0.00900347	0.00900347	0.00900347	0.00899897	0.00900798	0.00900347			
0.35	0.0142917	0.0143019	0.0143019	0.0143019	0.0142957	0.014308	0.0143019			
0.4	0.0213333	0.0213593	0.0213594	0.0213594	0.0213513	0.0213674	0.0213594			
0.45	0.030375	0.0304343	0.0304345	0.0304345	0.0304242	0.0304447	0.0304345			
0.5	0.0416667	0.0417907	0.0417911	0.0417911	0.0417785	0.0418038	0.0417911			
0.55	0.0554583	0.0557	0.0557013	0.0557013	0.0556859	0.0557168	0.0557013			
0.6	0.072	0.0724443	0.0724478	0.0724479	0.0724293	0.0724664	0.0724479			
0.65	0.0915417	0.0923198	0.0923282	0.0923283	0.0923063	0.0923503	0.0923283			
0.7	0.114333	0.115641	0.11566	0.11566	0.115634	0.115686	0.11566			
0.75	0.140625	0.142744	0.142785	0.142785	0.142755	0.142815	0.142785			
Продолжение на следующей странице										

Таблица 3.1 – продолжение

X	Метод решения							
	Пикарда (приближение)				Эйлера		Рунге-Кутта	
	1	2	3	4	Явный	Неявный		
0.8	0.170667	0.173995	0.174079	0.17408	0.174045	0.174115	0.17408	
0.85	0.204708	0.209797	0.209959	0.209963	0.209923	0.210004	0.209963	
0.9	0.243	0.250592	0.250897	0.250906	0.25086	0.250954	0.250907	
0.95	0.285792	0.296876	0.297431	0.297452	0.297398	0.297507	0.297453	
1	0.333333	0.349206	0.350185	0.35023	0.350169	0.350295	0.350232	
1.05	0.385875	0.40821	0.40989	0.409985	0.409917	0.410062	0.409989	
1.1	0.443667	0.474599	0.477414	0.477606	0.477533	0.477702	0.477617	
1.15	0.506958	0.549181	0.553793	0.554174	0.554101	0.554299	0.5542	
1.2	0.576	0.632876	0.640282	0.641016	0.64096	0.641193	0.641077	
1.25	0.651042	0.72673	0.738407	0.739786	0.739786	0.740062	0.739924	
1.3	0.732333	0.831934	0.850035	0.852572	0.852715	0.853045	0.85288	
1.35	0.820125	0.949842	0.977468	0.982048	0.982517	0.982919	0.982718	
1.4	0.914667	1.08199	1.12356	1.13168	1.13287	1.13336	1.13311	
1.45	1.01621	1.23012	1.29185	1.30602	1.30874	1.30935	1.30904	
1.5	1.125	1.39621	1.48677	1.51115	1.51705	1.51784	1.51745	
1.55	1.24129	1.58247	1.71385	1.75523	1.76777	1.7688	1.76829	
1.6	1.36533	1.79142	1.98002	2.04946	2.07572	2.07713	2.07642	
Продолжение на следующей странице								

Таблица 3.1 – продолжение

X	Метод решения						
	Пикарда (приближение)				Эйлера		Рунге-Кутта
	1	2	3	4	Явный	Неявный	
1.65	1.49737	2.02588	2.29401	2.40932	2.46411	2.46609	2.4651
1.7	1.63767	2.289	2.66677	2.85646	2.97133	2.97426	2.9728
1.75	1.78646	2.58432	3.1121	3.42159	3.66602	3.67066	3.66834
1.8	1.944	2.91578	3.64736	4.14864	4.6841	4.69217	4.68813
1.85	2.11054	3.28777	4.29442	5.10121	6.33839	6.3547	6.34652
1.9	2.28633	3.70518	5.08081	6.37221	9.54567	9.58846	9.56699
1.95	2.47162	4.1734	6.04115	8.0986	18.6449	18.8513	18.7472
2	2.66667	4.69841	7.21899	10.4839	270.068	391.93	317.566
Конец таблицы							

3.3 Вывод

В данном разделе были представлены результаты работы разработанного программного обеспечения.

4 Контрольные вопросы

4.1 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 на данный вопрос даётся на основании значений из таблицы, приведенной выше (Шаг равен 10^{-4}).

Учитывая тот факт, что i -ое приближение имеет порядок точности $= O(h^i)$, можно сделать вывод, что метод Пикара i -ого порядка точности будет корректен до тех пор, пока его значение совпадает (с определенной степенью точности, в нашем случае - 2 цифры после запятой) с методом Пикара $(i + 1)$ -ого порядка точности. Как только это значение начинает отклоняться, учитывая тот факт, что более высокое приближение имеет меньшую погрешность, метод нельзя считать корректным решением.

Таким образом:

1. 1 приближение считается решением на полуинтервале $[0; 0.85]$;
2. 2 приближение считается решением на полуинтервале $[0; 1.10]$;
3. 3 приближение считается решением на полуинтервале $[0; 1.30]$;
4. 4 приближение считается решением на отрезке $[0; 1.30]$, при этом дать более точную оценку для этого интервала в отсутствие 5-го приближения невозможно;

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

Для доказательства корректности полученного результата следует рассматривать малый интервал в окрестности некоторой точки. Если при устремлении шага вычисления к нулю полученные в результате работы алгоритма

значения совпадают (с заданной в задаче точностью), то расчет значений был произведен корректно.

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

Примерно 317.6

Заключение

В ходе лабораторной были приобретены навыки решения задачи Коши при помощи методов Пикара, Эйлера и Рунге–Кутта 2-го порядка. В процессе выполнения данной работы были выполнены следующие задачи:

- описана постановку задачи Коши;
- приведен конкретный пример задачи Коши, рассматриваемый в рамках лабораторной работы;
- описаны соответствующие методы решения задачи:
 - метод Пикара;
 - явный метод Эйлера;
 - неявный метод Эйлера;
 - метод Рунге–Кутта 2-го порядка;
- описана структуру разрабатываемого ПО;
- определены средства программной реализации;
- реализовано соответствующее ПО;
- приведены результаты работы разработанного ПО.