**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

**Дисциплина** Моделирование

**Тема** Программная реализация приближенного аналитического метода и простейших численных алгоритмов первого порядка точности при решении задачи Коши для ОДУ

**Студент** Мокеев Д. А.
**Группа** ИУ7-66Б
**Оценка (баллы)**
**Преподаватель** Градов В. М.

Москва, 2021 г.

# Оглавление

# Введение

Задачи данной лабораторной работы:

- Изучить метод Пикара;
- реализовать метод Пикарда, явный и неявный методы Эйлера, метод Рунге-Кутты второго порядка точности для решения уровнения $y'(x) = x^2 + y^2$;
- сравнить методы между собой.

# 1 Аналитическая часть

В данной части будут рассмотрены теоретические основы методов Пикара, явного и неявного методов Эйлера.

## 1.1 Постановка задачи

Пусть поставлена задача Коши:

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

## 1.2 Метод Пикара

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

Проинтегрируем выписанное уравнение

$$u(x) = u_0 + \int_{x_0}^{x} f(t, u(t))dt \tag{1.1}$$

Процедура последовательных приближений метода Пикара реализуется согласно следующей схеме.

$$y_s(x) = u_0 + \int_{x_0}^{x} f(t, y_{s-1}(t))dt \tag{1.2}$$

причем $y_0(t) = u_0$

## 1.3 Явный метод Эйлера

Самый простой метод решения уровнения - дискретизация расчетного интервала и замена производной в левой части $\frac{du(x)}{dt}$ разностным аналогом. Для некоторой i-ой точки сетки разностная производная определяется следующим образом:

$$\frac{du(t_i)}{dt} \approx \frac{y_{i+1} - y_i}{h} \tag{1.3}$$

Для того, чтобы схема имела простое решение, правую часть уравнения возьмем в той же точке $t_i$:

$$\frac{y_{i+1} - y_i}{h} = f(t_i, y_i) \tag{1.4}$$

Таким образом, мы сразу получаем рекуррентную формулу определения нового значения y в точке $t_{i+1}$, т.е. $y_{i+1}$ по значению y в точке $t_{i+1}$. Это значение обозначим как $y_i$, а $y_{i+1}$ запишем как:

$$y_{i+1} = y_i + h \cdot f(x_i, y_i) \tag{1.5}$$

## 1.4 Неявный метод Эйлера

$$y_{i+1} = y_i + h \cdot f(x_i + 1, y_i + 1) \tag{1.6}$$

Геометрическая интерпретация одного шага этого метода заключается в том, что решение на отрезке $[t_i; t_{i+1}]$ аппроксимируется касательной $y = y_{i+1} + y'(t_{i+1})(t - t_{i+1})$, проведенной в точке $(t_{i+1}, y_{i+1})$ к интегральной кривой, проходящей через эту точку.

# 2 Технологическая часть

## 2.1 Выбор ЯП

В качестве языка программирования был выбран golang. Время работы алгоритмов было замерено с помощью time.

## 2.2 Листинг кода алгоритмов

В данном разделе будут приведены листинги кода решения методом Пикара (Листинг 2.1) и Эйлера (Листинг 2.3)

Листинг 2.1: Метод Пикара

```
func picard(x float64, n int)float64{
        u0 := 0.0
        answer := 0.0
        poly := make(map[int]float64)
        curr := make(map[int]float64)
        poly[2] = 1.0
        var res float64
        for i:=0;i<n;i++{
                curr = poly_pow(curr)
                curr[2] = 1.0
                curr, res = integrate(curr, 0.0, x)
                answer = u0 + res
        }
        return answer
}

//add term to polinomial
func add(poly map[int]float64, term *term)map[int]float64{
        if _, ok := poly[term.pow]; ok{
                poly[term.pow] += term.coef
        }else{
                poly[term.pow] = term.coef
        }
        return poly
}

//multiply two terms and write to polinomial
func mult(poly map[int]float64, term1, term2 *term)map[int]float64{
        to_add := multterm(term1, term2)
        add(poly, to_add)
        return poly
}

//multiply two terms and return result
func multterm(term1, term2 *term)*term{
        res := term{term1.coef*term2.coef, term2.pow+term1.pow}
        return &res
}
```

```go
//squaring a polinomial
func poly_pow(poly map[int]float64) map[int]float64{
	res := make(map[int]float64)
	for i, j := range poly{
		for k, z := range poly{
			mult(res, &term{j, i}, &term{z,k})
		}
	}
	return res
}

func integrate(poly map[int]float64, x0, x float64) (map[int]float64, float64){
	var answer float64
	res := make(map[int]float64)
	for i, j := range poly{
		integr := term{j, i+1}
		integr.coef *= 1.0/float64(integr.pow)
		res[integr.pow] = integr.coef
		answer += integr.coef*math.Pow(x, float64(integr.pow)) -
			integr.coef*math.Pow(x0, float64(integr.pow))
	}
	return res, answer
}
```

Листинг 2.2: Явный и неявный методы Эйлера

```
func euler_explicit(xn float64, n int)float64{
	h := xn / float64(n)
	y:=0.0
	x:=0.0
	for i:=0; i<=n;i++{
		y = y + h*f(x, y)
		x+=h
	}
	return y
}
func euler_implicit(xn float64, n int)float64{
	h := xn / float64(n)
	y:=0.0
	x:=0.0
	var a, b, c, dis, x1 float64
	for i:=0;i<=n;i++{
		a = 1; b = -1.0/h; c = 1.0/h*y+(x+h)*(x+h)
		dis = D(a, b, c)
		if dis>=0{
			x1 = (-b - math.Sqrt(dis))/2/a
		}
		y = x1
		x+=h
	}
	return y
}
```

Листинг 2.3: Метод Рунге-Кутты

```
func runge_kutta(xn float64, n int)float64{
	h := xn / float64(n)
	alpha := 0.5
	y := 0.0
	x := 0.0

	for i:=0; i<n;i++{
		y += h * ( (1- alpha) * f(x,y) + alpha * f(x+h/(2*alpha), y+(h/2*alpha)*f(x,y)) )
		x+=h
	}
	_ = x
	return y
}
```

# 3 Примеры работы программы

```
C:\Users\daniil.mokeev\Desktop\Modeling\lab01>go run main.go
          x|                             Picard's|   Explicit|   Implicit|Runge-Kutta|
------------------------------------------------------------------------
           |7 -e        |8 -e        |9 -e        |           |           |           |
    0.00000|     0.00000|     0.00000|     0.00000|    0.00000|    0.00000|    0.00000|
    0.10000|     0.00033|     0.00033|     0.00033|    0.00033|    0.00017|    0.00033|
    0.20000|     0.00267|     0.00267|     0.00267|    0.00267|    0.00265|    0.00267|
    0.30000|     0.00900|     0.00900|     0.00900|    0.00900|    0.00897|    0.00900|
    0.40000|     0.02136|     0.02136|     0.02136|    0.02136|    0.02136|    0.02136|
    0.50000|     0.04179|     0.04179|     0.04179|    0.04179|    0.04179|    0.04179|
    0.60000|     0.07245|     0.07245|     0.07245|    0.07245|    0.07248|    0.07245|
    0.70000|     0.11566|     0.11566|     0.11566|    0.11566|    0.11564|    0.11566|
    0.80000|     0.17408|     0.17408|     0.17408|    0.17408|    0.17408|    0.17408|
    0.90000|     0.25091|     0.25091|     0.25091|    0.25091|    0.25092|    0.25091|
    1.00000|     0.35023|     0.35023|     0.35023|    0.35023|    0.35023|    0.35023|
    1.10000|     0.47762|     0.47762|     0.47762|    0.47762|    0.47761|    0.47762|
    1.20000|     0.64108|     0.64108|     0.64108|    0.64108|    0.64110|    0.64108|
    1.30000|     0.85288|     0.85288|     0.85288|    0.85288|    0.85289|    0.85288|
    1.40000|     1.13311|     1.13311|     1.13311|    1.13311|    1.13311|    1.13311|
    1.50000|     1.51743|     1.51745|     1.51745|    1.51744|    1.51747|    1.51745|
    1.60000|     2.07621|     2.07639|     2.07642|    2.07641|    2.07641|    2.07642|
    1.70000|     2.97033|     2.97228|     2.97270|    2.97277|    2.97287|    2.97279|
    1.80000|     4.65557|     4.67832|     4.68550|    4.68806|    4.68822|    4.68809|
    1.90000|     8.92014|     9.26263|     9.43899|    9.56659|    9.56718|    9.56676|
    2.00000|    27.39756|    36.48255|    47.91370|  316.57073|  318.93388|  316.93302|
    2.10000|   222.40896|1026.47753|           NaN|       +Inf|192708.68697|       +Inf|
------------------------------------------------------------------------
```

Рис. 3.1: Приблежения 7,8,9

| x | | | Picard's | Explicit | Implicit | Runge-Kutta |
|---|---|---|---|---|---|---|
| | 2 -e | 3 -e | 4 -e | | | |
| 0.00000 | 0.00000 | 0.00000 | 0.00000 | 0.00000 | 0.00000 | 0.00000 |
| 0.05000 | 0.00004 | 0.00004 | 0.00004 | 0.00004 | 0.00000 | 0.00004 |
| 0.10000 | 0.00033 | 0.00033 | 0.00033 | 0.00033 | 0.00000 | 0.00033 |
| 0.15000 | 0.00113 | 0.00113 | 0.00113 | 0.00113 | 0.00000 | 0.00113 |
| 0.20000 | 0.00267 | 0.00267 | 0.00267 | 0.00267 | 0.00000 | 0.00267 |
| 0.25000 | 0.00521 | 0.00521 | 0.00521 | 0.00521 | 0.00000 | 0.00521 |
| 0.30000 | 0.00900 | 0.00900 | 0.00900 | 0.00900 | 0.00662 | 0.00900 |
| 0.35000 | 0.01430 | 0.01430 | 0.01430 | 0.01430 | 0.00888 | 0.01430 |
| 0.40000 | 0.02136 | 0.02136 | 0.02136 | 0.02136 | 0.02421 | 0.02136 |
| 0.45000 | 0.03043 | 0.03043 | 0.03043 | 0.03043 | 0.03143 | 0.03043 |
| 0.50000 | 0.04179 | 0.04179 | 0.04179 | 0.04179 | 0.04041 | 0.04179 |
| 0.55000 | 0.05570 | 0.05570 | 0.05570 | 0.05570 | 0.05770 | 0.05570 |
| 0.60000 | 0.07244 | 0.07245 | 0.07245 | 0.07245 | 0.07285 | 0.07245 |
| 0.65000 | 0.09232 | 0.09233 | 0.09233 | 0.09233 | 0.08985 | 0.09233 |
| 0.70000 | 0.11564 | 0.11566 | 0.11566 | 0.11566 | 0.11498 | 0.11566 |
| 0.75000 | 0.14274 | 0.14278 | 0.14279 | 0.14279 | 0.14148 | 0.14279 |
| 0.80000 | 0.17400 | 0.17408 | 0.17408 | 0.17408 | 0.17077 | 0.17408 |
| 0.85000 | 0.20980 | 0.20996 | 0.20996 | 0.20996 | 0.20937 | 0.20996 |
| 0.90000 | 0.25059 | 0.25090 | 0.25091 | 0.25091 | 0.25236 | 0.25091 |
| 0.95000 | 0.29688 | 0.29743 | 0.29745 | 0.29745 | 0.29664 | 0.29745 |
| 1.00000 | 0.34921 | 0.35019 | 0.35023 | 0.35023 | 0.34839 | 0.35023 |
| 1.05000 | 0.40821 | 0.40989 | 0.40998 | 0.40999 | 0.41108 | 0.40999 |
| 1.10000 | 0.47460 | 0.47741 | 0.47761 | 0.47762 | 0.47788 | 0.47762 |
| 1.15000 | 0.54918 | 0.55379 | 0.55417 | 0.55420 | 0.55553 | 0.55420 |
| 1.20000 | 0.63288 | 0.64028 | 0.64102 | 0.64108 | 0.64181 | 0.64108 |
| 1.25000 | 0.72673 | 0.73841 | 0.73979 | 0.73992 | 0.74079 | 0.73992 |
| 1.30000 | 0.83193 | 0.85003 | 0.85257 | 0.85288 | 0.85299 | 0.85288 |
| 1.35000 | 0.94984 | 0.97747 | 0.98205 | 0.98272 | 0.98221 | 0.98272 |
| 1.40000 | 1.08199 | 1.12356 | 1.13168 | 1.13311 | 1.13221 | 1.13311 |
| 1.45000 | 1.23012 | 1.29185 | 1.30602 | 1.30904 | 1.30813 | 1.30904 |
| 1.50000 | 1.39621 | 1.48677 | 1.51115 | 1.51745 | 1.51774 | 1.51745 |
| 1.55000 | 1.58247 | 1.71385 | 1.75523 | 1.76828 | 1.76842 | 1.76828 |
| 1.60000 | 1.79142 | 1.98002 | 2.04946 | 2.07642 | 2.07182 | 2.07642 |
| 1.65000 | 2.02588 | 2.29401 | 2.40932 | 2.46509 | 2.46992 | 2.46510 |
| 1.70000 | 2.28900 | 2.66677 | 2.85646 | 2.97279 | 2.97489 | 2.97280 |
| 1.75000 | 2.58432 | 3.11210 | 3.42159 | 3.66833 | 3.66407 | 3.66833 |
| 1.80000 | 2.91578 | 3.64736 | 4.14864 | 4.68812 | 4.69038 | 4.68813 |
| 1.85000 | 3.28777 | 4.29442 | 5.10121 | 6.34651 | 6.33802 | 6.34652 |
| 1.90000 | 3.70518 | 5.08081 | 6.37221 | 9.56695 | 9.55489 | 9.56697 |
| 1.95000 | 4.17340 | 6.04115 | 8.09860 | 18.74707 | 18.80764 | 18.74715 |
| 2.00000 | 4.69841 | 7.21899 | 10.48392 | 317.60683 | 306.83411 | 317.64330 |
| 2.05000 | 5.28682 | 8.66900 | 13.83300 | +Inf | 1816516.16562 | +Inf |
| 2.10000 | 5.94587 | 10.45978 | 18.60527 | +Inf | 1685080.24503 | +Inf |
| 2.15000 | 6.68356 | 12.67726 | 25.49726 | +Inf | 1325922.83456 | +Inf |
| 2.20000 | 7.50863 | 15.42893 | 35.57014 | +Inf | 1288620.45601 | +Inf |

C:\Users\daniil.mokeev\Desktop\Modeling\lab01>

Рис. 3.2: Приблежения 2,3,4

# 4 Ответы на контрольные вопросы

## 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 новое приблежение метода Пикара увеличивает точность решения. Для оценки точности n-ого приблежения можно сравнить его со следующим.

$$|y_{n+1} - y_n| < \epsilon \tag{4.1}$$

где $\epsilon$ - заданная погрешность

Для точности $\epsilon = 0.01$:

- 1-ое приблежение $x \in [0; 0.69]$
- 2-ое приблежение $x \in [0; 1.11]$
- 3-е приблежение $x \in [0; 1.35]$
- 4-е приблежение $x \in [0; 1.51]$

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

Для подтверждения правильности полученного результата можно уменьшать значение шага при вычислении метода. Так как точность численных методов зависит от значения шага, если при уменьшении этого значения результат будет меняться на значение меньшее, чем $\epsilon$, это значение можно считать верным.

Так как из-за ограний ЭВМ нельзя уменьшать шаг до предельно малых значений, реальное решение можно получить только с заданной погрешностью.

## Из каких соображений выбирался корень уравнения в неявном методе?

Выбирался меньший корень для минимизации ошибки.

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

Значения приведены в примерах работы программы.