

Оглавление

Введение	3
1 Аналитическая часть	4
1.1 Постановка задачи	4
1.2 Метод Пикара	4
1.3 Явный метод Эйлера	4
1.4 Неявный метод Эйлера	5
2 Технологическая часть	6
2.1 Выбор ЯП	6
2.2 Листинг кода алгоритмов	6
3 Примеры работы программы	9
4 Ответы на контрольные вопросы	11

Введение

Задачи данной лабораторной работы:

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

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

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

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

Процедура последовательных приближений метода Пикара реализуется согласно следующей схеме.

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

причем $y_0(t) = u_0$

1.3 Явный метод Эйлера

Самый простой метод решения уравнения - 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:

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

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

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

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:

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

1.4 Неявный метод Эйлера

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

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 go lang. 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
}
```

```

//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.mokeyev\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: Приближения 7,8,9

x	Picard's	Explicit	Implicit	Runge-Kutta
0.00000	0.00000	0.00000	0.00000	0.00000
0.05000	0.00004	0.00004	0.00000	0.00004
0.10000	0.00033	0.00033	0.00000	0.00033
0.15000	0.00113	0.00113	0.00000	0.00113
0.20000	0.00267	0.00267	0.00000	0.00267
0.25000	0.00521	0.00521	0.00000	0.00521
0.30000	0.00900	0.00900	0.00662	0.00900
0.35000	0.01430	0.01430	0.00888	0.01430
0.40000	0.02136	0.02136	0.02421	0.02136
0.45000	0.03043	0.03043	0.03143	0.03043
0.50000	0.04179	0.04179	0.04041	0.04179
0.55000	0.05570	0.05570	0.05770	0.05570
0.60000	0.07244	0.07245	0.07285	0.07245
0.65000	0.09232	0.09233	0.08985	0.09233
0.70000	0.11564	0.11566	0.11498	0.11566
0.75000	0.14274	0.14278	0.14148	0.14279
0.80000	0.17400	0.17408	0.17077	0.17408
0.85000	0.20980	0.20996	0.20937	0.20996
0.90000	0.25059	0.25091	0.25236	0.25091
0.95000	0.29688	0.29743	0.29664	0.29745
1.00000	0.34921	0.35019	0.34839	0.35023
1.05000	0.40821	0.40989	0.41108	0.40999
1.10000	0.47460	0.47741	0.47788	0.47762
1.15000	0.54918	0.55379	0.55553	0.55420
1.20000	0.63288	0.64028	0.64181	0.64108
1.25000	0.72673	0.73841	0.73992	0.73992
1.30000	0.83193	0.85003	0.85288	0.85288
1.35000	0.94984	0.97747	0.98221	0.98272
1.40000	1.08199	1.12356	1.13311	1.13311
1.45000	1.23012	1.29185	1.30904	1.30904
1.50000	1.39621	1.48677	1.51745	1.51745
1.55000	1.58247	1.71385	1.76828	1.76828
1.60000	1.79142	1.98002	2.07642	2.07642
1.65000	2.02588	2.29401	2.46509	2.46510
1.70000	2.28900	2.66677	2.97279	2.97280
1.75000	2.58432	3.11210	3.66833	3.66833
1.80000	2.91578	3.64736	4.68812	4.68813
1.85000	3.28777	4.29442	5.10121	5.10121
1.90000	3.70518	5.08081	6.37221	6.37221
1.95000	4.17340	6.04115	8.09860	8.09860
2.00000	4.69841	7.21899	10.48392	10.48392
2.05000	5.28682	8.66900	13.83300	13.83300
2.10000	5.94587	10.45978	18.60527	18.60527
2.15000	6.68356	12.67726	25.49726	25.49726
2.20000	7.50863	15.42893	35.57014	35.57014
2.25000				
2.30000				
2.35000				
2.40000				
2.45000				
2.50000				
2.55000				
2.60000				
2.65000				
2.70000				
2.75000				
2.80000				
2.85000				
2.90000				
2.95000				
3.00000				
3.05000				
3.10000				
3.15000				
3.20000				
3.25000				
3.30000				
3.35000				
3.40000				
3.45000				
3.50000				
3.55000				
3.60000				
3.65000				
3.70000				
3.75000				
3.80000				
3.85000				
3.90000				
3.95000				
4.00000				
4.05000				
4.10000				
4.15000				
4.20000				
4.25000				
4.30000				
4.35000				
4.40000				
4.45000				
4.50000				
4.55000				
4.60000				
4.65000				
4.70000				
4.75000				
4.80000				
4.85000				
4.90000				
4.95000				
5.00000				
5.05000				
5.10000				
5.15000				
5.20000				
5.25000				
5.30000				
5.35000				
5.40000				
5.45000				
5.50000				
5.55000				
5.60000				
5.65000				
5.70000				
5.75000				
5.80000				
5.85000				
5.90000				
5.95000				
6.00000				
6.05000				
6.10000				
6.15000				
6.20000				
6.25000				
6.30000				
6.35000				
6.40000				
6.45000				
6.50000				
6.55000				
6.60000				
6.65000				
6.70000				
6.75000				
6.80000				
6.85000				
6.90000				
6.95000				
7.00000				
7.05000				
7.10000				
7.15000				
7.20000				
7.25000				
7.30000				
7.35000				
7.40000				
7.45000				
7.50000				
7.55000				
7.60000				
7.65000				
7.70000				
7.75000				
7.80000				
7.85000				
7.90000				
7.95000				
8.00000				
8.05000				
8.10000				
8.15000				
8.20000				
8.25000				
8.30000				
8.35000				
8.40000				
8.45000				
8.50000				
8.55000				
8.60000				
8.65000				
8.70000				
8.75000				
8.80000				
8.85000				
8.90000				
8.95000				
9.00000				
9.05000				
9.10000				
9.15000				
9.20000				
9.25000				
9.30000				
9.35000				
9.40000				
9.45000				
9.50000				
9.55000				
9.60000				
9.65000				
9.70000				
9.75000				
9.80000				
9.85000				
9.90000				
9.95000				
10.00000				

Рис. 3.2: Приближения 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 приближение метода Пикара увеличивает точность решения. Для оценки точности n -ого приближения можно сравнить его со следующим.

$$|y_{n+1} - y_n| < \epsilon \quad (4.1)$$

где ϵ - заданная погрешность

Для точности $\epsilon = 0.01$:

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

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 ϵ , это значение можно считать верным.

Так как из-за ограничений 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.