**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»

# Отчёт по лабораторной работе №1 по курсу «Моделирование»

**Тема** Решение задачи Коши методами Эйлера, Пиккара и Рунге-Кутта

**Студент** Прянишников А.Н.

**Группа** ИУ7-65Б

**Оценка (баллы)** _____

**Преподаватели** Градов В. М

Москва — 2022 г.

# Цель работы

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

# Аналитическая часть

## Постановка задачи

Имеется общее дифференциальное уравнение, у которого отсутствует аналитическое решение:

$$\begin{cases} u'(x) = x^2 + u^2 \\ u(0) = 0 \end{cases} \tag{1}$$

Для решения данного ОДУ были использованы 3 алгоритма.

## Метод Пикара

Имеем:

$$u(x) = \eta + \int_{\xi}^{x} f(t, u(t))\, dt \tag{2}$$

Строим ряд функций:

$$y^{(s)} = \eta + \int_{\xi}^{x} f(t, y^{(s-1)}(t))\, dt, \qquad y^{(0)} = \eta \tag{3}$$

Построим 4 приближения для уравнения (2):

$$y^{(1)}(x) = 0 + \int_{0}^{x} t^2\, dt = \frac{x^3}{3} \tag{4}$$

$$y^{(2)}(x) = 0 + \int_{0}^{x} (t^2 + \left(\frac{t^3}{3}\right)^2)\, dt = \frac{x^3}{3} + \frac{x^7}{63} \tag{5}$$

$$y^{(3)}(x) = 0 + \int_{0}^{x} (t^2 + \left(\frac{t^3}{3} + \frac{t^7}{63}\right)^2)\, dt = \frac{x^3}{3} + \frac{x^7}{63} + \frac{2x^{11}}{2079} + \frac{x^{15}}{59535} \tag{6}$$

$$y^{(4)}(x) = 0 + \int_0^x (t^2 + \left(\frac{t^3}{3} + \frac{t^7}{63} + \frac{2t^{11}}{2079} + \frac{t^{15}}{59535}\right)^2)\,dt = \frac{x^3}{3} + \frac{x^7}{63} + \frac{2x^{11}}{2079} + \frac{x^{15}}{59535} + \frac{2x^{15}}{93555} + \frac{2x^{19}}{3393495} + \frac{2x^{19}}{2488563} + \frac{2x^{23}}{86266215} + \frac{x^{23}}{99411543} + \frac{2x^{27}}{3341878155} + \frac{x^{31}}{109876902975} \tag{7}$$

# Метод Рунге-Кутта

$$y^{n+1}(x) = y^n(x) + h((1-\alpha)R_1 + \alpha R_2) \tag{8}$$

где $R1 = f(x_n, y^n)$, $R2 = f(x_n + \frac{h}{2\alpha}, y^n + \frac{h}{2\alpha}R_1)$, $\alpha = \frac{1}{2}$ или 1

Порядок точности: $O(h^2)$.

## Метод Эйлера

$$y^{(n+1)}(x) = y^{(n)}(x) + h \cdot f(x_n, y^{(n)}) \tag{9}$$

Порядок точности: $O(h)$.

# Код программы

На листинге 1 представлена реализация алгоритмов Пиккара, Рунге-Кутта и Эйлера.

Листинг 1: Реализация алгоритмов

```
import matplotlib.pyplot as plt
import numpy as np

EPS = 10**(-2)

def f(x, u):
    return x**2 + u**2

def f_1(x):
    return x**3 / 3

def f_2(x, f1):
    return f1 + x**7 / 63

def f_3(x, f2):
    return f2 + 2*x**11 / 2079 + x**15 / 59535

def f_4(x, f3):
    return f3 + 2*x**15 / 93555 + 2*x**19 / 3393495 + \
        + 2*x**19 / 2488563 + 2*x**23 / 86266215 + 2*x**23 / 99411543 + \
            + 2*x**27 / 3341878155 + x**31 / 109876902975

def pikkar_method(x_min, x_max, h):
    x = x_min
    result_1 = []
    result_2 = []
    result_3 = []
    result_4 = []
    while x <= x_max:
        f1 = f_1(x)
        f2 = f_2(x, f1)
        f3 = f_3(x, f2)
        f4 = f_4(x, f3)
        result_1.append(f1)
        result_2.append(f2)
        result_3.append(f3)
        result_4.append(f4)
        x += h
    return result_1, result_2, result_3, result_4

# Alpha = 1
```

```
def runge_kutt_1(x_min, x_max, h, y0):
    x = x_min
    y = y0
    result_y = []
    result_y.append(y)
    while x <= x_max:
        try:
            y_temp = y + h/2 * f(x, y)
            y_pr_temp = f(x + h/2, y_temp)
            y += h * y_pr_temp
            x += h
            result_y.append(y)
        except:
            while x <= x_max:
                result_y.append('-')
                x += h
            return result_y

    return result_y

# Alpha = 0.5
def runge_kutt_2(x_min, x_max, h, y0):
    x = x_min
    y = y0
    result_y = []
    result_y.append(y)
    while x <= x_max:
        try:
            y_temp = y + h/2 * f(x, y)
            y_pr_temp = f(x + h, y_temp)
            y_pr_temp_2 = 0.5 * (f(x, y) + y_pr_temp)
            y += h * y_pr_temp_2
            x += h
            result_y.append(y)
        except:
            while x <= x_max:
                result_y.append('-')
                x += h
            return result_y

    return result_y

def euler(x_min, x_max, h, y0):
    x = x_min
    y = y0
    result = []
    result.append(y)
    while x <= x_max:
```

```
        try:
            y += h * f(x, y)
            x += h
            result.append(y)
        except:
            while x <= x_max:
                result.append('-')
                x += h
            return result

    return result
```

# Демонстрация работы программы

По заданию требуется границу интервала $x_{max}$ выбирать максимально возможной из условия, чтобы численные методы обеспечивали точность вычисления решения уравнения до второго знака после запятой.

На листинге 2 представлена реализация поиска границы интервала $x_{max}$.

Листинг 2: Поиск границы интервала

```
def find_accuracy(x_min, x_max, h, y):
    res5 = runge_kutt_1(x_min, x_max, h, y)
    res6 = runge_kutt_2(x_min, x_max, h, y)
    res7 = euler(x_min, x_max, h, y)

    x = 0
    N = len(res5)
    for i in range(0, N):
        if (abs(res5[i] - res6[i]) <= EPS) and \
            (abs(res6[i] - res7[i]) <= EPS) and \
                (abs(res5[i] - res7[i]) <= EPS):
            x +=  h
        else:
            print()
            print("Xmax␣=␣", x)
            break
    return x
```

На рисунке 1 представлен результат поиска границы:

```
Xmax =  1.9493000000043028
```

Рисунок. 1: Поиск границы

На рисунке 2 представлена таблица решений уравнений различными методами до правой границы с интервалом 0.05:

| X | Pikkar_1 | Pikkar_2 | Pikkar_3 | Pikkar_4 | Runge(a=1) | Runge(a=0.5) | Euler |
|---|---|---|---|---|---|---|---|
| 0.00 | 0.00000000 | 0.00000000 | 0.00000000 | 0.00000000 | 0.00000000 | 0.00000000 | 0.00000000 |
| 0.05 | 0.00004167 | 0.00004167 | 0.00004167 | 0.00004167 | 0.00004167 | 0.00004167 | 0.00004165 |
| 0.10 | 0.00033333 | 0.00033333 | 0.00033333 | 0.00033333 | 0.00033333 | 0.00033333 | 0.00033328 |
| 0.15 | 0.00112500 | 0.00112503 | 0.00112503 | 0.00112503 | 0.00112503 | 0.00112503 | 0.00112491 |
| 0.20 | 0.00266667 | 0.00266687 | 0.00266687 | 0.00266687 | 0.00266687 | 0.00266687 | 0.00266667 |
| 0.25 | 0.00520833 | 0.00520930 | 0.00520930 | 0.00520930 | 0.00520930 | 0.00520930 | 0.00520899 |
| 0.30 | 0.00900000 | 0.00900347 | 0.00900347 | 0.00900347 | 0.00900347 | 0.00900347 | 0.00900302 |
| 0.35 | 0.01429167 | 0.01430188 | 0.01430189 | 0.01430189 | 0.01430189 | 0.01430189 | 0.01430127 |
| 0.40 | 0.02133333 | 0.02135934 | 0.02135938 | 0.02135938 | 0.02135938 | 0.02135938 | 0.02135858 |
| 0.45 | 0.03037500 | 0.03043431 | 0.03043446 | 0.03043446 | 0.03043446 | 0.03043446 | 0.03043344 |
| 0.50 | 0.04166667 | 0.04179067 | 0.04179114 | 0.04179115 | 0.04179115 | 0.04179114 | 0.04178988 |
| 0.55 | 0.05545833 | 0.05569999 | 0.05570133 | 0.05570133 | 0.05570134 | 0.05570133 | 0.05569979 |
| 0.60 | 0.07200000 | 0.07244434 | 0.07244784 | 0.07244785 | 0.07244786 | 0.07244785 | 0.07244601 |
| 0.65 | 0.09154167 | 0.09231980 | 0.09232824 | 0.09232828 | 0.09232831 | 0.09232829 | 0.09232611 |
| 0.70 | 0.11433333 | 0.11564054 | 0.11565965 | 0.11565975 | 0.11565985 | 0.11565982 | 0.11565727 |
| 0.75 | 0.14062500 | 0.14274379 | 0.14278465 | 0.14278494 | 0.14278523 | 0.14278518 | 0.14278222 |
| 0.80 | 0.17066667 | 0.17399548 | 0.17407871 | 0.17407948 | 0.17408026 | 0.17408019 | 0.17407676 |
| 0.85 | 0.20470833 | 0.20979686 | 0.20995931 | 0.20996124 | 0.20996322 | 0.20996311 | 0.20995916 |
| 0.90 | 0.24300000 | 0.25059201 | 0.25089736 | 0.25090195 | 0.25090668 | 0.25090652 | 0.25090199 |
| 0.95 | 0.28579167 | 0.29687639 | 0.29743135 | 0.29744180 | 0.29745263 | 0.29745240 | 0.29744721 |
| 1.00 | 0.33333333 | 0.34920635 | 0.35018515 | 0.35020796 | 0.35023184 | 0.35023152 | 0.35022557 |
| 1.05 | 0.38587500 | 0.40820993 | 0.40989019 | 0.40993829 | 0.40998919 | 0.40998873 | 0.40998192 |
| 1.10 | 0.44366667 | 0.47459868 | 0.47741355 | 0.47751177 | 0.47761702 | 0.47761639 | 0.47760857 |
| 1.15 | 0.50695833 | 0.54918087 | 0.55379315 | 0.55398804 | 0.55420008 | 0.55419922 | 0.55419020 |
| 1.20 | 0.57600000 | 0.63287589 | 0.64028242 | 0.64065925 | 0.64107673 | 0.64107554 | 0.64106510 |
| 1.25 | 0.65104167 | 0.72673010 | 0.73840666 | 0.73911851 | 0.73992422 | 0.73992260 | 0.73991042 |
| 1.30 | 0.73233333 | 0.83193415 | 0.85003452 | 0.85135123 | 0.85287999 | 0.85287777 | 0.85286345 |
| 1.35 | 0.82012500 | 0.94984168 | 0.97746847 | 0.97985817 | 0.98271805 | 0.98271499 | 0.98269798 |
| 1.40 | 0.91466667 | 1.08198969 | 1.12355960 | 1.12782283 | 1.13311267 | 1.13310843 | 1.13308797 |
| 1.45 | 1.01620833 | 1.23012049 | 1.29185290 | 1.29934157 | 1.30904442 | 1.30903848 | 1.30901347 |
| 1.50 | 1.12500000 | 1.39620536 | 1.48677133 | 1.49974327 | 1.51744754 | 1.51743913 | 1.51740796 |
| 1.55 | 1.24129167 | 1.58246992 | 1.71384867 | 1.73603804 | 1.76828516 | 1.76827302 | 1.76823324 |
| 1.60 | 1.36533333 | 1.79142136 | 1.98002380 | 2.01755403 | 2.07642338 | 2.07640542 | 2.07635311 |
| 1.65 | 1.49737500 | 2.02587752 | 2.29401209 | 2.35685053 | 2.46509615 | 2.46506871 | 2.46499726 |
| 1.70 | 1.63766667 | 2.28899789 | 2.66677353 | 2.77104085 | 2.97279722 | 2.97275336 | 2.97265088 |
| 1.75 | 1.78645833 | 2.58431668 | 3.11210160 | 3.28372784 | 3.66833693 | 3.66826238 | 3.66810523 |
| 1.80 | 1.94400000 | 2.91577783 | 3.64736281 | 3.92786202 | 4.68813070 | 4.68799187 | 4.68772698 |
| 1.85 | 2.11054167 | 3.28777229 | 4.29442366 | 4.74999897 | 6.34652546 | 6.34622695 | 6.34571003 |
| 1.90 | 2.28633333 | 3.70517736 | 5.08080977 | 5.81669111 | 9.56699529 | 9.56616341 | 9.56485626 |
| 1.95 | 2.47162500 | 4.17339840 | 6.04115248 | 7.22415275 | 18.74727220 | 18.74300215 | 18.73696032 |

Рисунок. 2: Поиск границы

На рисунке 3 представлен график функции в диапазоне $[-x_{max}; x_{max}]$:

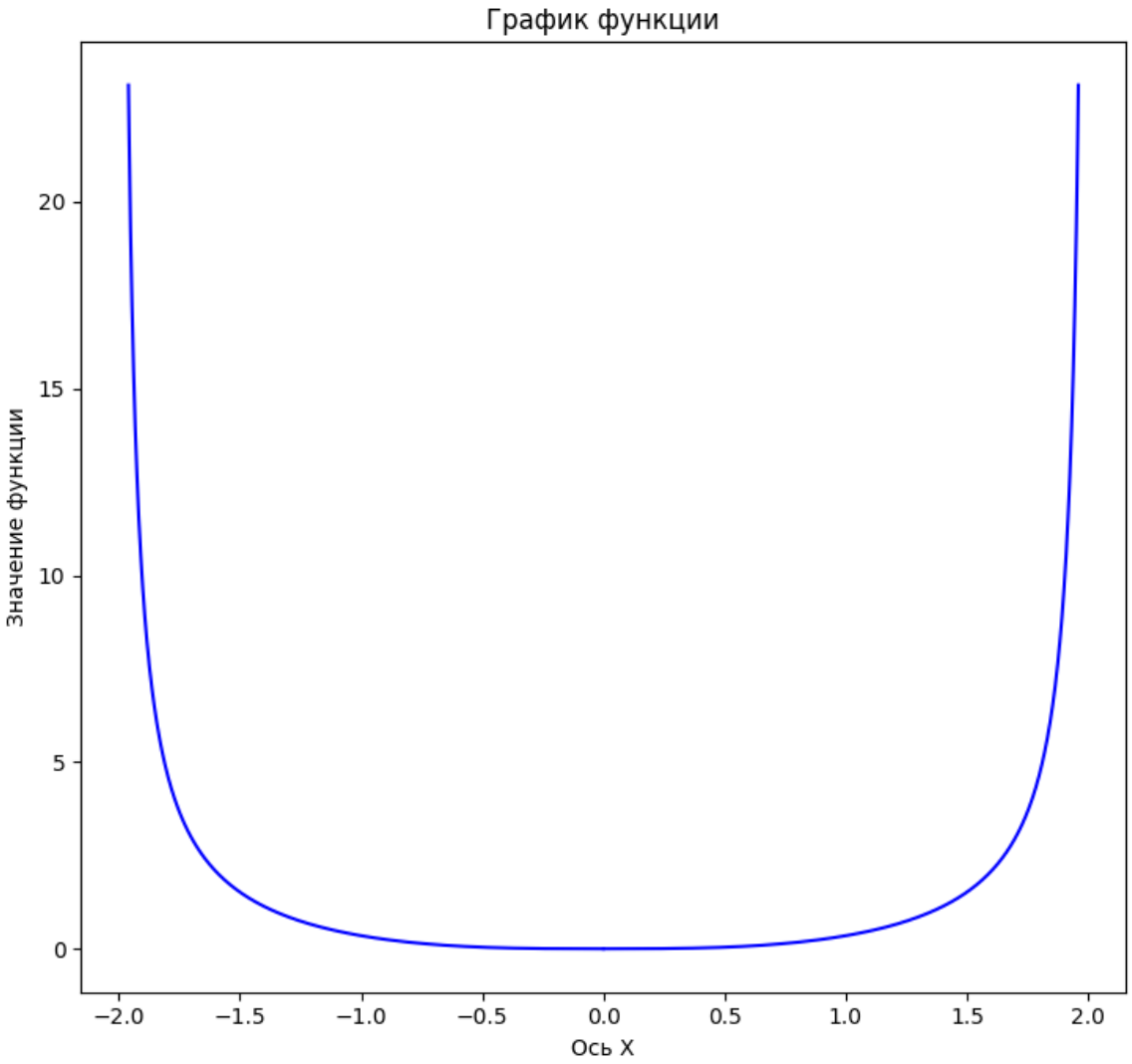


Рисунок. 3: График функции