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: «Приближенный аналитический метод Пикара в сравнении с численными методами»

Студент: Овчинникова А. П.

Группа: ИУ7-65Б

Оценка (баллы): _____

Преподаватель: Градов В. М.

Москва, 2020

Целью данной лабораторной работы является анализ и сравнение численных методов и приближенного аналитического метода Пикара.

Рассмотрим задачу для уравнения первого порядка:

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

Интегрируя дифференциальное уравнение, заменим эту задачу эквивалентным ей интегральным уравнением:

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

Решая это интегральное уравнение методом последовательных приближений, получим итерационный процесс Пикара:

$$y_s(x) = \eta + \int_{\varepsilon}^{x} f(t,y_{s-1}(t))dt\,, y_0(x) \equiv \eta$$

Рассмотрим пример:

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

Это уравнение не имеет аналитического решения. Его можно решить методом Пикара:

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

$$y^{(2)}(x) = 0 + \int_{0}^{x} [t^2 + \left(\frac{t^3}{3}\right)^2]dt = \frac{t^3}{3} + \frac{t^7}{63}|_0^x = \frac{x^3}{3} + \frac{x^7}{63}$$

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

$$y^{(4)}(x) = 0 + \int_0^x [t^2 + \left(\frac{x^3}{3} + \frac{x^7}{63} + \frac{2x^{11}}{2079} + \frac{x^{15}}{59535}\right)^2]dt =$$

$$= \frac{x^3}{3} + \frac{x^7}{63} + \frac{2x^{11}}{2079} + \frac{x^{15}}{59535} + \frac{4x^{15}}{93555} + \frac{2x^{19}}{3393495} + \frac{4x^{19}}{2488563}$$
$$+ \frac{2x^{23}}{86266215} + \frac{x^{23}}{99411543} + \frac{4x^{27}}{3341878155} + \frac{x^{31}}{109876902975}$$

Эту задачу можно также решить, используя численные методы.

Все численные методы решения ОДУ основаны на аппроксимации дифференциальных уравнений разностными аналогами. В зависимости от конкретной формы аппроксимации, получаются алгоритмы различной точности и быстродействия. Рассмотрим самый простой из алгоритмов, который в настоящее время практически не используется, ввиду малой точности.

Метод Эйлера — простейший численный метод решения систем обыкновенных дифференциальных уравнений. Метод Эйлера является явным, одношаговым методом первого порядка точности. Рекуррентная формула для определения нового значения *y* в точке *n+1* (т. е. y_{n+1}) по значению y в точке *n* (т. е. y_n) запишем как:

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

где h – шаг.

Существует также неявный метод Эйлера, который является небольшим усложнением явного метода:

$$y_{n+1} = y_n + h(f(x_{n+1}, y_{n+1}))$$

Неявный метод Эйлера устойчив и имеет первый порядок аппроксимации. Рассмотрим неявную схему на примере:

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

$$y_{n+1} = y_n + h(x_{n+1}^2 + y_{n+1}^2)$$

$$hy_{n+1}^2 - y_{n+1} + (y_n + hx_{n+1}^2) = 0$$

$$D = b^2 - 4ac = 1 - 4 \cdot h \cdot (y_n + h(x_n + h)^2)$$

$$y_{n+1} = \frac{-b \pm \sqrt{D}}{2a} = \frac{1 \pm \sqrt{D}}{2h}$$

Реализация явного и неявного методов Эйлера и метода Пикара приведена в листинге 1.

Листинг 1. Код программы.

```
from prettytable import PrettyTable
from math import ceil, sqrt

def f1(x):
    return (x ** 3) / 3

def f2(x):
    return f1(x) + (x ** 7) / 63

def f3(x):
    return f2(x) + 2 * (x ** 11) / 2079 + (x ** 15) / 59535

def f4(x):
    return f3(x) + 4 * (x ** 15) / 93555 + 2 * (x ** 19) / 3393495 + \
           4 * (x ** 19) / 2488563 + 2 * (x ** 23) / 86266215 + \
           4 * (x ** 23) / 99411543 + 4 * (x ** 27) / 3341878155 + \
           (x ** 31) / 109876902975

def pikar(n, h, x, y0):
    result = [[y0], [y0], [y0], [y0]]
    for i in range(n -1):
        x += h
        # y_f1 = f1(x)
        # y_f2 = f2(x)
        y_f3 = f3(x)
        y_f4 = f4(x)
        # result[0].append(y_f1)
        # result[1].append(y_f2)
        result[2].append(y_f3)
        result[3].append(y_f4)
    return result

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

def explicit_euler(n, h, x, y0):
    result = []
    for i in range(n):
        try:
            y0 += h * f(x, y0)
```

```
            result.append(y0)
            x += h
        except OverflowError:
            result.append("Overflow")
            for j in range(i, n):
                result.append("Overflow")
            break
    return result

def implicit_euler(n, h, x, y0):
    result = [y0]
    for i in range(n):
        D = 1 - 4 * h * (y0 + h * ((x + h) ** 2))
        # D = 1 - 4 * h * y0 - 4 * (h ** 2) * ((x + h) ** 2)
        if D < 0:
            result.append("D < 0")
            for j in range(i, n):
                result.append("D < 0")
            break
        y0 = (1 - sqrt(D)) / (2 * h)
        x += h
        result.append(y0)
    return result

def main():
    X = 0
    H = 10 ** -6
    Y0 = 0
    end = 2.1
    N = ceil(abs(end - X) / H) + 1  # количество повторений

    res = pikar(N, H, X, Y0)
    res_exlp = explicit_euler(N, H, X, Y0)
    res_impl = implicit_euler(N, H, X, Y0)
    table = PrettyTable()
    table.field_names = ["X", "Pikar3", "Pikar4", "Explicit Euler", "Implicit
Euler"]
    i_start = 0
    i_range = N
    if i_start + i_range > N:
        i_range = N - i_start
    X += i_start * H
    i_step = int(N / 100)
    for i in range(i_start, i_start + i_range, i_step):
        x = "{:^9.5f}".format(X)
        # r1 = "{:^15.15f}".format(res[0][i])
        # r2 = "{:^15.15f}".format(res[1][i])
        r3 = "{:^15.15f}".format(res[2][i])
        r4 = "{:^15.15f}".format(res[3][i])
        try:
            ex = "{:^15.15f}".format(res_exlp[i])
```

```
            except ValueError:
                ex = "Overflow"

            try:
                im1 = "{:^15.15f}".format(res_impl[i])
            except ValueError:
                im1 = "D < 0"
            table.add_row([x, r3, r4, ex, im1])
            X += (H * i_step)
        print(table)

main()
```