Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

Лабораторная работа №6

По вычислительной математике

Вариант 3

Выполнил:
Студент группы P3216
Векшин Арсений Иванович
Преподаватель:
Малышева Татьяна Алексеевна

г. Санкт-Петербург

2024 г.

Метод Эйлера

```python
def solve(f, x0, y0, h, n):
    result = [y0]
    prev_y = y0

    for i in range(n-1):
        prev_y += h * f(x0 + i*h, prev_y)
        result.append(prev_y)

    return result
```

Улучшенный метод Эйлера

```python
def solve(f, x0, y0, h, n):
    result = [y0]
    prev_x, prev_y = x0, y0
    x = prev_x + h

    for i in range(n-1):
        prev_y += (h/2) * (f(prev_x, prev_y) + f(x, prev_y + h*f(prev_x, prev_y)))
        prev_x, x = x, x + h
        result.append(prev_y)

    return result
```

Метод Рунге-Кутта

```python
def solve(f, x0, y0, h, n):
    n -= 1
    result = [y0]
    prev_y = y0
    x = x0

    for i in range(n):
        k1 = h * f(x, prev_y)
        k2 = h * f(x + h/2, prev_y + k1/2)
        k3 = h * f(x + h/2, prev_y + k2/2)
        k4 = h * f(x + h, prev_y + k3)
        x += h

        prev_y += 1/6 * (k1 + 2*k2 + 2*k3 + k4)
        result.append(prev_y)
    return result
```

Метод Адамса

```python
from methods import RungeKutta


def solve(f, x0, y0, h, n):
    result = RungeKutta.solve(f, x0, y0, h, 4)
    pre_f = []

    for i in range(4):
        pre_f.append(f(x0 + h*i, result[i]))


    prev_y = result[-1]

    for i in range(4, n):

        df = pre_f[-1] - pre_f[-2]
        d2f = pre_f[-1] - 2*pre_f[-2] + pre_f[-3]
        d3f = pre_f[-1] - 2*pre_f[-2] + 3*pre_f[-3] - pre_f[-4]

        prev_y += h*pre_f[-1] + (h**2)/2 * df + 5*(h**3)/12 * d2f + 3*(h**4)/8 * d3f
        result.append(prev_y)
        pre_f.append(f(x0 + h*i, result[-1]))

    return result
```

Метод Милна

```python
from methods import RungeKutta

def solve(f, x0, y0, h, n, accuracy):
    n = n+1
    result = RungeKutta.solve(f, x0, y0, h, 3)
    xs = [x0 + h * i for i in range(n)]

    for i in range(4, n):
        # Предиктор
        yp = result[i - 4] + 4 * h * (2 * f(xs[i - 3], result[i - 3]) - f(xs[i - 2], result[i - 2])
                                      + 2 * f(xs[i - 1], result[i - 1])) / 3

        # Корректор
        y_next = yp
        while True:
            yc = result[i - 2] + h * (f(xs[i - 2], result[i - 2]) + 4 * f(xs[i - 1], result[i - 1])
                                      + f(xs[i], y_next)) / 3
            if abs(yc - y_next) < accuracy:
                y_next = yc
                break
            y_next = yc

        result.append(y_next)

    return result
```

Проверка точности

```
def accuracy_check(values, values_half):
    flag = True
    out = []
    for i in range(len(values)):
        err = (values[i]-values_half[2*i])/(2**methods[method_idx][2]-1)

        if err > accuracy: flag = False
        out.append(err)
    return out, flag
```

Полный код программы:
https://github.com/ArsenyVekshin/ITMO/tree/master/CompMath/lab6