Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики»

Факультет программной инженерии и компьютерной техники

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ 6

ЧИСЛЕННОЕ РЕШЕНИЕ ОБЫКНОВЕННЫХ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ

ВАРИАНТ 12

Студент: Пышкин Никита Сергеевич, P3213

Преподаватель:

Санкт Петербург 2025

Содержание

Цель лабораторной работы

Цель лабораторной работы: решить задачу Коши для обыкновенных дифференциальных уравнений численными методами.

Рабочие формулы

Метод Эйлера:

$$y_{i+1} = y_i + hf(x_i, y_i)$$

Метод Рунге-Кутта 4-го порядка:

$$y_{i+1} = y_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

$$k_1 = hf(x_i, y_i)$$

$$k_2 = hf(x_i + \frac{h}{2}, y_i + \frac{k_1}{2})$$

$$k_3 = hf(x_i + \frac{h}{2}, y_i + \frac{k_2}{2})$$

$$k_4 = hf(x_i + h, y_i + k_3)$$

Метод Милна:

а) этап прогноза:

$$y_i^{\text{прогн}} = y_{i-4} + \frac{4h}{3}(2f_{i-3} - f_{i-2} + 2f_{i-1})$$

б) этап коррекции:

$$y_i^{\text{корр}} = y_{i-2} + \frac{h}{3}(f_{i-2} + 4f_{i-1} + f_i^{\text{прогн}})$$

$$f_i^{\text{прогн}} = f(x_i, y_i^{\text{прогн}})$$

Листинг программы

euler_method.py

```
from .accuracy import runge_rule


def euler_method(f, x0, y0, xn, h, use_runge=True, eps=0.01):
    x_vals = [x0]
    y_vals = [y0]
```

```
    while x_vals[-1] < xn:
        xi = x_vals[-1] + h
        xi_h2 = x_vals[-1] + h / 2

        yi = y_vals[-1] + h * f(xi, y_vals[-1])
        yi_h2 = y_vals[-1] + h / 2 * f(xi_h2, y_vals[-1])

        if not use_runge or runge_rule(yi, yi_h2, 1, eps):
            x_vals.append(xi)
            y_vals.append(yi)
        else:
            h = h / 2

        if h < 1e-8:
            raise ValueError("Невозможно вычислить значение с такой
точностью")

    return x_vals, y_vals
```

fourth_order_runge_kutta_method.py

```
from .accuracy import runge_rule


def fourth_order_runge_kutta_method(f, x0, y0, xn, h, use_runge=True,
eps=0.01):
    def calc_iter(x_last, y_last, h):
        xi = x_last + h

        k1 = h * f(xi,y_last)
        k2 = h * f(xi + h / 2, y_last + k1 / 2)
        k3 = h * f(xi + h / 2, y_last + k2 / 2)
        k4 = h * f(xi + h, y_last + k3)

        yi = y_last + 1 / 6 * (k1 + 2 * k2 + 2 * k3 + k4)
```

```
        return xi, yi

    x_vals = [x0]
    y_vals = [y0]

    while x_vals[-1] < xn:
        xi, yi = calc_iter(x_vals[-1], y_vals[-1], h)
        _, yi_h2 = calc_iter(x_vals[-1], y_vals[-1], h / 2)

        if not use_runge or runge_rule(yi, yi_h2, 4, eps):
            x_vals.append(xi)
            y_vals.append(yi)
        else:
            h = h / 2

        if h < 1e-8:
            raise ValueError("Невозможно вычислить значение с такой
точностью")

    return x_vals, y_vals
```

milne_method.py

```
def milne_method(f, x0, y0, xn, h, eps):
    x = x0
    y = y0
    x_values = [x]
    y_values = [y]

    for _ in range(3):
        k1 = h * f(x, y)
        k2 = h * f(x + h / 2, y + k1 / 2)
        k3 = h * f(x + h / 2, y + k2 / 2)
        k4 = h * f(x + h, y + k3)
        y += (k1 + 2 * k2 + 2 * k3 + k4) / 6
```

```
        x += h
        x_values.append(x)
        y_values.append(y)

    while x < xn:
        i = len(y_values) - 1

        f_im3 = f(x_values[i - 3], y_values[i - 3])
        f_im2 = f(x_values[i - 2], y_values[i - 2])
        f_im1 = f(x_values[i - 1], y_values[i - 1])

        y_prog = y_values[i - 4] + (4 * h / 3) * (2 * f_im3 - f_im2 +
2 * f_im1)
        x_prog = x + h

        y_corr_prev = y_prog
        for _ in range(100):
            f_prog = f(x_prog, y_corr_prev)
            y_corr = y_values[i - 2] + (h / 3) * (f_im2 + 4 * f_im1 +
f_prog)

            if abs(y_corr - y_corr_prev) <= eps:
                break

            y_corr_prev = y_corr

        x = x_prog
        y = y_corr
        x_values.append(x)
        y_values.append(y)

    return x_values, y_values
```

Результаты работы программы

```
Выберите ОДУ:
```

1. y' = x + y

2. y' = y * cos(x)

3. y' = x^2 - y

Введите номер уравнения (1-3): 2

Начальное значение x0: 1

Начальное значение y0: 1

Конечное значение xn: 10

Шаг h: 1

Точность eps: 0.01

Метод Эйлера:

[1.0, 1.03125, 1.0625, 1.09375, 1.125, 1.15625, 1.1875, 1.21875, 1.25,
1.28125, 1.3125, 1.34375, 1.375, 1.40625, 1.4375, 1.46875, 1.5,
1.53125, 1.5625, 1.59375, 1.625, 1.65625, 1.6875, 1.71875, 1.75,
1.78125, 1.8125, 1.84375, 1.875, 1.90625, 1.9375, 1.96875, 2.0,
2.03125, 2.0625, 2.09375, 2.125, 2.15625, 2.1875, 2.21875, 2.25,
2.28125, 2.3125, 2.34375, 2.375, 2.40625, 2.4375, 2.46875, 2.5,
2.53125, 2.5625, 2.59375, 2.625, 2.65625, 2.6875, 2.71875, 2.75,
2.78125, 2.8125, 2.84375, 2.875, 2.90625, 2.9375, 2.96875, 3.0,
3.03125, 3.0625, 3.09375, 3.125, 3.15625, 3.1875, 3.21875, 3.25,
3.28125, 3.3125, 3.34375, 3.375, 3.40625, 3.4375, 3.46875, 3.5,
3.53125, 3.5625, 3.59375, 3.625, 3.65625, 3.6875, 3.71875, 3.75,
3.78125, 3.8125, 3.84375, 3.875, 3.90625, 3.9375, 3.96875, 4.0,
4.03125, 4.0625, 4.09375, 4.125, 4.15625, 4.1875, 4.21875, 4.25,
4.28125, 4.3125, 4.34375, 4.375, 4.40625, 4.4375, 4.46875, 4.5,
4.53125, 4.5625, 4.59375, 4.625, 4.65625, 4.6875, 4.71875, 4.75,
4.78125, 4.8125, 4.84375, 4.875, 4.90625, 4.9375, 4.96875, 5.0,
5.03125, 5.0625, 5.09375, 5.125, 5.15625, 5.1875, 5.21875, 5.25,
5.28125, 5.3125, 5.34375, 5.375, 5.40625, 5.4375, 5.46875, 5.5,
5.53125, 5.5625, 5.59375, 5.625, 5.65625, 5.6875, 5.71875, 5.75,
5.78125, 5.8125, 5.84375, 5.875, 5.90625, 5.9375, 5.96875, 6.0,
6.03125, 6.0625, 6.09375, 6.125, 6.15625, 6.1875, 6.21875, 6.25,
6.28125, 6.3125, 6.34375, 6.375, 6.40625, 6.4375, 6.46875, 6.5,
6.53125, 6.5625, 6.59375, 6.625, 6.65625, 6.6875, 6.71875, 6.75,
6.78125, 6.8125, 6.84375, 6.875, 6.90625, 6.9375, 6.96875, 7.0,
7.03125, 7.0625, 7.09375, 7.125, 7.15625, 7.1875, 7.21875, 7.25,
7.28125, 7.3125, 7.34375, 7.375, 7.40625, 7.4375, 7.46875, 7.5,
7.53125, 7.5625, 7.59375, 7.625, 7.65625, 7.6875, 7.71875, 7.75,
7.78125, 7.8125, 7.84375, 7.875, 7.90625, 7.9375, 7.96875, 8.0,
8.03125, 8.0625, 8.09375, 8.125, 8.15625, 8.1875, 8.21875, 8.25,
8.28125, 8.3125, 8.34375, 8.375, 8.40625, 8.4375, 8.46875, 8.5,
8.53125, 8.5625, 8.59375, 8.625, 8.65625, 8.6875, 8.71875, 8.75,
8.78125, 8.8125, 8.84375, 8.875, 8.90625, 8.9375, 8.96875, 9.0,
9.03125, 9.0625, 9.09375, 9.125, 9.15625, 9.1875, 9.21875, 9.25,
9.28125, 9.3125, 9.34375, 9.375, 9.40625, 9.4375, 9.46875, 9.5,
9.53125, 9.5625, 9.59375, 9.625, 9.65625, 9.6875, 9.71875, 9.75,
9.78125, 9.8125, 9.84375, 9.875, 9.90625, 9.9375, 9.96875, 10.0]

```
[1.0, np.float64(1.016054588103449), np.float64(1.0315078152862966),
np.float64(1.0463085751113732), np.float64(1.0604068153274075),
np.float64(1.0737538485647342), np.float64(1.0863026631860673),
np.float64(1.0980082313143533), np.float64(1.1088278109839624),
np.float64(1.1187212393263997), np.float64(1.127651213709514),
np.float64(1.135583557801284), np.float64(1.142487469626291),
np.float64(1.14833574882474), np.float64(1.1531050005092245),
np.float64(1.1567758133413644), np.float64(1.1593329097161347),
np.float64(1.1607652662424106), np.float64(1.161066203039588),
np.float64(1.16023344072692), np.float64(1.1582691243587628),
np.float64(1.155179813949077), np.float64(1.1509764416257946),
np.float64(1.1456742358533811), np.float64(1.1392926135533776),
np.float64(1.1318550413313369), np.float64(1.1233888673780221),
np.float64(1.1139251259471217), np.float64(1.103498316615634),
np.float64(1.0921461608017793), np.float64(1.0799093382447782),
np.float64(1.0668312063379446), np.float64(1.0529575053489661),
np.float64(1.0383360526576115), np.float64(1.0230164291909367),
np.float64(1.007049661239821), np.float64(0.9904879007996369),
np.float64(0.9733841074941979), np.float64(0.9557917350186833),
np.float64(0.9377644248775343), np.float64(0.9193557100014147),
np.float64(0.9006187306077255), np.float64(0.8816059644266766),
np.float64(0.8623689731545717), np.float64(0.8429581667228389),
np.float64(0.82342258669052), np.float64(0.8038097097843359),
np.float64(0.7841652723287591), np.float64(0.7645331160331401),
np.float64(0.744955055337803), np.float64(0.7254707662697111),
np.float64(0.7061176965238147), np.float64(0.6869309962710749),
np.float64(0.6679434690003734), np.float64(0.649185541530541),
np.float64(0.6306852521814765), np.float64(0.6124682559702342),
np.float64(0.594557845598984), np.float64(0.5769749869264458),
np.float64(0.5597383675619323), np.float64(0.5428644571903356),
np.float64(0.5263675782258406), np.float64(0.5102599854001769),
np.float64(0.4945519529160252), np.float64(0.4792518678358389),
np.float64(0.46436632842884007), np.float64(0.44990024626228775),
np.float64(0.43585695089531273), np.float64(0.42223829611273955),
np.float64(0.40904476672053175), np.float64(0.3962755850120821),
np.float64(0.38392881610393176), np.float64(0.37200147142920087),
np.float64(0.36048960976576405), np.float64(0.3493884352628879),
np.float64(0.3386923920137027 7), np.float64(0.32839525480071813),
np.float64(0.3184902157169617), np.float64(0.3089699664357421),
np.float64(0.2998267759671416), np.float64(0.2910525637989144),
np.float64(0.282638968373382), np.float64(0.2745774109001602),
np.float64(0.26685915454720677), np.float64(0.2594753590898766),
np.float64(0.2524171311296368), np.float64(0.245675570021075),
np.float64(0.2392418096681 3413), np.float64(0.23310705636844545),
np.float64(0.22726262289855795), np.float64(0.2216999590431245),
np.float64(0.21641067877806344), np.float64(0.2113865843217189),
np.float64(0.20661968726944102), np.float64(0.20210222702612615),
np.float64(0.19782668674841491), np.float64(0.19378580700373724),
np.float64(0.1899725973474921), np.float64(0.1863803460126104),
np.float64(0.18300262789780056), np.float64(0.17983331103212713),
np.float64(0.17686656168440426), np.float64(0.17409684827636596),
np.float64(0.17151894424883907), np.float64(0.16912793002031942),
np.float64(0.16691919416753456), np.float64(0.16488843394785582),
np.float64(0.16303165527386465), np.float64(0.1613451722410369),
```

np.float64(0.159825606300425), np.float64(0.15846988515941857),
np.float64(0.15727524148516656), np.float64(0.1562392114770561),
np.float64(0.1553596333667634), np.float64(0.15463464589681297),
np.float64(0.15406268682128854), np.float64(0.1536424914653119),
np.float64(0.15337309137312208), np.float64(0.15325381306801852),
np.float64(0.15328427694104665), np.float64(0.15346439627907354),
np.float64(0.15379437643678703), np.float64(0.15427471415112337),
np.float64(0.15490619699064742), np.float64(0.15568990292644508),
np.float64(0.15662720000510214), np.float64(0.15771974609830927),
np.float64(0.1589694886975153), np.float64(0.16037866471582493),
np.float64(0.1619498002529783), np.float64(0.16368571027273487),
np.float64(0.1655894981353006), np.float64(0.16766455492057092),
np.float64(0.16991455847090806), np.float64(0.17234347207493306),
np.float64(0.17495554270639593), np.float64(0.17775529872461188),
np.float64(0.18074754693524217), np.float64(0.1839373689023929),
np.float64(0.18733011639515007), np.float64(0.19093140584382726),
np.float64(0.19474711167344386), np.float64(0.19878335837436628),
np.float64(0.20304651116273337), np.float64(0.20754316507636844),
np.float64(0.21228013234548923), np.float64(0.21726442787181355),
np.float64(0.2225032526447929), np.float64(0.22800397491987348),
np.float64(0.23377410898108672), np.float64(0.23982129130913046),
np.float64(0.24615325397664925), np.float64(0.2527777950949094),
np.float64(0.2597027461407485), np.float64(0.26693593599983273),
np.float64(0.2744851515721497), np.float64(0.2823580947985817),
np.float64(0.2905623359836127), np.float64(0.29910526330899345),
np.float64(0.3079940284567704), np.float64(0.3172354882877059),
np.float64(0.3268361425529838), np.float64(0.33680206765336024),
np.float64(0.3471388465007133), np.float64(0.3578514945823124),
np.float64(0.3689443823780737), np.float64(0.3804211543355042),
np.float64(0.3922846446657958), np.float64(0.4045367902873469),
np.float64(0.41717854130949467), np.float64(0.4302097695189504),
np.float64(0.4436291754037397), np.float64(0.4574341943236315),
np.float64(0.47162090251123273), np.float64(0.4861839236631399),
np.float64(0.5011163369546626), np.float64(0.516409587383415),
np.float64(0.5320533994151665), np.float64(0.5480356949682823),
np.float64(0.564342516829332), np.float64(0.5809579586403818),
np.float64(0.5978641026364699), np.float64(0.6150409663381263),
np.float64(0.6324664594168976), np.float64(0.6501163519500953),
np.float64(0.6679642552629091), np.float64(0.685981616520276),
np.float64(0.7041377281763053), np.float64(0.7223997533146995),
np.float64(0.7407327678188174), np.float64(0.7590998201944502),
np.float64(0.7774620097320462), np.float64(0.7957785835383983),
np.float64(0.8140070527915443), np.float64(0.8321033283780556),
np.float64(0.8500218758607264), np.float64(0.8677158894990885),
np.float64(0.8851374848077641), np.float64(0.9022379088914954),
np.float64(0.9189677675441684), np.float64(0.9352772678460897),
np.float64(0.9511164747432476), np.float64(0.9664355798486259),
np.float64(0.9811851804732752), np.float64(0.9953165666783294),
np.float64(1.0087820139429486), np.float64(1.0215350788716255),
np.float64(1.0335308952215152), np.float64(1.0447264674201848),
np.float64(1.0550809586697416), np.float64(1.064555970697436),
np.float64(1.0731158122176818), np.float64(1.080727753217405),
np.float64(1.0873622622663601), np.float64(1.092993224186398),
np.float64(1.097598135587598), np.float64(1.1011582759928593),

np.float64(1.1036588525233033), np.float64(1.1050891164011971),
np.float64(1.1054424498408895), np.float64(1.1047164222365886),
np.float64(1.102912814913322), np.float64(1.1000376140782486),
np.float64(1.0961009719874846), np.float64(1.0911171367224273),
np.float64(1.0851043513428094), np.float64(1.0780847235451343),
np.float64(1.0700840672986764), np.float64(1.06113171825122),
np.float64(1.051260324987988), np.float64(1.040505618485226),
np.float64(1.0289061623208124), np.float64(1.0165030863849644),
np.float64(1.0033398069724035), np.float64(0.9894617362318244),
np.float64(0.9749159839986989), np.float64(0.9597510550437109),
np.float64(0.9440165447326496), np.float64(0.9277628360163773),
np.float64(0.911040800554213), np.float64(0.8939015066240522),
np.float64(0.8763959362916152), np.float64(0.8585747141036372),
np.float64(0.8404878493401621), np.float64(0.8221844936141381),
np.float64(0.8037127153470951), np.float64(0.7851192923826326),
np.float64(0.7664495237294735), np.float64(0.747747061157416),
np.float64(0.7290537611068363), np.float64(0.7104095571192703),
np.float64(0.691852352756417), np.float64(0.6734179347505954),
np.float64(0.6551399059236701), np.float64(0.6370496372256604),
np.float64(0.6191762380800889), np.float64(0.6015465440815052),
np.float64(0.5841851209719979), np.float64(0.5671142837278624),
np.float64(0.550354129514522), np.float64(0.5339225832165292),
np.float64(0.5178354542189129), np.float64(0.5021065031049362),
np.float64(0.48674751694193114), np.float64(0.4717683918495527),
np.float64(0.4571772215817214), np.float64(0.44298039090280183),
np.float64(0.42918267259828613), np.float64(0.4157873270285199),
np.float64(0.4027962032089838), np.float64(0.390209840480576),
np.float64(0.3780275699165735), np.float64(0.3662476146979676),
np.float64(0.3548671887742875), np.float64(0.3438825932116103),
np.float64(0.33328930971213766), np.float64(0.3230820908695673),
np.float64(0.3132550468007421), np.float64(0.3038017278660986),
np.float64(0.294715203258784), np.float64(0.2859881353046186),
np.float64(0.27761284937212655), np.float64(0.2695813993435229),
np.float64(0.2618856286438121), np.float64(0.2545172268660876),
np.float64(0.2474677820668548), np.float64(0.24072882883593577),
np.float64(0.23429189227148692), np.float64(0.2281485280121603)]

Метод Рунге-Кутта:

[1.0, 1.5, 1.75, 2.0, 2.25, 2.5, 2.75, 3.0, 3.25, 3.5, 3.75, 4.0,
4.25, 4.5, 4.75, 5.0, 5.25, 5.5, 5.75, 6.0, 6.25, 6.5, 6.75, 7.0,
7.25, 7.5, 7.75, 8.0, 8.25, 8.5, 8.75, 9.0, 9.25, 9.5, 9.75, 10.0]

[1.0, np.float64(0.9155729097220886), np.float64(0.8496811405921072),
np.float64(0.74518789492795), np.float64(0.622681885915982),
np.float64(0.5013115562876641), np.float64(0.394134942345227),
np.float64(0.3071654303702456), np.float64(0.24100201541969835),
np.float64(0.1932589153554753), np.float64(0.16057969111626635),
np.float64(0.13985455420641196), np.float64(0.1287743753208319),
np.float64(0.12600216917253743), np.float64(0.13119275089422527),
np.float64(0.14498888736766358), np.float64(0.169025596284018),
np.float64(0.205883129798682), np.float64(0.2588289253367407),
np.float64(0.3310922098300191), np.float64(0.4244054259920238),
np.float64(0.5367904436942172), np.float64(0.6602027713762074),
np.float64(0.7794899874088577), np.float64(0.8744179716969671),

```
np.float64(0.9253362959579783), np.float64(0.9204997175342216),
np.float64(0.8610584417402272), np.float64(0.7605521938551039),
np.float64(0.6392411579802819), np.float64(0.5168119246934458),
np.float64(0.4072617796641771), np.float64(0.31748284028784085),
np.float64(0.2486541464073013), np.float64(0.1986541008087272),
np.float64(0.1641677542205023)]
```

```
Метод Милне:
```

```
[1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0]
```

```
[1.0, np.float64(1.0701532987460647), np.float64(0.4944983318977382),
np.float64(0.20174355775206448), np.float64(0.29759768205054304),
np.float64(0.22490571569161072), np.float64(0.3579825374186355),
np.float64(0.5853300953973649), np.float64(0.5028249263412505),
np.float64(0.263276491571117)]
```

```
Вектор погрешностей: [np.float64(1.5543122344752192e-15),
np.float64(0.9298204458443571), np.float64(2.5035889555213613),
np.float64(3.7977546025654996), np.float64(4.834766434818909),
np.float64(5.674009318704403), np.float64(6.1684670767209875),
np.float64(6.840617817647513), np.float64(8.349069563960997),
np.float64(9.749799353169099)]
```

Вывод

В ходе выполнения лабораторной работы я изучил методы решения ОДУ и реализовал их на языке Python.