МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский университет ИТМО»

ФАКУЛЬТЕТ ПРОГРАММНОЙ ИНЖЕНЕРИИ И КОМПЬЮТЕРНОЙ ТЕХНИКИ

ЛАБОРАТОРНАЯ РАБОТА №2

по дисциплине

‘ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА’

Вариант №1

Выполнил:
Студент группы P3207
Алферов Г. А.

Преподаватель:
Рыбаков С.Д.

УНИВЕРСИТЕТ ИТМО

Санкт-Петербург, 2025

Цель работы

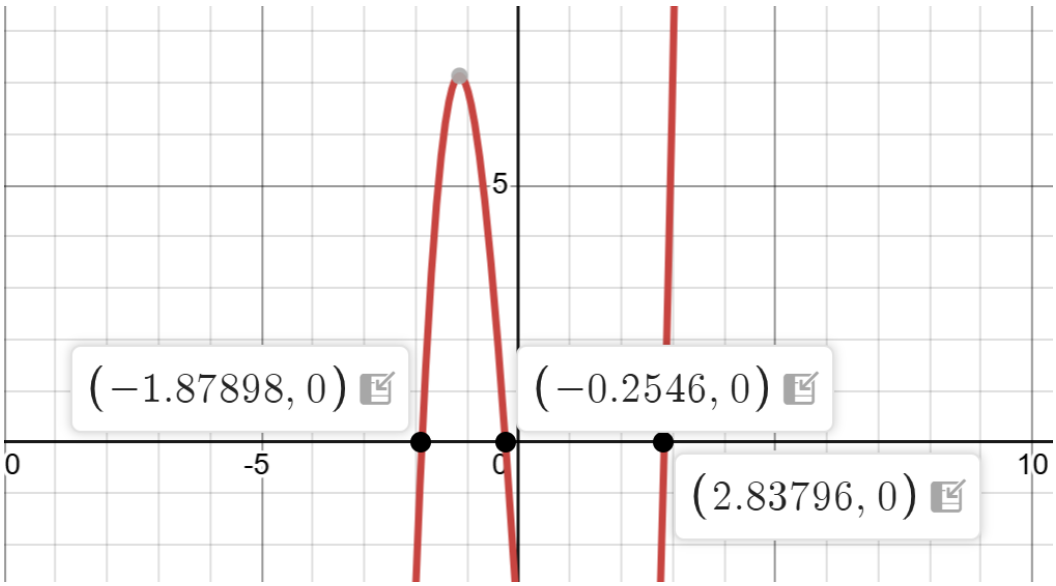
Изучить численные методы решения нелинейных уравнений и реализовать три из них средствами программирования. Понять их сходства и различия.

Ход работы

Часть 1.

$$2.74x^3 - 1.93x^2 - 15.28x - 3.72$$

Графическое отделение корней:



Интервалы изоляции корней:

Приближенные значения корней:

X ≈ -1.9, X ≈ -0.3, X ≈ 2.9

Интервалы на оси X:

(-∞, -1.9),(-1.9,-0,3),(-0,3,2.9),(2.9,+∞)

Вычисление знака функции на интервалах:

1 интервал: x = -2 f(-2) = -2.8 Знак: -

2 интервал: x = -1 f(-1) = 6.89 Знак: +

3 интервал: x = 0 f(0) = -3.72 Знак: -

4 интервал: x = 3 f(3) = 7.05 Знак: +

(-2,-1)(-1, 0) (2, 3)

Уточнение корней:

X1 ≈ -1.88

X2 ≈ -0.25

X3 ≈ 2.84

Правый корень – **метод половинного деления**

№ шага	a	b	x	f(a)	f(b)	f(x)	|a-b|
1	2.000	3.000	2.500	-20.079	7.050	-11.170	1.000
2	2.500	3.000	2.750	-11.170	7.050	-3.352	0.500
3	2.750	3.000	2.875	-3.352	7.050	1.509	0.250
4	2.750	2.875	2.812	-3.352	1.509	-1.004	0.125
6	2.812	2.875	2.843	-1.023	1.509	0.222	0.063
7	2.812	2.843	2.827	-1.023	0.202	-0.416	0.031
8	2.827	2.843	2.835	-0.436	0.202	-0.118	0.016
9	2.835	2.843	2.839	-0.118	0.202	0.041	0.008

Рабочая формула метода: $$x_i = \frac{a_i + b_i}{2}$$

Центральный корень – Метод простой итерации

№ итерации	X_k	X_{k+1}	$f(X_{k+1})$	$\lvert X_{k+1} - X_k \rvert$
1	-1.000	-0.549	3.634	0.451
2	-0.549	-0.311	0.765	0.238
3	-0.311	-0.261	0.089	0.050
4	-0.261	**-0.255**	0.009	**0.005**

Проверка сходимости:

Отрезок [-1, 0]

$f'(x) = 8.22x^2 - 3.86x - 15.28$

$f'(a) = -3.2 < 0$, $f'(b) = -15.28 < 0$

$\max(|f'(a)|, |f'(b)|) = 15.28$

$\lambda = 1/15.28$

$$\varphi(x) = x + \lambda f(x) = x + \frac{2.74x^3 - 1.93x^2 - 15.28x - 3.72}{15.28}$$

$$\varphi'(x) = 1 + \lambda f'(x) = 1 + \frac{8.22x^2 - 3.86x \quad - 15.28}{15.28}$$

$|\varphi'(a)| = 0.790$

$|\varphi'(b)| = 0$

$0 \le q < 1$ *Итерационная последовательность сходится*

Рабочая формула:

$$x_{i+1} = \varphi(x_i)$$

Левый корень – метод секущих

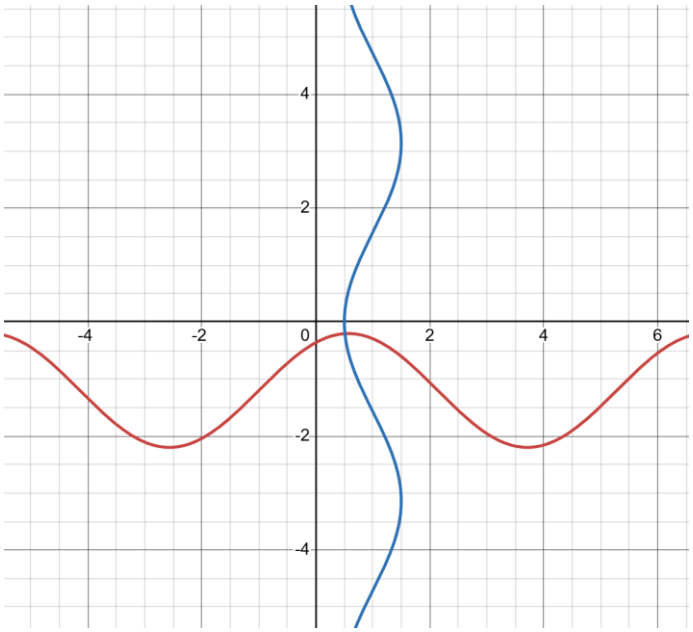
№ итерации	X_{k-1}	X_k	X_{k+1}	$f(X_{k+1})$	$\lvert X_{k+1} - X_k \rvert$
1	-2.000	-1.000	-1.711	3.049	0.711
2	-1.000	-1.711	-2.275	-11.227	0.564
3	-1.711	-2.275	-1.831	0.956	0.443
4	-2.275	-1.831	-1.866	0.264	0.035
5	-1.831	-1.866	-1.879	-0.011	0.013
6	-1.866	-1.879	**-1.878**	0.000004	**0.00002**

Рабочая формула:

$$x_{i+1} = x_i - \frac{x_i - x_{i-1}}{f(x_i) - f(x_{i-1})} f(x_i) \qquad i = 1,2 \ldots$$

Часть 2.

$\begin{cases} sin(x+1) - y = 1.2 \\ 2x + cosy = 2 \end{cases}$, Метод простой итерации



$\begin{cases} x = \varphi 1(x,y) \\ y = \varphi 2(x,y) \end{cases}$

$\begin{cases} x = 1 - cosy/2 \\ y = \sin(x+1) - 1.2 \end{cases}$

Проверка сходимости:

Условие: $|\partial\varphi 1/\partial x| + |\partial\varphi 1/\partial y| < 1$ и $|\partial\varphi 2/\partial x| + |\partial\varphi 2/\partial y| < 1$

$0 + |\sin(y)/2| < 1$ и $|\cos(x+1)| + 0 < 1$

Это условие выполняется.

Итерации:

Начальное приближение x0 = 0, y0 = 0

Формула: $x_{k+1} = 1 - \cos(y_k) / 2$, $y_{k+1} = \sin(x_k + 1) - 1.2$

1.

X1 = 1 – cos(0)/2 = 0.500

Y1 = sin(1) – 1.2 = -0.358

Проверка точности: X |0.500-0| = 0.500 > 0.01 и Y |-0.358-0| > 0.01

2.

X2 = 1 – cos(-0.358)/2 = 0.532

Y2 = sin(1.5) – 1.2 = -0.202

Проверка точности: X |0.532-0.500| = 0.032 > 0.01 и Y |-0.202+0.358| = 0.156 > 0.01

3.

X3 = 1 – cos(-0.202)/2 = 0.510

Y3 = sin(1.532) – 1.2 = -0.200

Проверка точности: X |0.510-0.532| = 0.022 > 0.01 и Y |-0.200+0.202| = 0.002 < 0.01

4.

X3 = 1 – cos(-0.200)/2 = 0.509

Y3 = sin(1.510) – 1.2 = -0.200

Проверка точности: X |0.509-0.510| = 0.001 < 0.01 и Y |-0.200+0.200| = 0 < 0.01

Решение системы с точностью до 0.01:

X≈0.509, Y≈-0.200

Код используемых методов.

Метод Хорд

```
import additional_functions.validation
import properties.properties
import tasks.task


def calc(e, a, b, f):
    func = tasks.task.get_function(f)
    for n in range(properties.properties.max_iter):
        x_n = (a * func(b) - b * func(a)) / (func(b) - func(a))
        if abs(x_n - b) < e or abs(func(x_n)) < e:
            return x_n, func(x_n), n + 1
        elif additional_functions.validation.has_root(f, x_n, a):
            b = x_n
        else:
            a = x_n
    return 0, 0, 1000
```

Метод Ньютона

```
import additional_functions.derivative
import properties.properties
import tasks.task


def calc_newton(f, x, e):
    func = tasks.task.get_function(f)
    df = additional_functions.derivative.fabric_df(func)

    x0 = x
    for i in range(properties.properties.max_iter):
        fx = func(x0)
        dfx = df(x0)
        if dfx == 0:
            print("Производная равна нулю!")
            return 0, 0, 0
        x_next = x0 - fx/dfx

        if abs(x_next - x0) <= e:
            return x_next, func(x_next), i + 1

        x0 = x_next

    return 0, 0, properties.properties.max_iter
```

Метод простых итераций

```
import numpy as np
import additional_functions.derivative
import additional_functions.validation
import properties.properties
import tasks.task


def phi_calc(func, lambda_):
    return lambda x: x + lambda_ * func(x)


def calc(f, a, b, e):
    func = tasks.task.get_function(f)
    df = additional_functions.derivative.fabric_df(func)
    max_df = max(abs(df(a)), abs(df(b)))
    if max_df == 0:
        print('Производная равна нулю на интервале')
        return 0, 0, 0
    if df(a) > 0 and df(b) > 0:
        lambda_ = -1 / max_df
    elif df(a) < 0 and df(b) < 0:
        lambda_ = 1 / max_df
    else:
        print("Не выполняется условие для получения lambda")
        return 0, 0, 0

    phi = phi_calc(func, lambda_)

    dhi = additional_functions.derivative.d_phi(df, lambda_)

    for x in np.linspace(a, b, 10):
        if abs(dhi(x)) >= 1:
            print(f"Условие сходимости нарушено на интервале")
            return 0, 0, 0

    x0 = (a + b) / 2
```

```
    for i in range(properties.properties.max_iter):
        x_next = phi(x0)

        if abs(x_next - x0) <= e:
            return x_next, func(x_next), i + 1
        x0 = x_next

    return 0, 0, properties.properties.max_iter
```

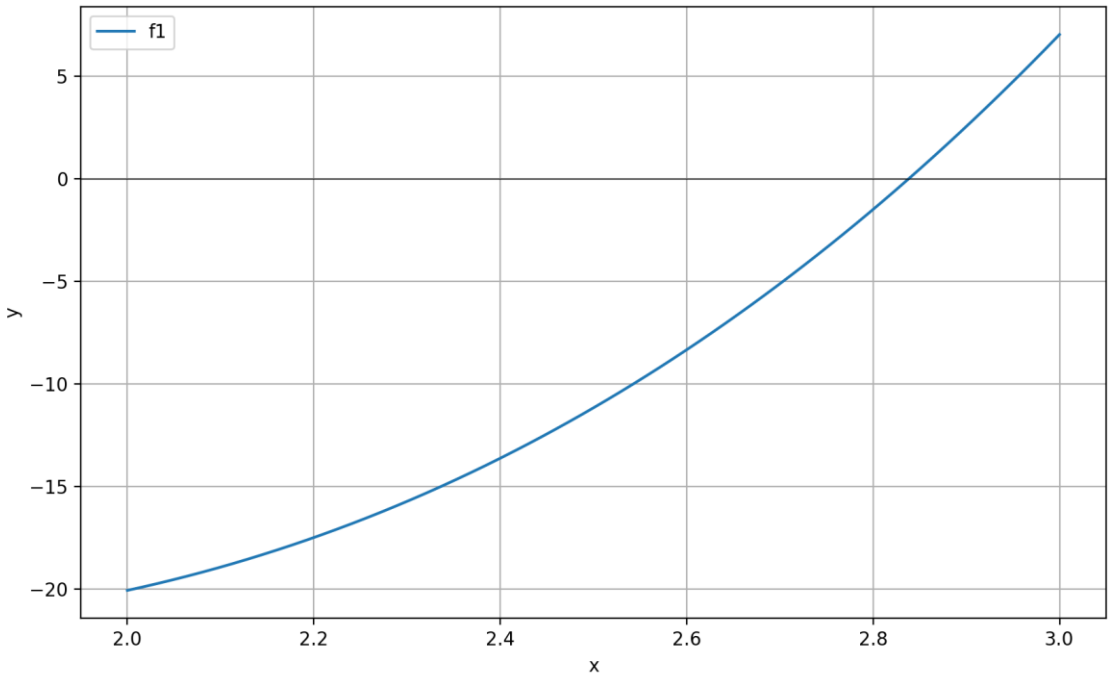
Метод Ньютона для системы

```
import additional_functions.derivative
import numpy as np

import properties.properties
import tasks.task


def build_jac(f, x, y):
    df1_dx = additional_functions.derivative.partial_d(f, x, y, 'x', 0)
    df1_dy = additional_functions.derivative.partial_d(f, x, y, 'y', 0)
    df2_dx = additional_functions.derivative.partial_d(f, x, y, 'x', 1)
    df2_dy = additional_functions.derivative.partial_d(f, x, y, 'y', 1)

    J = np.array([[df1_dx, df1_dy], [df2_dx, df2_dy]])

    return J


def newton_system_calc(f, x, y, e):
    x0, y0 = x, y
    system = tasks.task.get_system(f)
    print("№ |     x0    |     y0     |     x_n    |     y_n     | |x_n - x0| |
|y_n - y0|")
    for i in range(properties.properties.max_iter):
        f1_val = system[0](x0, y0)
        f2_val = system[1](x0, y0)

        J = build_jac(f, x0, y0)

        if np.abs(np.linalg.det(J)) < 1e-12:
            print("Матрица близка к 0!")
            return 0, 0, 0, 0, 0

        rhs = -np.array([f1_val, f2_val])
        dx, dy = np.linalg.solve(J, rhs)

        x_n = x0 + dx
        y_n = y0 + dy

        print(
            f"{i + 1} | {x0:.6f} | {y0:.6f} | {x_n:.6f} | {y_n:.6f} |
{abs(x_n - x0):.6f} | {abs(y_n - y0):.6f}")

        if abs(x_n - x0) <= e and abs(y_n - y0) <= e:
            return x_n, y_n, i + 1, abs(x_n - x0), abs(y_n - y0)

        x0 = x_n
        y0 = y_n

    print("Метод не сошелся за максимальное кол-во итераций")
    return 0, 0, properties.properties.max_iter, 0, 0
```

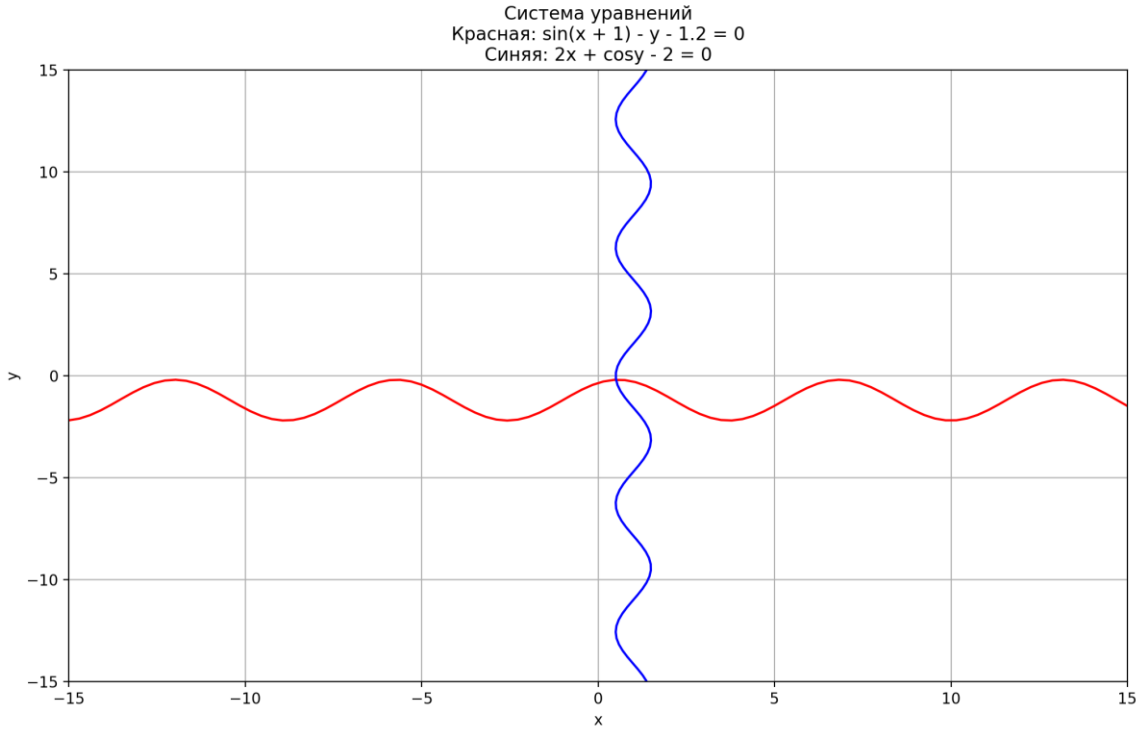
https://github.com/wrakelft/CompMathITMO/tree/main/lab2

Результат работы программы.

```
Выберите объект решения: уравнение - y или систему уравнений - s: y
Выберите уравнение:
 1. 2.74x^3 - 1.93x^2 - 15.28x - 3.72
 2. 2x^3 + 3.41x^2 - 23.74x + 2.95
 3. e^-x - x^2
1
Выберите метод решения:
 1. Метод хорд
 2. Метод Ньютона
 3. Метод простых итераций
1
Выберите откуда ввести границы интервала/начальное приближение и погрешность. f - из файла, k - с клавиатуры: k
Выберите куда вывести результаты. f - в файл, m - на экран: m
Введите точность системы >0 и <1: 0.01
Введите интервал через пробел: 2 3
x: 2.837909
f(x): -0.0021919883778251936
Кол-во итераций: 4
Закройте окно с показом функции чтобы завершить программу
```



```
Выберите объект решения: уравнение - y или систему уравнений - s: s
Выберите систему:
 1. sin(x + 1) - y - 1.2
 2x + cosy - 2
 2. sin(x + y) - 1.1x - 0.1
 x^2 + y^2 - 1
1
Выберите откуда ввести начальное приближение и погрешность. f - из файла, k - с клавиатуры: k
Выберите куда вывести результаты. f - в файл, m - на экран: m
Введите точность системы >0 и <1: 0.01
Введите x0 и y0 для системы через пробел: 0 0
№ |     x0    |     y0    |    x_n    |    y_n    | |x_n - x0| | |y_n - y0|
1 | 0.000000 | 0.000000 | 0.500000 | -0.088378 | 0.500000 | 0.088378
2 | 0.500000 | -0.088378 | 0.506966 | -0.202012 | 0.006966 | 0.113634
3 | 0.506966 | -0.202012 | 0.510150 | -0.201833 | 0.003183 | 0.000179
Вектор неизвестных: 0.510150, -0.201833
Кол-во итераций: 3
Вектор погрешностей: 0.003183, 0.000179
```



Выводы

Во время выполнения лабораторной работы я познакомился с методами решения нелинейных уравнений. Применил их для ручного вычисления значений и реализовал методы Хорд, Ньютона и простой итерации на Python.