Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

«Национальный исследовательский университет ИТМО»

Мегафакультет компьютерных технологий и управления

Факультет программной инженерии и компьютерной техник

Образовательная программа: «Компьютерные технологии в дизайне»

Отчет

«Численное решение нелинейных уравнений и систем».

Вариант 15

Студенты: Савельева Елизавета, Фурзикова Александра

Группа КТвД

Практик: Машина Екатерина Алексеевна

Санкт-Петербург, 2025

Содержание

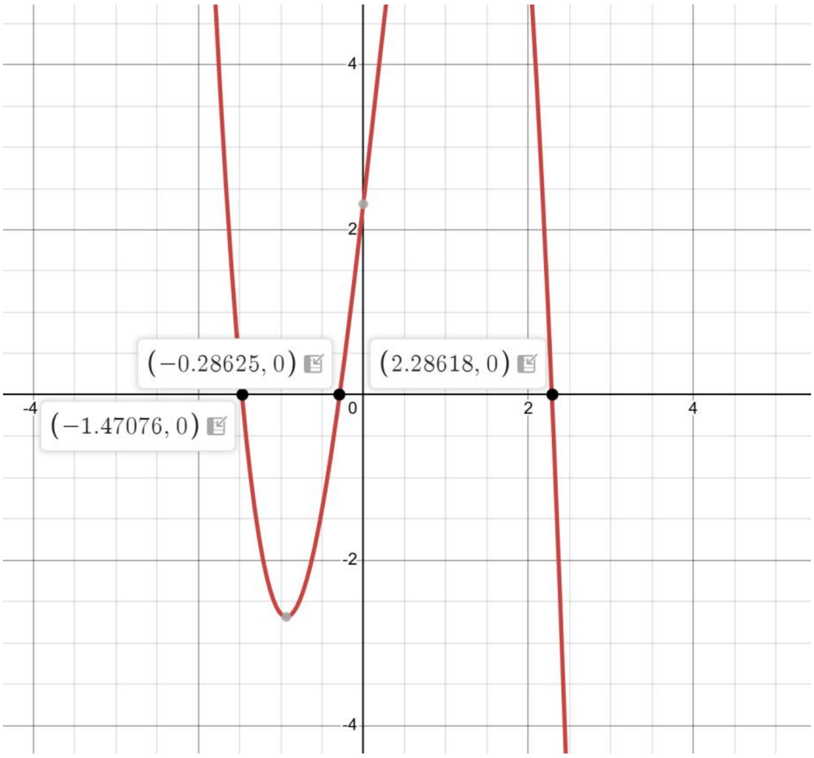
Цель лабораторной работы

Изучить и научиться решать нелинейные уравнения и системы уравнений численными методами: метод половинного деления, метод Ньютона, метод хорд, метод простой итерации. Реализовать алгоритмы решения уравнений на языке Python.

1. Вычислительная реализация задачи

<u>1 часть. Решение нелинейного уравнения</u>

Отделение корней заданного нелинейного уравнения графически



Определение интервалов изоляции корней:

x	f(x)
-3	52.65
-2.5	26.172
-2	9.330
-1.5	0.322
-1	-2.65
-0.5	-1.388
0	2.31
0.5	6.642
1	9.81

1.5	10.013
2	5.45
2.5	-5.677
3	-25.17
3.5	-54.827
4	-96.45

Левый корень: интервал изоляции: [−1.5, −1]

Центральный корень: интервал изоляции: [-0.5, 0]

Правый корень: интервал изоляции: [2, 2.5]

Уточнение корня уравнения методом секущих (Центральный корень)

① Центральный корень (метод секущих)

$$f(x) = -2{,}4x^3 + 1{,}27x^2 + 8{,}63x + 2{,}31, \quad \varepsilon = 0{,}01$$

$x_0 = -0{,}5 \quad x_1 = 0$

$$x_{k+1} = x_{k-1} - \frac{f(x_k)(x_k - x_{k-1})}{f(x_k) - f(x_k - 1)}$$

① Итерация

$x_0 = -0{,}5 \quad x_1 = 0$

$f(x_0) = f(-0{,}5) = -1{,}987$

$f(x_1) = f(0) = 2{,}31$

$$x_2 = -0{,}5 - \frac{-1{,}9875(0+0{,}5)}{2{,}31 + 1{,}987} \approx -0{,}2688$$

$f(x_1) = -0{,}0526$

$|x_1 - a| = |-0{,}046 + 0{,}5| = 0{,}231 > \varepsilon$

② Итерация

$x_1 = 0 \quad x_2 = -0{,}2688$

$x_3 \approx -0{,}2364$

$f(-0{,}2364) \approx -0{,}0002$

$|x_3 - x_2| > \varepsilon$

③ Итерация

$x_2 = -0{,}2688 \quad x_3 = -0{,}2364$

$x_4 = -0{,}2360$

$|x_4 - x_3| = 0{,}0004 < \varepsilon \Rightarrow$ Центральный корень: $x = -0{,}236$

Центральный корень: -0.286

№ итерации	x_{k-1}	x_k	x_{k+1}	$f(x_{k+1})$	$\lvert x_{k+1} - x_k \rvert$
1	-0.5	0	-0.269	-0.053	0.231
2	0	-0.269	-0.2864	-0.0001	0.018
3	-0.269	-0.2864	-0.2860	0.0002	0.0004

Уточнение корня уравнения методом хорд (Крайний правый корень)

№ итерации	a	b	x	f(a)	f(b)	f(x)	|x(k+1)-x(k)|
1	2	2.5	2.245	5.45	-5.677	0.929	0.245
2	2.245	2.5	2.281	0.929	-5.677	0.119	0.036
3	2.281	2.5	2.285	0.119	-5.677	0.027	0.004

Крайний правый корень: 2.285

2) Правый корень, метод простой итерации

2) $f(x) = -2,4x^3 + 1,27x^2 + 8,63x + 2,31$ $[2; 2,5]$

$f'(x) = -7,2x^2 + 2,54x + 8,63$ $\varepsilon = 0,01$

Начальное приближение: $x_0 = 2$

$x_{k+1} = x_k + \lambda f(x_k)$

Проверка сходимости:

$f'(2) = -7,2 \cdot 4 + 2,54 \cdot 2 + 8,63 = -15,09$

$f'(2,5) = -30,02$

$$\lambda = -\frac{1}{\max(|f'(a)|, |f'(b)|)} = -\frac{1}{30,02} = -0,0333$$

$\varphi(x) = x + \lambda f(x) = x - 0,0333(-2,4x^3 + 1,27x^2 + 8,63x + 2,31)$

$\varphi'(x) = 1 + \lambda f'(x) = 1 - 0,0333(-7,2x^2 + 2,54x + 8,63)$

$\left.\begin{array}{l}\varphi'(2) = 1,498 > 1 \\ \varphi'(2,5) = 1,991 > 1\end{array}\right\}$ метод не сходится к корню, воспользуемся методом хорд

Метод хорд

$$x_{k+1} = a - \frac{f(a)(b-a)}{f(b) - f(a)}$$ интервал $[2; 2,5]$

1. $a = 2$ $b = 2,5$

$f(a) = f(2) = -2,4 \cdot 8 + 1,27 \cdot 4 + 8,63 \cdot 2 + 2,31 = 5,45$

$f(b) = f(2,5) = -5,677$ $f(x_1) = 0,929$

$$x_1 = 2 - \frac{5,45 \cdot 0,5}{-5,677 - 5,45} \approx 2,245$$

$|x_{k+1} - x_k| = |x_1 - a| = |2,245 - 2| = 0,245 > \varepsilon$

2. $a = 2,245$ $b = 2,5$

$f(a) = f(2,245) = 0,929$

$f(b) = -5,677$ $f(x_2) = 0,119$

$$x_2 = 2,245 - \frac{0,929 \cdot 0,255}{-5,677 - 0,929} \approx 2,281$$

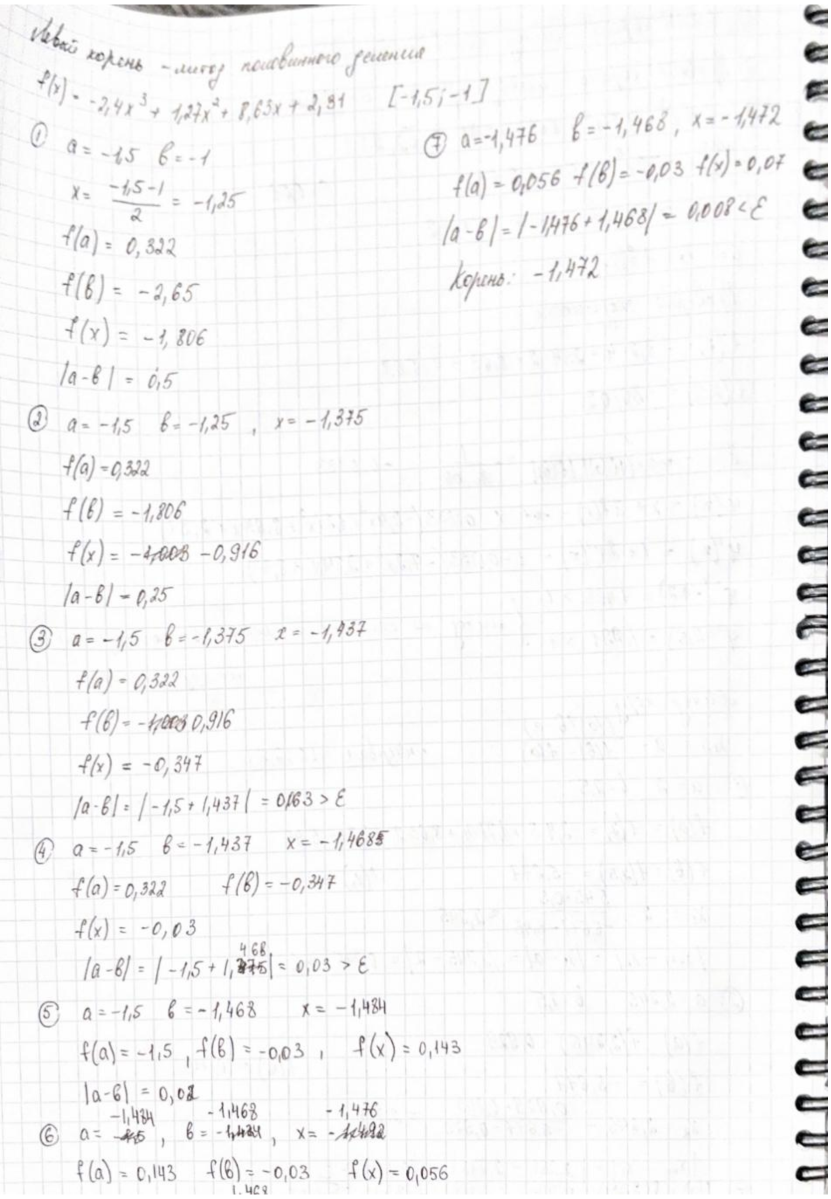
$|x_{k+1} - x_k| = |2,281 - 2,245| = 0,036 > \varepsilon$

3. $f(a) = f(2,281) = 0,119$; $f(b) = f(2,5) = -5,677$

$$x_3 = 2,281 - \frac{0,119 \cdot 0,219}{-5,677 - 0,119} = 2,285$$ $f(x_3) = 0,027$

$|x_{k+1} - x_k| = |x_2 - x_1| = |2,285 - 2,281| = |0,004| < \varepsilon$ Корень: 2,285

Уточнение корня уравнения методом половинного деления (Крайний левый корень)


Левый корень – метод половинного деления

$f(x) = -2{,}4x^3 + 1{,}27x^2 + 8{,}63x + 2{,}31 \quad [-1{,}5; -1]$

① $a = -1{,}5 \quad b = -1$

$x = \frac{-1{,}5 - 1}{2} = -1{,}25$

$f(a) = 0{,}322$

$f(b) = -2{,}65$

$f(x) = -1{,}806$

$|a-b| = 0{,}5$

② $a = -1{,}5 \quad b = -1{,}25, \quad x = -1{,}375$

$f(a) = 0{,}322$

$f(b) = -1{,}806$

$f(x) = \cancel{-1{,}003}\ -0{,}916$

$|a-b| = 0{,}25$

③ $a = -1{,}5 \quad b = -1{,}375 \quad x = -1{,}437$

$f(a) = 0{,}322$

$f(b) = \cancel{-1{,}003}\ 0{,}916$

$f(x) = -0{,}347$

$|a-b| = |-1{,}5 + 1{,}437| = 0{,}063 > \varepsilon$

④ $a = -1{,}5 \quad b = -1{,}437 \quad x = -1{,}4685$

$f(a) = 0{,}322 \qquad f(b) = -0{,}347$

$f(x) = -0{,}03$

$|a-b| = |-1{,}5 + 1{,}468| = 0{,}03 > \varepsilon$

⑤ $a = -1{,}5 \quad b = -1{,}468 \quad x = -1{,}484$

$f(a) = -1{,}5, \quad f(b) = -0{,}03, \quad f(x) = 0{,}143$

$|a-b| = 0{,}02$

⑥ $a = -1{,}484, \quad b = -1{,}468, \quad x = -1{,}476$

$f(a) = 0{,}143 \quad f(b) = -0{,}03 \quad f(x) = 0{,}056$

⑦ $a = -1{,}476 \quad b = -1{,}468, \quad x = -1{,}472$

$f(a) = 0{,}056 \quad f(b) = -0{,}03 \quad f(x) = 0{,}07$

$|a-b| = |-1{,}476 + 1{,}468| = 0{,}008 < \varepsilon$

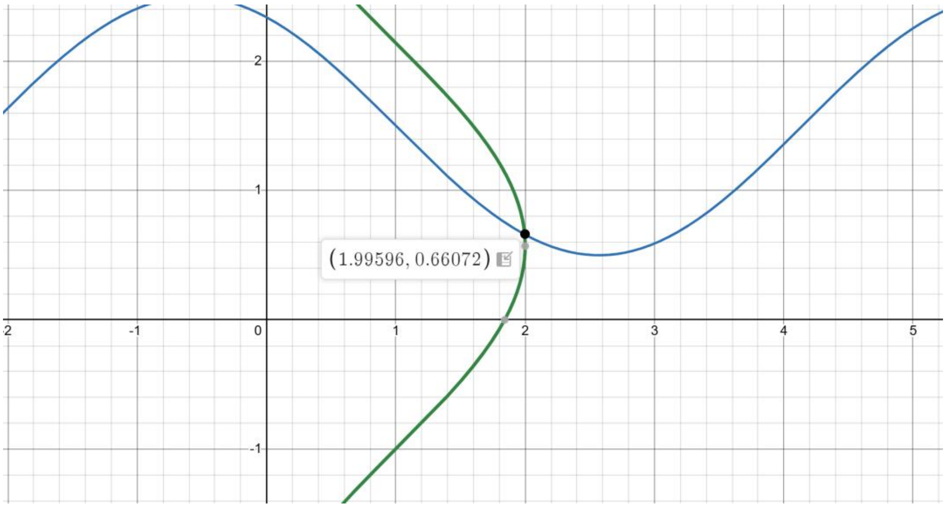
Корень: $-1{,}472$

№ итерации	a	b	x	f(a)	f(b)	f(x)	|a-b|
1	-1.5	-1	-1.25	0.322	-2.65	-1.806	0.5
2	-1.5	-1.25	-1.375	0.322	-1.806	-0.916	0.25
3	-1.5	-1.375	-1.437	0.322	0.916	-0.347	0.1
4	-1.5	-1.437	-1.468	0.322	-0.347	-0.03	0.06
5	-1.5	-1.468	-1.484	0.322	-0.03	0.143	0.03
6	-1.484	-1.468	-1.476	0.143	-0.03	0.056	0.02
7	-1.476	-1.468	-1.472	0.056	-0.03	0.07	0.008

Крайний левый корень: -1.472

2 часть. Решение системы нелинейных уравнений

Отделение корней заданной системы нелинейных уравнений графически



Решение системы методом простой итерации:

7

$$\begin{cases} \sin(x-1)+y=1{,}5 \\ x-\sin(y+1)=1 \end{cases} \qquad \varepsilon=0{,}01$$

Приведем к эквивалентному виду

$$\begin{cases} y=1{,}5-\sin(x-1) \\ x=1+\sin(y+1) \end{cases}$$

Выберем начальное приближение

$x^0=(x_0,y_0)=(1;0{,}5)$

Шаг 1

$x_1=1+\sin(1{,}5)=1{,}997$

$y_1=1{,}5-\sin 0=1{,}5$

Проверим точность

$|x_1-x_0|=|1{,}997-1|=0{,}997>\varepsilon$

$|y_1-y_0|=|1{,}5-0{,}5|=1>\varepsilon$

Шаг 2

$x_2=1+\sin(y_1+1)=1+\sin(2{,}5)=1{,}598$

$y_2=1{,}5-\sin(x_1-1)=1{,}5-\sin(1{,}997-1)=0{,}66$

Проверим точность

$|x_2-x_1|=|1{,}598-1{,}997|=0{,}399>\varepsilon$

$|y_2-y_1|=|0{,}66-1{,}5|=0{,}84>\varepsilon$

Шаг 3

$y_3=1{,}5-\sin(x_2-1)=0{,}937$

$x_3=1+\sin(y_2+1)=1{,}996$

$|x_3-x_2|=|1{,}996-1{,}598|=0{,}398>\varepsilon$

$|y_3-y_2|=|0{,}937-0{,}66|=0{,}277>\varepsilon$

Шаг 4

$y_4=1{,}5-\sin(x_3-1)=0{,}661$

$x_4=1+\sin(y_3+1)=1{,}934$

$|x_4-x_3|=|1{,}934-1{,}996|=0{,}062>\varepsilon$

$|y_4-y_3|=|0{,}661-0{,}937|=0{,}276>\varepsilon$

Шаг 5

$y_5 = 1{,}5 - \sin(x_4 - 1) = 1{,}5 - \sin(1{,}934 - 1) = 0{,}696$

$x_5 = 1 + \sin(y_4 + 1) = 1 + \sin(0{,}661 + 1) = 1{,}996$

$|x_5 - x_4| = |1{,}996 - 1{,}934| = 0{,}062 > \varepsilon$

$|y_5 - y_4| = |0{,}696 - 0{,}661| = 0{,}035 > \varepsilon$

Шаг 6

$y_6 = 1{,}5 - \sin(x_5 - 1) = 0{,}661$

$x_6 = 1 + \sin(y_5 + 1) = 1 + \sin(0{,}696 + 1) = 1{,}992$

$|x_6 - x_5| = |1{,}992 - 1{,}996| = 0{,}004 < \varepsilon$

$|y_6 - y_5| = |0{,}661 - 0{,}696| = 0{,}03 > \varepsilon$

Шаг 7

$y_7 = 1{,}5 - \sin(x_6 - 1) = 0{,}663$

$x_7 = 1 + \sin(y_6 + 1) = 1{,}996$

Проверим точность:

$|x_7 - x_6| = |1{,}996 - 1{,}992| = 0{,}004 < \varepsilon$

$|y_7 - y_6| = |0{,}663 - 0{,}661| = 0{,}002 < \varepsilon$

Таким образом решение системы:

$$\begin{cases} x \approx 1{,}996 \\ y = 0{,}663 \end{cases}$$

2 Программная реализация задачи:

Main

```
# variant number: 15


# 3 - Метод Ньютона
# 5 - Метод простой итерации
# 6 - Метод Ньютона


from math import sqrt

import numpy as np

from solvers import horde_method, newton_method, simple_iteration_method,
system_newton_method
from graph import show_2d

A, B, C, D = -2.40, 1.27, 8.63, 2.31
```

```python
f = lambda x: A * x ** 3 + B * x ** 2 + C * x + D
df = lambda x: 3 * A * x ** 2 + 2 * B * x + C
ddf = lambda x: 6 * A * x + 2 * B


def dphi(x):
    return abs(sqrt(abs(1/A)) / 3 * (B * x**2 + C * x + D) ** (-2/3) *
(2*B*x + C))


def phi(x):
    base = -(B * x ** 2 + C * x + D) / A
    if base < 0:
        return -(-base) ** (1 / 3)
    return base ** (1 / 3)


def non_linear(output_file):
    left, right = map(float, input('Границы левого корня через пробел (-2 -
1): ').split())
    #c_x0 = float(input('Нулевое приближение центрального корня (0): '))

    a, b = map(float, input('Границы для второго корня: (-1 0): ').split())
    if f(a) * ddf(a) > 0:
        c_x0 = a
    else:
        c_x0 = b

    print(f'x0 для второго корня равен равен: {c_x0}')

    # r_x0 = float(input('Нулевое приближение правого корня (5): '))
    a1, b1 = map(float, input('Границы для правого корня: (2 3): ').split())
    if f(a1) * ddf(a1) > 0:
        r_x0 = a1
    else:
        r_x0 = b1

    print(f'dpfi в a1 = {dphi(a1)}')
    print(f'dpfi в b1 = {dphi(b1)}')

    epsilon = float(input('Погрешность (0.01): '))

    left_x = horde_method(f, left=left, right=right, epsilon=epsilon)
    center_x = newton_method(f, df, c_x0, epsilon=epsilon)
    right_x = simple_iteration_method(f, phi, r_x0, epsilon=epsilon)

    show_2d(f, [(left_x.root, 0), (center_x.root, 0), (right_x.root, 0)])

    print(f'Левый корень x_0={left_x.root:.7f}, знач-е:
{left_x.znach_f:.7f}, n: {left_x.iter_count}')
    # Значение функции + количество итераций
    print(f'Центральный корень x_1={center_x.root:.7f}, знач-е:
{center_x.znach_f:.7f}, n: {center_x.iter_count}')
    # Значение функции + количество итераций
    print(f'Правый корень x_2={right_x.root:.7f}, знач-е:
{right_x.znach_f:.7f}, n: {right_x.iter_count}')
    # Значение функции + количество итераций

    print(f'Таблица для метода хорд:')
    print(left_x)

    print('Таблица для метода Ньютана:')
    print(center_x)

    print('Таблица для метода простой итерации:')
```

```python
    print(right_x)

    if output_file != None:
        with open(output_file, 'a') as fl:
            fl.write(f'Таблица для метода хорд:')
            fl.write(str(left_x))

            fl.write('Таблица для метода Ньютана:')
            fl.write(str(center_x))

            fl.write('Таблица для метода простой итерации:')
            fl.write(str(right_x))


def function_1(x, y):
    return [
        (-1)*(x+(2*y)-2),
        (-1)*((x**2)+(4*(y**2))-4)
    ]


def jacobian_1(x, y):
    return [
        [1, 2],
        [2*x, 8*y]
    ]


def function_2(x, y, z):
    return [
        (-1)*(x+y+z-3),
        (-1)*((x**2)+(y**2)+(z**2)-5),
        (-1)*((np.exp(x))+(x*y)-(x*z)-1)
    ]


def jacobian_2(x, y, z):
    return [[1, 1, 1], [2*x, 2*y, 2*z], [np.exp(x), x, -x]]


FUNCTIONS = [
    {
        'disp': 'f1(x1, x2) = x + 2*y - 2\nf2(x1, x2) = x^2 + 4*y^2 - 4)',
        'func': function_1,
        'jacob': jacobian_1,
        'init': [1, 1],
    },
    {
        'disp': 'f1(x1, x2, x3) = x + y + z - 3\nf2(x1, x2, x3) = x^2 + y^2
+ z^2 - 5\nf3(x1, x2, x3) = exp(x) + x*y - x*z -1',
        'func': function_2,
        'jacob': jacobian_2,
        'init': [100, 200, 3],
    },
]


def non_linear_system(output_file):
    print(f'Выберите систему нелинейных уравнений:')
    for i, group in enumerate(FUNCTIONS, 1):
        print(f'Система №{i}')
        print(group['disp'])
        print()
    n = int(input())
    f = FUNCTIONS[n - 1]['func']
```

```
    hint = FUNCTIONS[n-1]['init']
    jacob = FUNCTIONS[n - 1]['jacob']

    x0 = list(map(float, input(f'Начальные приближения ({" ".join(str(num)
for num in hint)}): ').split()))
    eps = float(input('Погрешность (0.001): '))

    res = system_newton_method(f, jacob, x0, eps)

    print(res)
    if output_file != None:
        with open(output_file, 'a') as fl:
            fl.write(f'Решение системы нелинейных уравнений:')
            fl.write(str(res))

def main():
    read_from_file = input('Нужно ли записать результат в файл (y/n): ')
    output_file = None
    if(read_from_file == 'y'):
        output_file = input('Введите название файла (out.txt): ')


    print('Решение нелейного уравнения: ')
    non_linear(output_file)
    print('Решение системы нелинейных уравнений: ')
    non_linear_system(output_file)


if __name__ == '__main__':
    main()
```

```
Solvers
from typing import Callable, Optional, List

import numpy as np
from attr import dataclass, field
from prettytable import PrettyTable


MAX_ITER_COUNT = 100


@dataclass
class Result:
    root: Optional[float] = None
    error: Optional[float] = None
    header: list = field(factory=list)
    data: list = field(factory=list)
    znach_f: float = field(default=0)
    iter_count: int = field(default=0)

    def __str__(self):
        tt = PrettyTable(self.header)
        data = [(n, *map(lambda x: f'{x:.3f}', floats)) for n, *floats in
self.data]
        tt.add_rows(data)
        return str(tt)


@dataclass
class SystemResult:
    iteration: int = field(default=0)
    solved: bool = field(default=False)
    roots: list = field(factory=list)
```

```python
    errors: list = field(factory=list)

    def __str__(self):
        if self.solved:
            return 'Решение: ' + ' '.join(f'{x:.3f}' for x in self.roots) +
'\n' + \
            'Погрешности: ' + ' '.join(str(x) for x in self.errors) + '\n'
+ \
            f'Количество итераций {self.iteration}\n'
        else:
            return 'Решений не найдено!'


def horde_method(f: Callable, left: float, right: float, fix=-1,
epsilon=10e-3):
    # [left, right] - интервал изоляции корня
    res = Result(header=['№', 'a', 'b', 'x', 'f(a)', 'f(b)', 'f(x)', '|a-
b|'])

    x0 = left if fix == -1 else right
    for i in range(1, MAX_ITER_COUNT + 1):
        res.iter_count = i
        x1 = (left * f(right) - right * f(left)) / (f(right) - f(left))
        res.data.append([i, left, right, x1, f(left), f(right), f(x1),
abs(left - right)])
        if abs(x1 - x0) <= epsilon or abs(f(x1)) <= epsilon:
            res.root = x1
            res.error = abs(x1 - x0)
            break
        if f(x1) * f(left) < 0:
            right = x1
        else:
            left = x1
        x0 = x1

    res.znach_f = f(res.root)
    return res


def newton_method(y: Callable, df: Callable, x0: float, epsilon=10e-3):
    res = Result(header=['№', 'x_k', 'f(x_k)', "f'(x_k)", 'x_{k+1}', '|x_k-
x_{k+1}|'])

    for i in range(1, MAX_ITER_COUNT + 1):
        res.iter_count = i
        x1 = x0 - y(x0) / df(x0)
        res.data.append([i, x0, y(x0), df(x0), x1, abs(x1 - x0)])

        if abs(x1 - x0) <= epsilon or abs(y(x1) / df(x1)) <= epsilon or
abs(y(x1)) <= epsilon:
            res.root = x1
            res.error = abs(x1 - x0)
            break
        x0 = x1
    res.znach_f = y(res.root)
    return res


def simple_iteration_method(f: Callable, phi: Callable, x0=1, epsilon=10e-
3):
    res = Result(header=['№', 'x_k', 'f(x_k)', 'x_{k+1}', 'phi(x_k)',
'|x_k-x_{k+1}|'])

    for i in range(1, MAX_ITER_COUNT + 1):
        res.iter_count = i
```

```python
        x1 = phi(x0)

        res.data.append([i, x0, f(x0), x1, phi(x0), abs(x1 - x0)])

        if abs(x1 - x0) <= epsilon:
            res.root = x1
            res.error = abs(x1 - x0)
            break
        x0 = x1
    res.znach_f = f(res.root)
    return res


def _newton_method(f, jack, x_init):
    jacobian = jack(*x_init)
    vector_b_f_output = f(*x_init)
    x_delta = np.linalg.solve(jacobian, vector_b_f_output)
    x_plus_1 = x_delta + x_init
    return x_plus_1


def system_newton_method(f, jack, x_init, epsilon):
    result = SystemResult()

    x_old = x_init
    x_new = _newton_method(f, jack, x_old)
    diff = np.linalg.norm(x_old-x_new)

    while diff > epsilon or result.iteration == MAX_ITER_COUNT:
        x_new = _newton_method(f, jack, x_old)
        diff = np.linalg.norm(x_old-x_new)
        x_old = x_new
        result.iteration += 1
    convergent_val = x_new

    if result.iteration != MAX_ITER_COUNT:
        result.solved = True

    result.roots = convergent_val
    return result
```

```python
Graph
from typing import Callable, List, Tuple

import numpy as np
from matplotlib import pyplot as plt


def show_2d(y: Callable, points: List[Tuple]):
    width = max(abs(points[0][0]), abs(points[len(points) - 1][0])) + 1
    height = abs(y(width))

    vf = np.vectorize(y)
    x = np.linspace(-width, width, 100)

    fig = plt.figure()
    ax = fig.add_subplot(1, 1, 1)

    plt.grid(True)
    plt.xlim((-width, width))
    plt.ylim((-height, height))

    ax.spines['left'].set_position('center')
    ax.spines['bottom'].set_position('center')
```

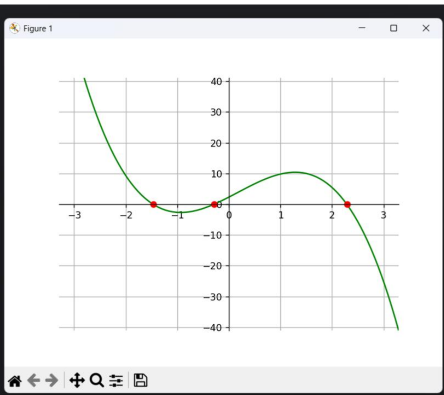
```python
    ax.spines['right'].set_color('none')
    ax.spines['top'].set_color('none')

    ax.plot(x, vf(x), 'g', label='y=f(x)')
    ax.plot(*list(zip(*points)), 'ro')

    plt.show()
```

Вывод кода:

```
Нужно ли записать результат в файл (y/n): n
Решение нелейного уравнения:
Границы левого корня через пробел (-2 -1): -2 -1
Границы для второго корня: (-1 0): -1 0
x0 для второго корня равен равен: 0.0
Границы для правого корня: (2 3): 2 3
dpfi в a1 = 0.3482832834571333
dpfi в b1 = 0.3007994987279191
Погрешность (0.01): 0.01
```




```
Левый корень x_0=-1.4676889, знач-е: -0.0326803, n: 6
Центральный корень x_1=-0.2861018, знач-е: 0.0011011, n: 2
Правый корень x_2=2.2916407, знач-е: -0.1271546, n: 5
Таблица для метода хорд:
+---+--------+--------+--------+-------+--------+--------+-------+
| № |   a    |   b    |   x    | f(a)  |  f(b)  |  f(x)  | |a-b| |
+---+--------+--------+--------+-------+--------+--------+-------+
| 1 | -2.000 | -1.000 | -1.221 | 9.330 | -2.650 | -1.964 | 1.000 |
| 2 | -2.000 | -1.221 | -1.357 | 9.330 | -1.964 | -1.068 | 0.779 |
| 3 | -2.000 | -1.357 | -1.423 | 9.330 | -1.068 | -0.486 | 0.643 |
| 4 | -2.000 | -1.423 | -1.451 | 9.330 | -0.486 | -0.203 | 0.577 |
| 5 | -2.000 | -1.451 | -1.463 | 9.330 | -0.203 | -0.082 | 0.549 |
| 6 | -2.000 | -1.463 | -1.468 | 9.330 | -0.082 | -0.033 | 0.537 |
+---+--------+--------+--------+-------+--------+--------+-------+
Таблица для метода Ньютана:
+---+--------+--------+---------+---------+---------------+
| № |  x_k   | f(x_k) | f'(x_k) | x_{k+1} | |x_k-x_{k+1}| |
+---+--------+--------+---------+---------+---------------+
| 1 | 0.000  | 2.310  |  8.630  | -0.268  |     0.268     |
| 2 | -0.268 | 0.137  |  7.434  | -0.286  |     0.018     |
+---+--------+--------+---------+---------+---------------+
Таблица для метода простой итерации:
+---+-------+---------+---------+----------+---------------+
| № |  x_k  |  f(x_k) | x_{k+1} | phi(x_k) | |x_k-x_{k+1}| |
+---+-------+---------+---------+----------+---------------+
| 1 | 3.000 | -25.170 |  2.546  |  2.546   |     0.454     |
| 2 | 2.546 |  -7.109 |  2.384  |  2.384   |     0.162     |
| 3 | 2.384 |  -2.418 |  2.323  |  2.323   |     0.061     |
| 4 | 2.323 |  -0.886 |  2.300  |  2.300   |     0.023     |
| 5 | 2.300 |  -0.334 |  2.292  |  2.292   |     0.009     |
+---+-------+---------+---------+----------+---------------+
Решение системы нелинейных уравнений:
```

```
Выберите систему нелинейных уравнений:
Система №1
f1(x1, x2) = x + 2*y - 2
f2(x1, x2) = x^2 + 4*y^2 - 4)

Система №2
f1(x1, x2, x3) = x + y + z - 3
f2(x1, x2, x3) = x^2 + y^2 + z^2 - 5
f3(x1, x2, x3) = exp(x) + x*y - x*z -1
```

Вывод: В ходе работы были рассмотрены численные методы решения нелинейных уравнений и систем: метод половинного деления, секущих, хорд и простой итерации. Каждый из методов позволяет находить корни с заданной точностью, используя разные подходы: деление интервала, касательные, секущие. Очень важным является выбор начального приближения или интервалов изоляции корней: чем выбор точнее, тем быстрее найдётся решение задачи. Методы обладают разной скоростью сходимости и требованиями к выбору начальных данных, что влияет на точность и эффективность решения.