Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
Высшего образования
«Национальный исследовательский университет ИТМО»

Факультет Программной Инженерии и Компьютерной Техники

Лабораторная работа 2 по вычислительной математике

Численное решение нелинейных уравнений и систем

Вариант №10

Преподаватель: Малышева Татьяна Алексеева

Выполнил: Состанов Тимур Айратович

Группа: P3214

Г. Санкт-Петербург

2024

Оглавление

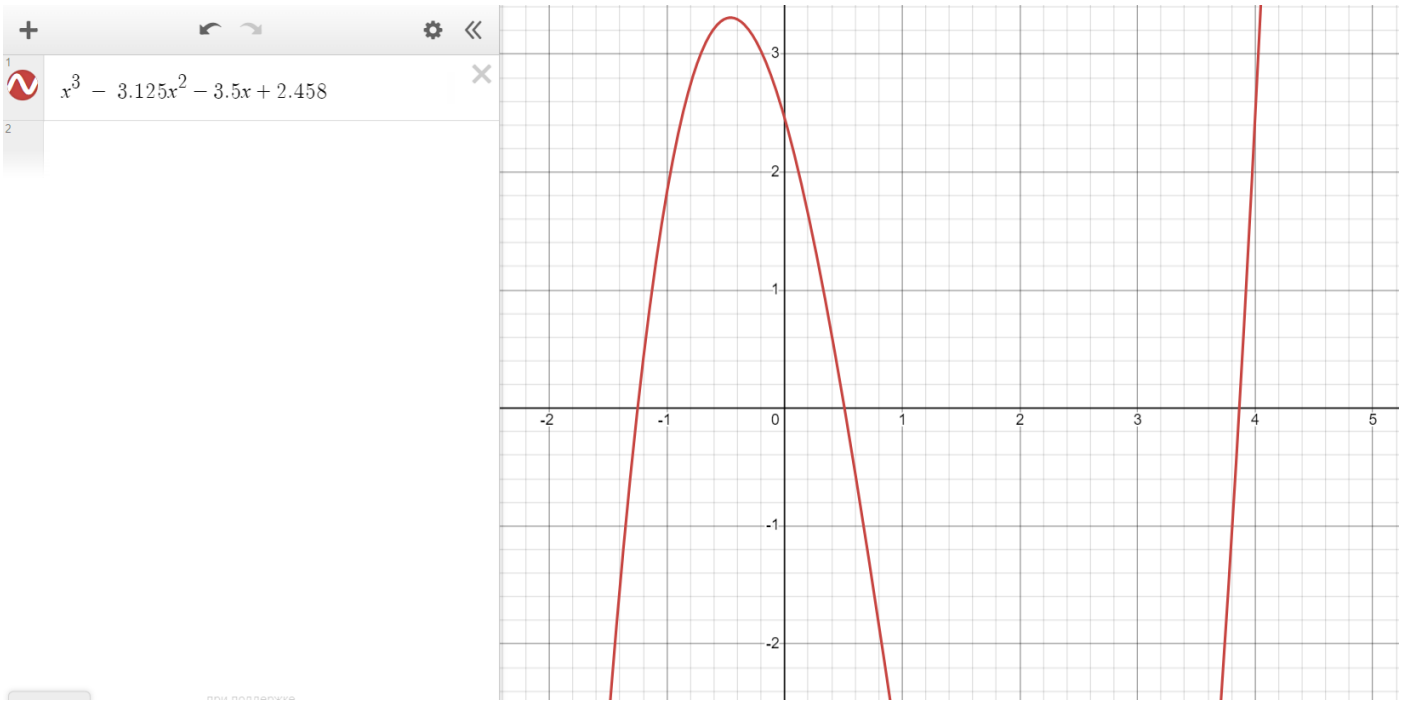
Цель работы

Изучить численные методы решения нелинейных уравнений и их систем, найти корни заданного нелинейного уравнения/системы нелинейных уравнений, выполнить программную реализацию методов.

Вычислительная реализация задачи

Решение нелинейного уравнения

$$x^3 - 3{,}125x^2 - 3{,}5x + 2{,}458$$



Интервалы изоляции корней:

1) [-2; -1]
2) [0;1]
3) [3;4]

Для уточнения значения корней воспользуемся:

1) Метод половинного деления
2) Метод простой итерации
3) Метод Ньютона

Уточняем крайний левый корень:

Метод половинного деления							
№ шага	a	b	x	f(a)	f(b)	f(x)	$\lvert b_n - a_n \rvert \leq \varepsilon$
0	-2,000	-1,000	-1,500	-11,042	1,833	-2,698	1
1	-1,500	-1,000	-1,250	-2,698	1,833	-0,003	0,5
2	-1,250	-1,000	-1,125	-0,003	1,833	1,017	0,25
3	-1,250	-1,125	-1,1875	-0,003	1,017	0,533	0,125
4	-1,250	-1,1875	-1,21875	-0,003	0,533	0,272	0,0625
5	-1,250	-1,21875	-1,234375	-0,003	0,272	0,136	0,03125
6	-1,250	-1,234375	-1,2421875	-0,003	0,136	0,067	0,015625
7	-1,250	-1,2421875	-1,24609375	-0,003	0,067	0,032	0,0078125

Уточняем средний корень:

$$f'(x) = 3x^2 - 6{,}25x - 3{,}5$$

$$f'(0) = -3{,}5$$

$$f'(1) = -6{,}75$$

$$f'[0,1] < 0, \text{поэтому}$$

$$\lambda = \frac{1}{6{,}75} = 0{,}148$$

$$x = x + 0{,}148(x^3 - 3{,}125x^2 - 3{,}5x + 2{,}458)$$

$$\varphi(x) = 0{,}148x^3 - 0{,}463x^2 + 0{,}482x + 0{,}364$$

Условие сходимости:

$$\varphi'(x) = 0{,}444x^2 - 0{,}926x + 0{,}482$$

$$\varphi'(0) = 0{,}482$$

$$\varphi'(1) = 0$$

Условие сходимости выполняется

Метод простой итерации				
№ шага	x	x_{k+1}	$f(x_k+1)$	$\lvert x_{k+1} - x_k]$
0	1,000	0,531	-0,132	0,469
1	0,531	0,512	-0,019	0,019
2	0,512	0,509	-0,0013	0,003

Уточняем крайний правый корень:

$$f'(x) = 3x^2 - 6{,}25x - 3{,}5$$

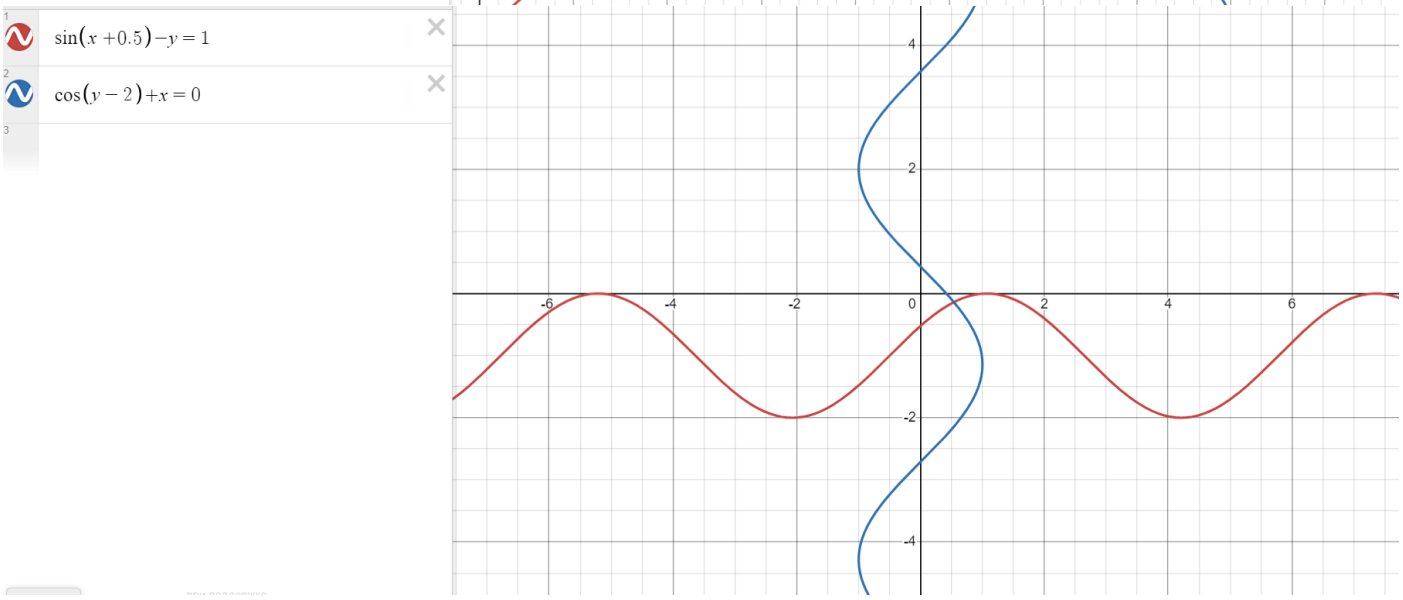
$$f''(x) = 6x - 6{,}25$$

$f(a_0)$	*-9,167*
$f(b_0)$	*2,458*
$f''(a_0)$	*11,75*
$f''(b_0)$	*17,75*

Метод Ньютона					
№ шага	x_k	$f(x_k)$	$f'(x_k)$	X_{k+1}	$\lvert x_{k+1} - x_k]$
1	4,000	2,458	19,5	3,874	0,126
2	3,874	0,140	17,311	3,866	0,008

Решение системы нелинейных уравнений

$$\begin{cases}\sin(x+0{,}5)-y=1\\ cos(y\ -\ 2)+x=0\end{cases}$$



Решение уравнений находится в области

$$\begin{cases}\ 0{,}5<x<1\\ -0{,}5<y<0\end{cases}$$

Выразим $\varphi(x)$

$$\begin{cases}\ \ x=-cos(y-2)\\ y=\sin(x+0{,}5)-1\end{cases}$$

Проверяем условие сходимости

$\frac{\partial\varphi_1}{\partial x}=0$ $\frac{\partial\varphi_1}{\partial y}=\sin(y-2)$

$\frac{\partial\varphi_2}{\partial x}=cos(x+0{,}5)$ $\frac{\partial\varphi_2}{\partial y}=0$

$|\sin(y-2)|<\sin(2)<1$

$|cos(x+0{,}5)|<\cos(1)<1$

$max(\sin(2)\,;\cos(1))\ =\sin(2)<1$ => *Процесс сходящийся*

Начальное приближение: $x = 0{,}5, y = -0{,}1$

Метод итераций						
№ шага	x_k	y_k	x_{k+1}	y_{k+1}	$\lvert x_{k+1} - x_k \rvert$	$\lvert y_{k+1} - y_k \rvert$
0	0,500	-0,100	0,506	-0,158	0,006	0,058
1	0,506	-0,158	0,554	-0,155	0,048	0,030
2	0,554	-0,155	0,552	-0,131	0,002	0,024
3	0,552	-0,131	0,531	-0,132	0,021	0,001
4	0,531	-0,132	0,532	-0,142	0,001	0,010
5	0,532	-0,142	0,541	-0,142	0,009	0,000

Программная реализация задачи

Метод Ньютона

```
x_i = x_0                                                    ⚠3 ⚠1 ✓29 ^
table = [["Итерация", "x_i", "f(x_i)", "f'(x_i)", "x_(i + 1)", "|x_(i + 1) - x_i|"]]

while True:
    if iterations >= max_iterations:
        print("Достигнуто максимальное количество итераций. Решение не найдено.")
        return None

    f_x_i = equation(x_i)
    f_prime_x_i = derivative(x_i)
    f_double_prime_x_i = second_derivative(x_i)

    x_i_plus_1 = x_i - (f_x_i / f_prime_x_i)
    difference = abs(x_i_plus_1 - x_i)

    table.append([iterations + 1, x_i, f_x_i, f_prime_x_i, x_i_plus_1, difference])

    if abs(f_x_i) <= epsilon or difference <= epsilon:
        print("Метод Ньютона завершен после", iterations + 1, "итераций.")
        print("Приближенное значение корня:", x_i_plus_1)
        print(tabulate(table, headers="firstrow", tablefmt="fancy_grid"))
        return x_i_plus_1

    if f_double_prime_x_i == 0:
        print("Производная второго порядка равна нулю. Метод Ньютона не применим.")
        return None

    x_i = x_i_plus_1
    iterations += 1
```

Метод хорд

```
    a = interval_start
    b = interval_end
    iterations = 0

    table = [["Итерация", "a", "b", "x_i", "f(a)", "f(b)", "f(x_i)"]]

    while True:
        if iterations >= max_iterations:
            print("Достигнуто максимальное количество итераций. Решение не найдено.")
            return None

        x_i = a - (b - a) * equation(a) / (equation(b) - equation(a))

        table.append([iterations+1, a, b, x_i, equation(a), equation(b), equation(x_i)])

        if abs(equation(x_i)) <= epsilon:
            print("Метод хорд завершен после", iterations+1, "итераций.")
            print("Приближенное значение корня:", x_i)
            print(tabulate(table, headers="firstrow", tablefmt="fancy_grid"))
            return x_i

        if np.sign(equation(a)) * np.sign(equation(x_i)) < 0:
            b = x_i
        else:
            a = x_i

        iterations += 1
```

Метод простых итераций

```
max_abs_derivative = max(abs(derivative(a)), abs(derivative(b)))
lmbda = 1 / max_abs_derivative * ((-1) if (derivative(a) >= 0 and derivative(b) >= 0) else 1)

 tsostanov
def phi(x):
    return x + lmbda * equation(x)

# Условие сходимости
max_phi_derivative = max(abs(derivative_by_definition(phi, a)), abs(derivative_by_definition(phi, b)))
if max_phi_derivative >= 1:
    # print("Метод простой итерации не гарантирует сходимость на данном интервале.")
    # return None
    pass

iteration_data = []

while True:
    if iterations > max_iterations:
        print("Достигнуто максимальное количество итераций. Решение не найдено.")
        return None

    f_x = equation(initial_guess)
    next_x = initial_guess + lmbda * f_x
```

```
iteration_data.append({
    "№ итерации": iterations,
    "x_i": initial_guess,
    "φ(x_i)": phi(initial_guess),
    "f(x_i)": f_x,
    "|x_i+1 - x_i|": abs(next_x - initial_guess)
})

if abs(next_x - initial_guess) <= epsilon:
    print("Метод завершен после", iterations, "итераций.")
    print("Приближенное значение корня:", next_x)

    headers = "keys"
    print(tabulate(iteration_data, headers=headers, tablefmt="fancy_grid"))
    return next_x

initial_guess = next_x
iterations += 1
```

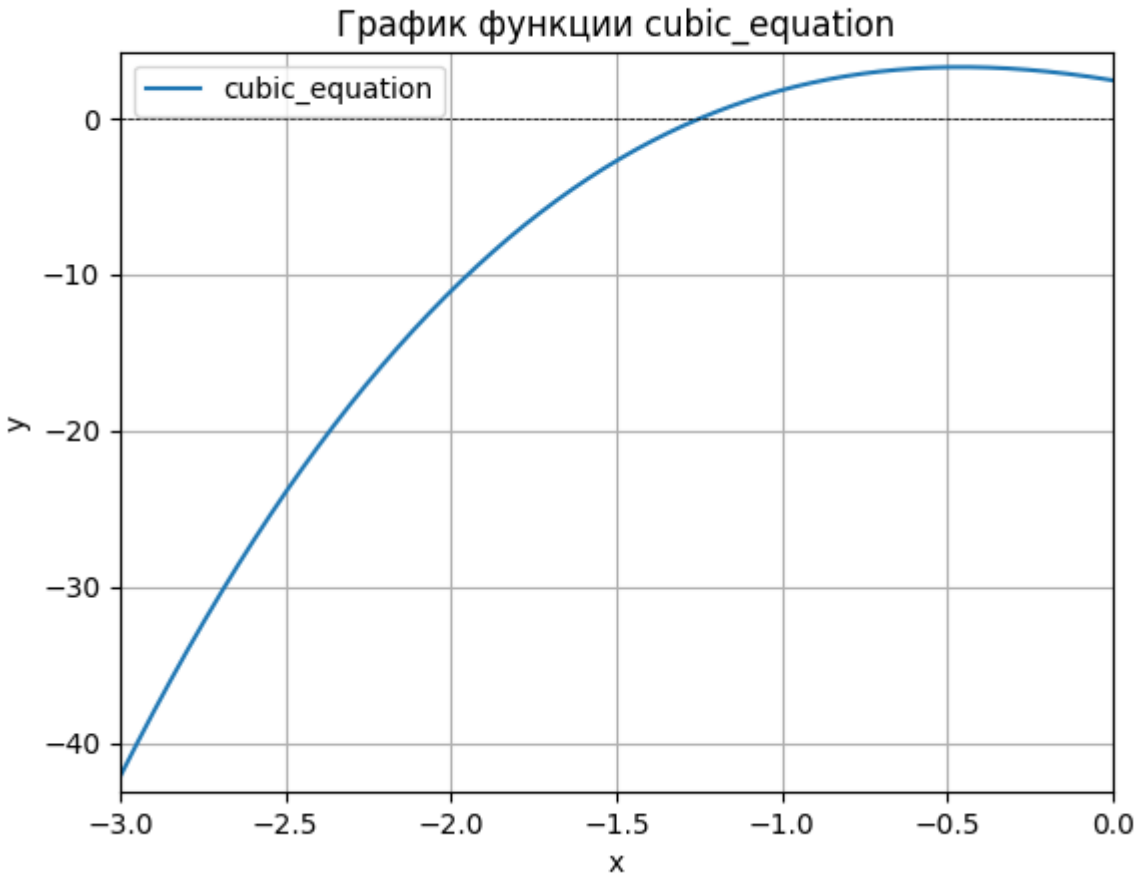
Метод Ньютона для систем нелинейных уравнений

```
2 usages
def newton_solver(equation_system, x_guess, y_guess):
    max_iterations = 100
    tolerance = 1e-6

    data = []

    for iteration in range(max_iterations):
        equations = equation_system(x_guess, y_guess)

        if equation_system.__name__ == "first_system":
            jacobi_matrix = np.array([
                first_jacobi_for_first(x_guess, y_guess),
                second_jacobi_for_first(x_guess, y_guess)
            ])
        else:
            jacobi_matrix = np.array([
                first_jacobi_for_second(x_guess, y_guess),
                second_jacobi_for_second(x_guess, y_guess)
            ])

        deltas = np.linalg.solve(jacobi_matrix, -np.array(equations))

        x_guess += deltas[0]
        y_guess += deltas[1]
```

```
        data.append([iteration, x_guess, y_guess, deltas[0], deltas[1]])

        if max(abs(i) for i in deltas) < tolerance:
            print("Решение найдено:")
            print("x =", x_guess)
            print("y =", y_guess)
            break

    else:
        print("Достигнуто максимальное количество итераций. Решение не найдено.")

    headers = ['Итерация', 'x', 'y', 'Δx', 'Δy']
    print(tabulate(data, headers=headers, tablefmt='pretty'))
```

Результат работы программы

Для нелинейных уравнений:

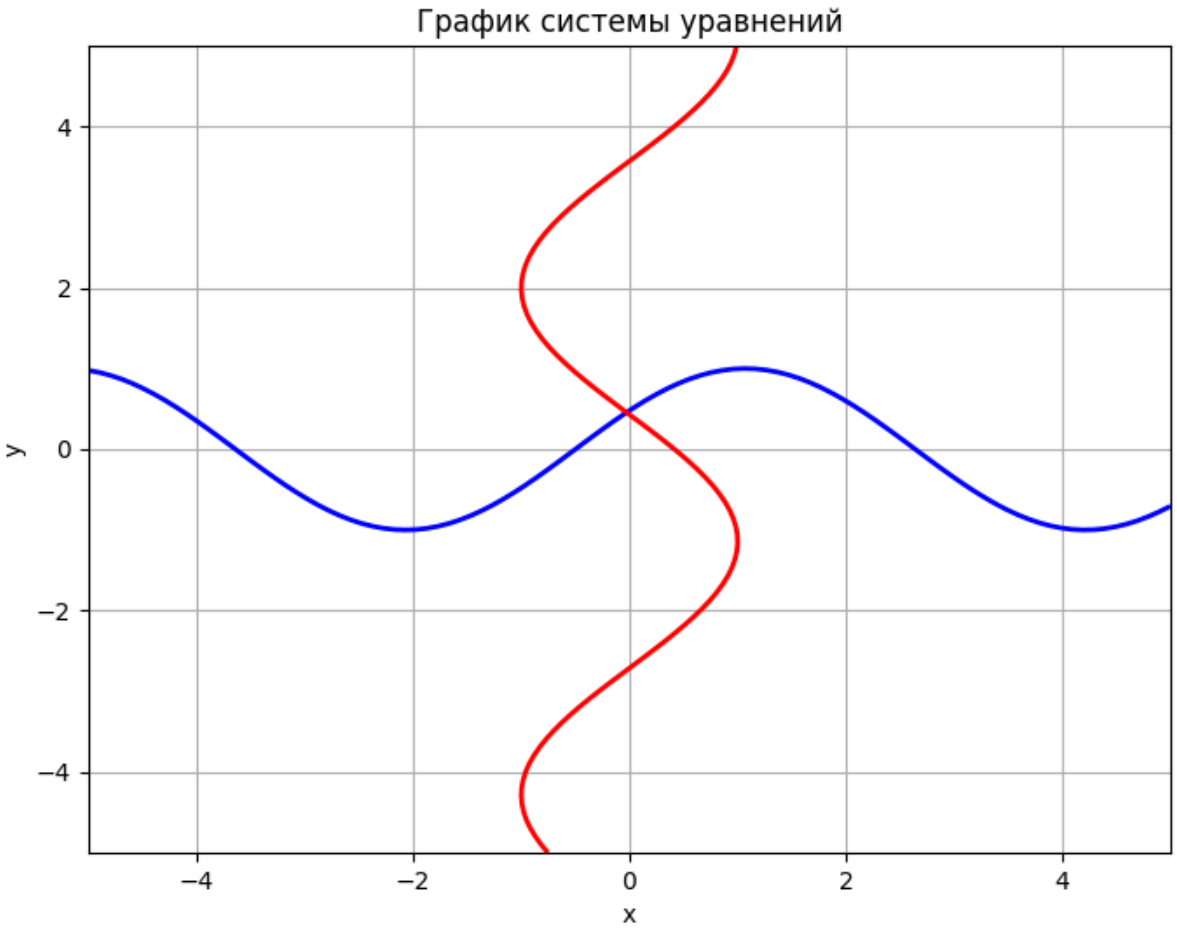



```
Выберите уравнение для решения:
1. Кубическое уравнение: y = x ** 3 - 3.125 * x ** 2 - 3.5 * x + 2.458
2. Уравнение с тригонометрическими функциями: y = cos(x ** 2) + sin(x)
3. Еще одно уравнение с тригонометрическими функциями: y = sin(x ** 2) + cos(x)
Введите номер уравнения (1, 2 или 3): 1
Выберите источник ввода данных:
1. Ввод с клавиатуры
2. Чтение данных из файла
Выберите источник данных (1 - ввод с клавиатуры, 2 - ввод из файла): 1
Введите начало интервала: -2
Введите конец интервала: -1
Введите начальное приближение к корню (ноль, если начальное приближение не требуется): -2
Введите погрешность вычисления: 0.001
Выбранное уравнение: cubic_equation
Интервал: -2.0 - -1.0
Начальное приближение к корню: -2.0
Погрешность вычисления: 0.001
Производная функции сохраняет знак на интервале, что гарантирует наличие только одного корня.
Выберите метод решения уравнения:
1. Метод Ньютона
2. Метод хорд
3. Метод простых итераций
Введите номер метода (1, 2 или 3): 3
Метод завершен после 9 итераций.
Приближенное значение корня: -1.250772372758154
```

№ итерации	x_i	φ(x_i)	f(x_i)	|x_i+1 - x_i|
0	-2	-1.47419	-11.042	0.52581
1	-1.47419	-1.36098	-2.37747	0.113213
2	-1.36098	-1.30918	-1.08778	0.0517989
3	-1.30918	-1.28252	-0.559832	0.0266587
4	-1.28252	-1.2681	-0.302925	0.014425
5	-1.2681	-1.26009	-0.168052	0.00800246
6	-1.26009	-1.25559	-0.0944738	0.00449875
7	-1.25559	-1.25305	-0.053499	0.00254757
8	-1.25305	-1.2516	-0.0304194	0.00144854
9	-1.2516	-1.25077	-0.0173362	0.000825534

Process finished with exit code 0

Для системы уравнений:




```
Выберите систему уравнений:
Первая система уравнений:
/sin(x + 0.5) - y = 0
\cos(y - 2) + x = 0

Вторая система уравнений:
/cos(x + 0.5) - y = 0
\sin(y - 2) + x = 0

Введите номер системы (1 или 2): 1
Первая система уравнений:
/sin(x + 0.5) - y = 0
\cos(y - 2) + x = 0

Выбранная система: first_system
Введите начальное приближение для переменной x: 1
Введите начальное приближение для переменной y: 1
Начальное приближение для x: 1.0
Начальное приближение для y: 1.0
Решение найдено:
x = -0.026657227832235595
y = 0.45586405918632056
```

Итерация	x	y	Δx	Δy
0	-0.4517796878904705	0.8948001540446715	-1.4517796878904705	-0.10519984595532855
1	-0.050564825581120354	0.44895012790288125	0.40121486230935016	-0.44585002614179026
2	-0.02672389900729022	0.455930216254222	0.023840926573830135	0.006980088351340736
3	-0.026657228399053694	0.4558640596948876	6.667060823652501e-05	-6.615655933439551e-05
4	-0.026657227832235595	0.45586405918632056	5.668180994383211e-10	-5.085670159279799e-10

Исходный код: https://github.com/tsostanov/CompMath2

https://github.com/tsostanov/CompMath2.2

Вывод

В ходе работы были изучены численные методы решения нелинейных уравнений и систем нелинейных уравнений. В результате работы были найдены корни заданных уравнений и систем с использованием различных численных методов, а также были построены графики функций и были написана программа для автоматического нахождения корней в заданной области