Санкт-Петербургский Государственный университет ИТМО
Факультет програмной инженерии и компьютерной техники

Лабораторная работа №2 по дисциплине "Вычислительная математика"

Численное решение нелинейных уравнений и систем

Вариант №7

Работу выполнила:
Д. А. Карасева
Группа: P3217
Преподаватель:
Т. А. Малышева

Санкт-Петербург
2024

Содержание

1. Цель работы

Изучить численные методы решения нелинейных уравнений и их систем, найти корни заданного нелинейного уравнения/системы нелинейных уравнений, выполнить программную реализацию *метода половинного деления, метода секущих, метода простой итерации и метода Ньютона.*

Метод должен быть реализован в виде отдельной подпрограммы/метода/класса. Пользователь выбирает уравнение, корень/корни которого требуется вычислить (3-5 функций, в том числе и трансцендентные), из тех, которые предлагает программа. Предусмотреть ввод исходных данных (границы интервала/начальное приближение к корню и погрешность вычисления) из файла или с клавиатуры по выбору конечного пользователя. Выполнить верификацию исходных данных. Необходимо анализировать наличие корня на введенном интервале. Если на интервале несколько корней или они отсутствуют – выдавать соответствующее сообщение. Программа должна реагировать на некорректные введенные данные. Для методов, требующих начальное приближение к корню (методы Ньютона, секущих, хорд с фиксированным концом, простой итерации), выбор начального приближения (a или b) вычислять в программе. Для метода простой итерации проверять достаточное условие сходимости метода на введенном интервале. Предусмотреть вывод результатов (найденный корень уравнения, значение функции в корне, число итераций) в файл или на экран по выбору конечного пользователя. Организовать вывод графика функции, график должен полностью отображать весь исследуемый интервал (с запасом).

Для *систем нелинейных уравнений* должно быть реализовано:

- Пользователь выбирает предлагаемые программой системы двух нелинейных уравнений (2-3 системы).
- Организовать вывод графика функций.
- Начальные приближения ввести с клавиатуры.
- Для метода простой итерации проверить достаточное условие сходимости.
- Вывод вектора неизвестных: $x_1, x_2, ..., x_n$.
- Вывод количества итераций, за которое было найдено решение.
- Вывод вектора погрешностей: $|x_i^{(k)} - x_i^{(k-1)}|$

2. Описание методов. Расчетные формулы

Метод половинного деления

Наиболее примитивным, и в то же время и надежным алгоритмом определения вещественного корня является метод половинного деления (или дихотомия, бисекция, метод взятия в вилку). Пусть известно, что непрерывная на отрезке *[a, b]* функция *f(x)* в точках a и b имеет разные знаки, то есть *f(a) · f(b) < 0*. Тогда, как известно из курса математического анализа, на этом отрезке имеется хотя бы один корень.

Идея метода: отрезок, где находится хотя бы один корень, делится на два равных отрезка и из них выбирается тот, на концах которого функция опять имеет разные знаки. Далее, с полученным отрезком опять проделываем ту же операцию. Так как длина отрезка каждый раз уменьшается в два раза, то после n шагов получим отрезок длины $\frac{b-a}{2^n}$, который содержит корень x_*. Понятно, что какова бы ни была длина исходного отрезка, для $\forall\epsilon > 0$ через конечное число шагов получим отрезок длины меньше , который содержит корень. Следовательно, любая точка ε этого последнего отрезка удовлетворяет неравенству $|\ \varepsilon - x_*| < \epsilon$, то есть является корнем уравнения с точностью $\epsilon > 0$. К простому корню метод приводит для любой непрерывной, в том числе и недифференцируемой функции. К сожалению, метод имеет медленную скорость сходимости, но точность ответа всегда можно гарантировать.

Метод секущих

Пусть, как и в методе половинного деления, имеем отрезок *[a, b]* , где $f(a) \cdot f(b) < 0$. Для определенности будем полагать, что $f(a) < 0$, $f(b) > 0$ и $f''(x) > 0$ на всем отрезке. Тогда итерационный процесс для метода секущих имеет вид:

$$x_{n+1} = x_n - \frac{f(x_n}{f(b) - f(x_n)}(b - x_n), x_0 = a$$

Для данного случая задания итерационного процесса точка b является неподвижной точкой.

Случай $f''(x) < 0$ сводится к рассматриваемому, если уравнение записать в виде $-f(x) = 0$. Тогда итерационный процесс примет вид:

$$x_{n+1} = x_n - \frac{f(x_n}{f(x_n) - f(a)}(x_n - a), x_0 = b$$

Здесь точка a является неподвижным концом отрезка *[a, b]*.

Теорема. Пусть на отрезке [a, b] уравнение $f(x) = 0$ имеет единственный корень и $f''(x)$ сохраняет знак на [a, b]. Тогда метод секущих сходится.

Метод простых итераций

Пусть с точностью ϵ необходимо найти корень уравнения $f(x) = 0$, принадлежащий интервалу изоляции *[a, b]*. Функция *f(x)* и ее первая производная непрерывны на этом отрезке. Для применения метода итераций (метода последовательных приближений) исходное уравнение *f(x) = 0* должно быть приведено к виду

$$x = \phi(x)$$

.

В качестве начального приближения выбираем любую точку интервала *[a, b]*. Далее итерационный процесс поиска корня строится по схеме:

$$x_{n+1} = \phi(x_n), n = 0, 1, ...$$

Процесс поиска прекращается, как только выполняется условие $|x_{n+1} - x_n| < \epsilon$ или число итераций превысит заданное число *N*. При определенных условиях на функцию $\phi(x)$ итерационная по- следовательность $x_{n+1} = \phi(x_n), n = 0, 1, ...$ сходится к корню уравнения $x = \phi(x)$.

Метод Ньютона

Основная идея этого метода состоит в выделении из уравнений системы линейных частей, которые являются главными при малых приращениях аргументов. Это позволяет свести исходную задачу к решению последовательности систем линейных уравнений. Данный метод является обобщением метода касательных. Существенную роль в этом методе играет специальная матрица – матрица Якоби

$$\mathbf{J_{i,j}} = \begin{bmatrix} \frac{\partial u_1}{\partial x_1} & \frac{\partial u_1}{\partial x_2} & \frac{\partial u_1}{\partial x_3} \\ \frac{\partial u_2}{\partial x_1} & \frac{\partial u_2}{\partial x_2} & \frac{\partial u_2}{\partial x_3} \\ \frac{\partial u_3}{\partial x_1} & \frac{\partial u_3}{\partial x_2} & \frac{\partial u_3}{\partial x_3} \end{bmatrix}$$

Очевидно, что построить её можно лишь при условии, что каждая из функций, входящая в систему дифференцируема по каждой из переменных. Напомним, что метод касательных применительно к одному уравнению *f(x) = 0* заключается в построении итерационной последовательности:

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

Обобщением этой формулы на системы уравнений является следующая формула:

$$X^{k+1} = X^k - J^{-1}(X^k) * f(X^k)$$

О сходимости метода Ньютона для систем уравнений можно, сказать то же, что и о сходимости метода касательных для одного уравнения: если начальное приближение выбрано достаточно близко к решению системы, то итерационная последовательность сходится к этому решению, и сходимость является квадратичной. Метод Ньютона весьма трудоемок, поскольку на каждом шаге итерационного процесса необходимо найти матрицу, обратную матрице Якоби.

3. Вычислительная часть

Часть 1. Решение нелинейного уравнения

$$f(x) = x^3 + 2.28x^2 - 1.934x - 3.907$$

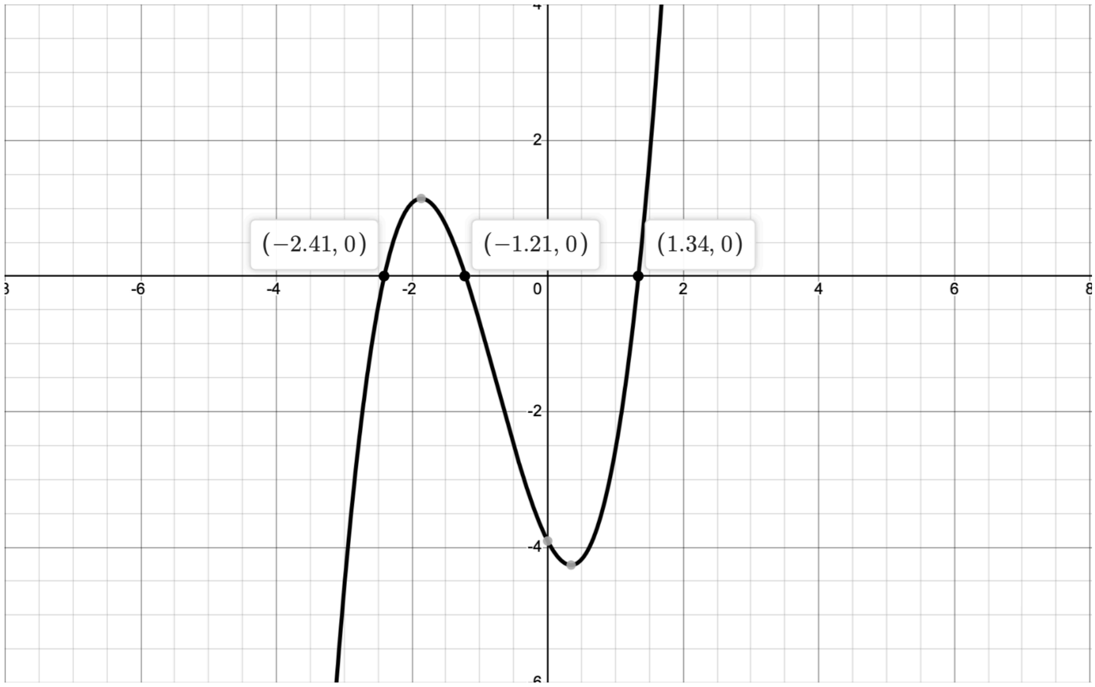

Рисунок 3.1

Определим интервалы изоляции корней.

Прибближенные значения корней $x \sim -2.41, x \sim -1.21, x \sim 1.34$ Исследуем функцию $f(x) = x^3+2.28x^2-1.934x-3.907$. Продифференцируем функцию $f'(x) = 3x^2+4.56x-1.934$ и приравняем полученный результат к 0. $y' = 0 \rightarrow \left[\begin{array}{l} x \sim -1.86 \\ x \sim 0.34 \end{array}\right.$

$$f(-1.86) = (-1.86)^3 + 2.28 * (-1.86)^2 - 1.934 * (-1.86) - 3.907 = 1.14$$

$$f(0.34) = (0.34)^3 + 2.28 * (0.34)^2 - 1.934 * (0.34) - 3.907 = -4.34$$

При $x \in (-\infty; -1.86) \cup (0.34; +\infty) f(x)' > 0$, следовательно, f(x) монотонно возрастает. При $x \in (-1.86; 0.34) f'(x) < 0$, следовательно, y монотонно убывает. (-1.86; 1.14) - точка максимума, (0.34; -4.34) - точка минимума

Повтороно продифференцируем функцию $f''(x) = 6x+4.56$ и приравняем полученный результат к 0. $y'' = 0 \rightarrow x \sim= -0.76$

$$f(-0.76) = (-0.76)^3 + 2.28 * (-0.76)^2 - 1.934 * (-0.76) - 3.907 = -1.55$$

При $x \in (-\infty; -0.76) f''(x) < 0$, следовательно, график функции выпуклый. При $x \in (-0.76; \infty) f''(x) > 0$, следовательно, график функции вогнутый. (-0.76; -1.55) - точка перегиба.

Составим таблицу знакопостоянства функции и сравним значения в разных точках.

x	-3	-2	-1	0	1	2
y	-	+	-	-	-	+

Получаем три интервала изоляции корней: (-3; -2), (-2; -1), (1; 2)

Крайний правый корень - метод половинного деления

№ шага	a	b	x	f(a)	f(b)	f(x)	|a-b|
1	-3	-2	-2,5	-4,585	1,081	-0,447	1
2	-2,5	-2	-2,25	-0,447	1,081	0,596375	0,5
3	-2,5	-2,25	-2,375	-0,447	0,596375	0,150390625	0,25
4	-2,5	-2,375	-2,4375	-0,447	0,150390625	-0,128646484	0,125
5	-2,4375	-2,375	-2,40625	-0,128646484	0,150390625	0,015695068	0,0625
6	-2,4375	-2,40625	-2,421875	-0,128646484	0,015695068	-0,055258514	0,03125
7	-2,4375	-2,421875	-2,4296875	-0,128646484	-0,055258514	-0,09164677	0,015625

Крайний левый корень - метод простой итерации

Проверка условия сходимости на выбранном интервале:

$$f(x) = x^3 + 2.28x^2 - 1.934x - 3.907$$

$$f'(x) = 3x^2 + 4.56x - 1.934$$

$$f'(a) = 3 * 1^2 + 4.56 * 1 - 1.934 = 5.626 > 0, f'(b) = 3 * 2^2 + 4.56 * 2 - 1.934 = 19.188 > 0$$

$$max(|f'(a)|, |f'(b)|) = 19.188 \longrightarrow \lambda = \frac{1}{max(|f'(x)|)} = \frac{1}{19.188}$$

$$\phi(x) = x + \lambda * f(x) = x + \frac{x^3 + 2.28x^2 - 1.934x - 3.907}{19.188}$$

$$\phi'(x) = 1 + \lambda * f'(x) = 1 + \frac{3x^2 + 4.56x - 1.934}{19.188}$$

$|\phi'(a)| = 1.999$
$|\phi'(b)| = 1.29$
$|\phi'(x)| >= 1$, итерационная последовательность не сходится к корню ни при каком x из данного отрезка.

Центральный корень - метод Ньютона

№ шага	x_k	$f(x_k)$	$f'(x_k)$	x_{k+1}	$\|x_{k+1} - x_k\|$
1	-1	-0,693	-3,494	-1,198340011	0,198340011
2	-1,198340011	-0,036126357	-3,090374103	-1,210029974	0,011689963
3	-1,210029974	-0,000181302	-3,059219068	-1,210089238	5,92641E-05

$$\begin{cases} 2x - sin(y - 0.5) = 1 \\ y + cos(x) = 1.5 \end{cases}$$

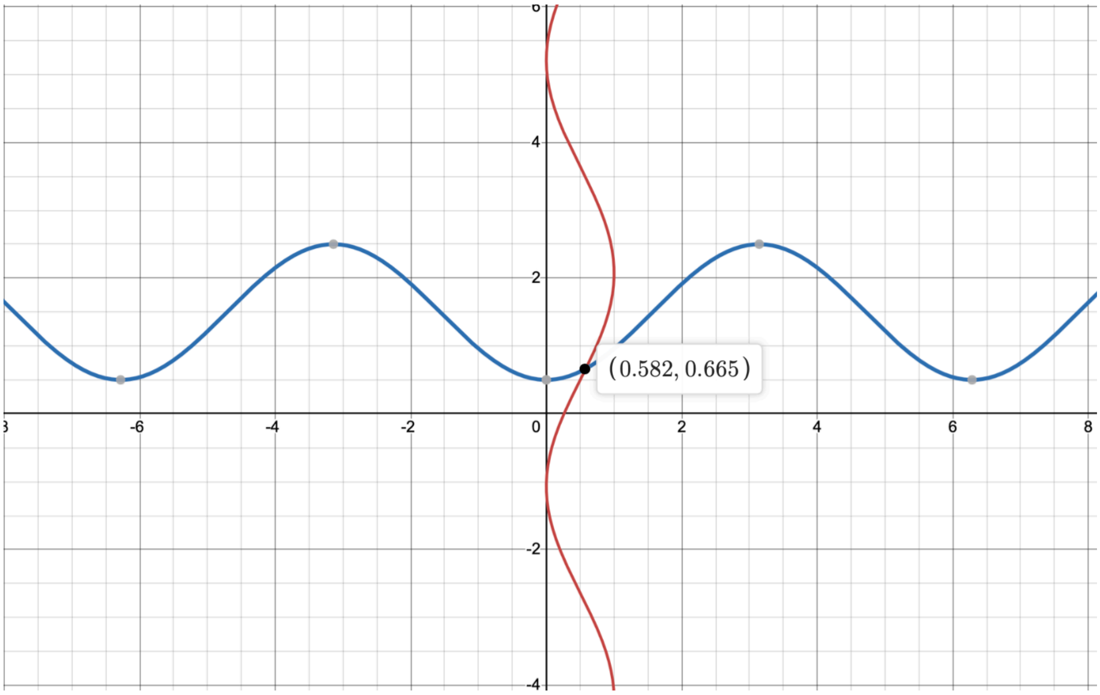

Рисунок 3.2

Метод простой итерации

$$\begin{cases} 2x - sin(y - 0.5) = 1 \\ y + cos(x) = 1.5 \end{cases} \rightarrow \begin{cases} f(x,y) = 0 \\ g(x,y) = 0 \end{cases} \rightarrow \begin{cases} 2x - sin(y - 0.5) - 1 = 0 \\ y + cos(x) - 1.5 = 0 \end{cases}$$

Проверим условие сходимости:

$$\frac{\partial f}{\partial x} = -1, \frac{\partial f}{\partial y} = cos(y - 0.5)$$

$$\frac{\partial g}{\partial x} = sin(x), \frac{\partial g}{\partial y} = 0$$

$$|\frac{\partial f}{\partial x}| + |\frac{\partial f}{\partial y}| = |-1| + |cos(y - 0.5)| <= 2$$

$$|\frac{\partial g}{\partial x}| + |\frac{\partial g}{\partial y}| = |sin(x)| + |0| <= 1$$

$\max|\phi'(x)| > 1 \rightarrow$ процесс не является сходящимся

$$x_{k+1} = \frac{1 + sin(y_k - 0.5)}{2}, y_{k+1} = 1.5 - cos(x_k)$$

№ шага	x_k	y_k	x_{k+1}	y_{k+1}	$\lvert x_{k+1} - x_k \rvert$	$\lvert y_{k+1} - y_k \rvert$
1	0	0	0,260287231	0,5	0,260287231	0,5
2	0,260287231	0,5	0,5	0,533683903	0,239712769	0,033683903
3	0,5	0,533683903	0,516838767	0,622417438	0,016838767	0,088733535
4	0,516838767	0,622417438	0,561055954	0,630614405	0,044217187	0,008196967
5	0,561055954	0,630614405	0,565121669	,65330627	0,004065715	0,022691864
6	0,565121669	0,65330627	0,576353227	0,65547655	0,011231557	0,00217028
7	0,576353227	0,65547655	0,577425459	0,661544398	0,001072232	0,006067848
8	0,577425459	0,661544398	0,580421344	0,662129214	0,002995885	0,000584816
9	0,580421344	0,662129214	0,580709931	0,663768331	0,000288587	0,001639117

После 8й итерации получаем $x \sim 0.58, y \sim 0.66$

4. Листинг программы

Репозиторий на GitHub

Функция, отвечающая за реализацию метода половинного деления

```
def bisection_method(func, a, b, epsilon=1e-6, max_iterations=10,
derive_f=lambda x: np):
    iterations = 0
    c = (a + b) / 2
    while abs(func(c)) > epsilon:
        c = (a + b) / 2
        if abs(func(c)) <= epsilon:
            break
        elif func(c) * func(a) < 0:
            b = c
        else:
            a = c
        iterations += 1
    return c, iterations, func(c)
```

Функция, отвечающая за реализацию метода секущих

```
def secant_method(func, a, b, epsilon=1e-6, max_iter=100,
derive_f=lambda x: np):
    xi = a
    xii = b
    iterations = 0

    while abs(xi - xii) > epsilon and iterations < max_iter:
        x_next = xi - func(xi) * (xii - xi) / (func(xii) - func(xi))
        xi, xii = xii, x_next
        iterations += 1
        if abs(func(x_next)) <= epsilon:
            break
    return x_next, iterations, func(x_next)
```

Функция, отвечающая за реализацию метода простых итераций

```
def simple_iterations_method(func, a, b, epsilon=1e-6,
max_iter=1000, derive_f=lambda x: np):
    iter_count = 0
    x0 = b
    if derive_f((a + b) / 2) < 0:
        L = 1 / max(abs(derive_f(a)), abs(derive_f(b)))
    else:
        L = -1 / max(abs(derive_f(a)), abs(derive_f(b)))
    while iter_count < max_iter:
        x1 = x0 + L * func(x0)
        # print(iter_count, x1, func(x1))
        if abs(x1 - x0) < epsilon:
            return x1, iter_count, func(x1)
        x0 = x1
```

```
        iter_count += 1
    print("The solution does not converge after {}
    iterations".format(max_iter))
    return x0, iter_count, func(x0)
```

Функция, отвечающая за реализацию метода Ньютона

```
def newton_method(f, x0, epsilon=1e-6, max_iter=100):
    x = x0
    iterations = 0
    errors = []
    for _ in range(max_iter):
        J = jacobian(f, x)
        delta_x = np.linalg.solve(J, -f(x))
        x = x + delta_x
        errors.append(np.linalg.norm(delta_x))
        iterations += 1
        if (abs(delta_x[0])< epsilon) and (abs(delta_x[1])< epsilon):
            break
    return x, iterations, errors
```

5. Примеры и результаты работы программы

Для нелинейного уравнения

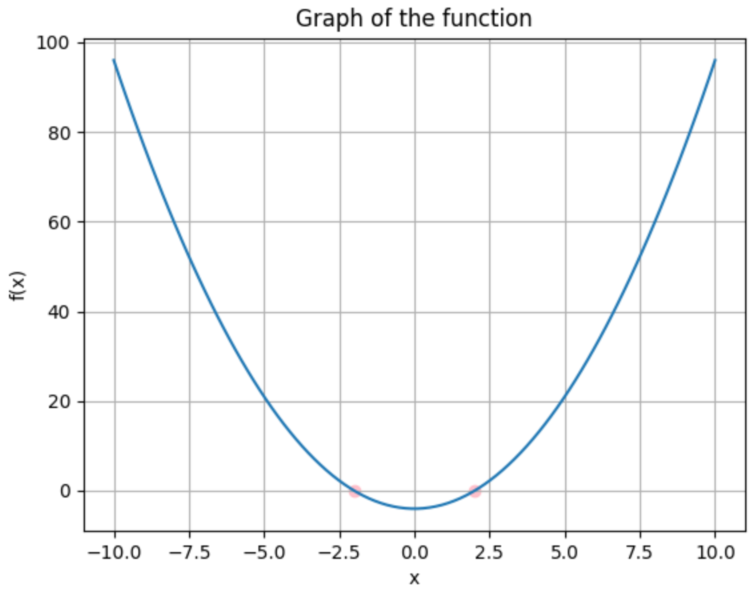

Рисунок 5.1

Решить систему или нелинейное уравнение? 1-Нелинейное уравнение 2-Система нелинейных уравнений - 1

Выберите функцию для вычисления:

1. $x^2 - 4$
2. sin(x)
3. exp(x) - 3
4. $x^3 - 5x - 9$
5. cos(x) - x

Введите номер выбранной функции: 1
Введите нижнюю границу интервала: -5
Введите верхнюю границу интервала: 5
Введите точность: 0.1

На выбранном интервале [-5.0; 5.0] 2 корней

Предлагаем разбить ваш интервал на отрезки следующими точками и последовательно их исследовать.

Рассматриваемый отрезок [-5.0; -0.005005005005005003]

Условие сходимости метода простой итерации на выбранном интервале выполнено

Метод: Половинного деления
Root: -1.9951983233233233
Iterations: 6
Value at the root: -0.0192
Accuracy: 0.1

Метод: Секущих
Root: -1.9957388643707576
Iterations: 6
Value at the root: -0.017
Accuracy: 0.1

Метод: Простой итерации
Root: -1.8414665347313155
Iterations: 6
Value at the root: -0.609
Accuracy: 0.1

Рассматриваемый отрезок [-0.005005005005005003; 5.0]
Условие сходимости метода простой итерации на выбранном интервале не выполнено

Метод: Половинного деления
Root: 1.9891766766766767
Iterations: 6
Value at the root: -0.0432
Accuracy: 0.1

Метод: Секущих
Root: 1.9953376222723
Iterations: 5
Value at the root: -0.0186
Accuracy: 0.1

Метод: Простой итерации
Root: 2.085542983851963
Iterations: 4
Value at the root: 0.3495
Accuracy: 0.1

Для системы нелинейных уравнений

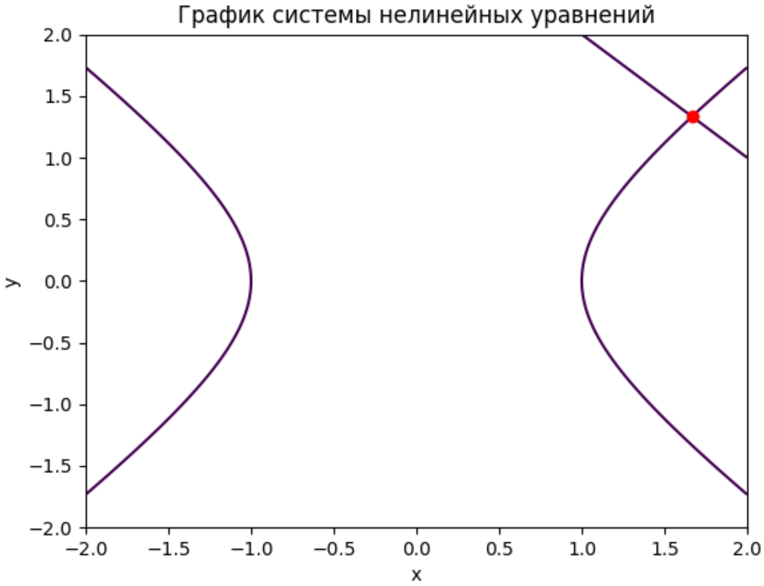

Рисунок 5.2

Решить систему или нелинейное уравнение? 1-Нелинейное уравнение 2-Система нелинейных уравнений - 2

Выберите систему для вычисления:

1. $x^2 - y^2 - 1 = 0; x + y - 3 = 0$
2. $x^2 + y^2 - 4 = 0; 3x^2 - y = 0$

Введите номер выбранной системы 1

Введите начальные приближения для x: 0.5

Введите начальные приближения для y: 1

Решение системы: [1.66666667 1.33333333]

Количество итераций: 3

6. Вывод

В результате выполнения лабораторной работы я изучила численные методы решения нелинейных уравнений и систем нелинейных уравнений, смогла программно реализовать некоторые из методов на языке Python.