Учреждение образования
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»
Кафедра информатики

Отчет по лабораторной работе №4
Решение систем нелинейных уравнений

Выполнил:
студент группы 953501
Кременевский В.С

Руководитель:
доцент
Анисимов В.Я.

Минск 2021

Содержание

1. Цель работы

1) Изучить методы численного решения систем нелинейных уравнений (метод простой итерации, метод Ньютона)
2) Составить программу численного решения нелинейных уравнений методами простой итерации и Ньютона
3) Проверить правильность работы программы на тестовых примерах
4) Численно решить нелинейное уравнение заданного варианта
5) Сравнить число итераций, необходимого для достижения заданной точности вычисления разными методами

2. Теоретические сведения

Краткие теоретические сведения. Пусть дана система нелинейных уравнений (система не линейна, если хотя бы одно из входящих в нее уравнений не линейно):

$$\begin{cases} f_1(x_1,\ldots,x_n)=0 \\ f_2(x_1,\ldots,x_n)=0 \\ \ldots\ldots\ldots\ldots\ldots \\ f_n(x_1,\ldots,x_n)=0 \end{cases}$$

Мы можем записать систему в более компактной векторной форме

$$\boldsymbol{f(x)=0}, \qquad (4.1)$$

где $\boldsymbol{f} = (f, \ldots, f)$, $\boldsymbol{x}=(x, \ldots, x)^T$.

Для решения системы (4.1) иногда можно применить метод последовательного исключения неизвестных, который приводит решение системы к решению одного уравнения с одним неизвестным. Однако в подавляющем большинстве случаев систему уравнений (4.1) решают итерационными методами. Для решения системы (4.1) существует набор методов, из которых рассмотрим простейшие методы: метод простых итераций, базирующийся на принципе сжимающих отображений и метод Ньютона (многомерный аналог метода касательных).

Метод простых итераций. Чтобы воспользоваться методом простых итераций, необходимо предварительно привести систему к следующему виду:

$$\begin{cases} x_1 = \varphi_1(x_1,...,x_n) \\ x_2 = \varphi_2(x_1,...,x_n) \\ .....\\ x_n = \varphi_{n1}(x_1,...,x_n) \end{cases}$$

Или в векторной форме

$$\bar{x} = \varphi(\bar{x}) \qquad (4.2)$$

где $\vec{\varphi} = (\varphi_1, \varphi_2, ..., \varphi_n).$

Исходя из некоторого начального приближения $\bar{x}^0$, построим итерационную последовательность точек

$\bar{x}^k = \varphi(\bar{x}^{k-1})$, $k=1,2,....$

Пусть точка $\bar{x}^0$ есть некоторое начальное приближение к решению. Рассмотрим δ-окрестность $U_\delta(\bar{x})$ этой точки. В силу принципа сжимающих отображений итерационная последовательность

$\bar{x}^k = \varphi(\bar{x}^{k-1})$, $k=1,2,....$

будет сходиться, если существует число q<1 такое, что

выполнено условие:

$$\left\| \varphi(\bar{x}^1) - \varphi(\bar{x}^2) \right\| \le q \left\| \bar{x}^1 - \bar{x}^2 \right\|, \quad \forall \bar{x}^1, \bar{x}^2 \in U_\delta(\bar{x}^0), \qquad (4.3)$$

называемое условием сжатия для отображения φ. В частности, это условие всегда выполняется, если векторная функция φ непрерывно дифференцируема и норма матрицы производных функции φ удовлетворяет неравенству:

$$\left\| \frac{\partial \varphi}{\partial x}(\bar{x}) \right\| \le q$$

во всех точках *x* из δ-окрестности $U_\delta(\bar{x})$ точки $\bar{x}^0$.

Следующая теорема дает условие сходимости и оценку скорости сходимости метода простых итераций.

Теорема. Пусть отображение φ является сжатием в $U_\delta(\overline{x^0})$ *и пусть*

$$\left\|\varphi(\bar{x}^0) - \bar{x}^0\right\| < \delta(1-q).$$

Тогда итерационная последовательность:

$$\bar{x}^k = \varphi(\bar{x}^{k-1}),$$

с начальной точкой $\bar{x}^0$*, сходится к решению* $\bar{x}^*$ *системы (1). При этом справедлива следующая оценка погрешности:*

$$\left\|\bar{x}^k - \bar{x}^*\right\| \le \frac{q^k}{1-q}\left\|\varphi(\bar{x}^0) - \bar{x}^0\right\|.$$

Отметим, что начальное приближение $\bar{x}^0$ выбирают экспериментально. (Например, на основе грубого графического решения системы, если порядок системы не высок. По точкам строят график первого уравнения, потом второго и ищут приблизительно точку их пересечения).

Метод Ньютона.

Пусть дана система нелинейных уравнений :

$$\begin{cases} f_1(x_1,\ldots,x_n) = 0 \\ f_2(x_1,\ldots,x_n) = 0 \\ \ldots\ldots\ldots\ldots\ldots \\ f_n(x_1,\ldots,x_n) = 0 \end{cases}$$

Запишем ее в векторной форме:

$$f(\bar{x}) = 0 \qquad (4.1)$$

Найдем начальное приближение $\bar{x}^0$.

Будем предполагать, что векторная функция f непрерывна дифференцируема в некоторой окрестности начального приближения. Вместо системы (4.1) будем искать решение соответствующей ей линеаризованной системы

$$f(\bar{x}^0) + \frac{\partial f}{\partial x}(\bar{x}^0)(\bar{x} - \bar{x}^0) = 0 \Rightarrow \quad f(\bar{x}^0) + J(\bar{x}^0)(\bar{x} - \bar{x}^0) = 0,$$

где через $J(\bar{x}^0)$ обозначена для удобства записи матрица производных векторной функции f в точке $\bar{x}^0$ (матрица Якоби системы (4.1) в этой точке).

При этом при применении метода Ньютона предполагается, что $\det J(\bar{x}^0) \neq 0$ в окрестности точки $\bar{x}^0$.

Тогда из линеаризованной системы, которая линейна относительно переменных x, можно найти первое приближение

$$\bar{x}^k = \bar{x}^{k-1} - J^{-1}(\bar{x}^{k-1}) f(\bar{x}^{k-1}), k = 1,...$$

Рассматривая линеаризованную систему в точках при $k=1,2,....$, найдем к-ое приближение

$$\bar{x}^k = \bar{x}^{k-1} - J^{-1}(\bar{x}^{k-1}) f(\bar{x}^{k-1}), k = 1,...$$

Построенная таким способом рекуррентная последовательность Ньютона сходится при определенных дополнительных условиях к решению системы (4.1). Легко видеть, что рассматриваемый метод совпадает с методом касательных в случае $n=1$, т.е. является многомерным вариантом метода касательных.

На практике обратную матрицу не считают, а на каждом шаге решают линеаризованную систему:

$$f(\bar{x}^{k-1}) + J(\bar{x}^{k-1})(\bar{x} - \bar{x}^{k-1}) = 0 \Rightarrow \bar{x} = \bar{x}^k$$

При сделанных выше предположениях, последовательнсоть Нбютона сходится к решению системы если начальное приближение выбрано достаточно близко к решению.

Отметим в заключениие, что метод Ньютона сходится достаточно быстро, если начальное приближение выбрано удачно, то скорость сходимости почти всегда будет иметь квадратичный характер.

Теорема. При сделанных выше предположениях, последовательность Ньютона сходится к решению системы (4.1), если начальное приближение выбрано достаточно близко к решению.

Отметим в заключение, что метод Ньютона сходится достаточно быстро (скорость сходимости квадратичная), если начальное приближение выбрано удачно. На практике итерационный процесс заканчивают, когда норма разности двух последовательных приближений меньше заданной точности вычисления решения.

Еще пару слов о сходимости метода простых итераций.

Теорема 3.13 (о достаточном условии сходимости метода простых итераций). *Пусть функции* $\varphi_i(x)$ *и* $\varphi_i'(x),\ i = 1, \ldots, n$, *непрерывны в области* G, *причем выполнено неравенство (где* q *— некоторая постоянная)*

$$\max_{x \in G} \max_{i} \sum_{j=1}^{n} \left| \frac{\partial \varphi_i(x)}{\partial x_j} \right| \leqslant q < 1, \tag{3.28}$$

Если последовательные приближения $x^{(k+1)} = \Phi(x^{(k)}),\ k = 0, 1, 2, \cdots$ *не выходят из области* G*, то процесс последовательных приближений сходится:* $x_* = \lim\limits_{k \to \infty} x^{(k)}$ *и вектор* x_* *является в области* G *единственным решением системы*

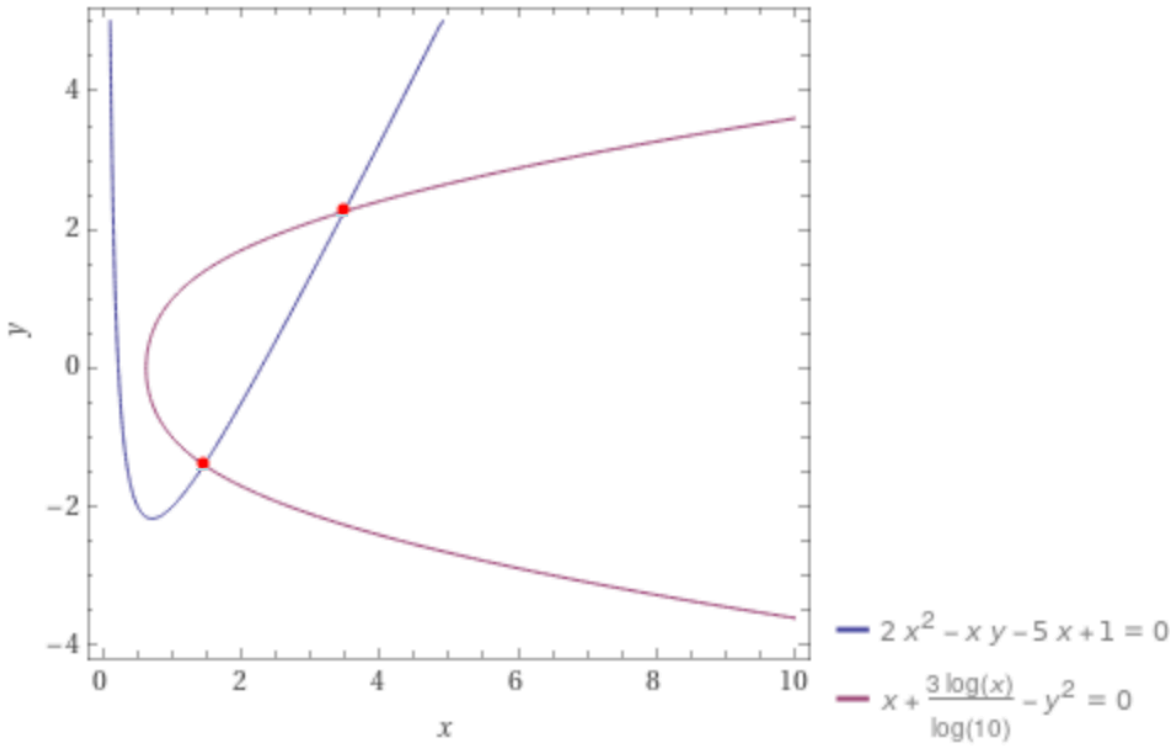
3. Тестовые примеры

Тестовый пример 1

Решить систему нелинейных уравнений с точностью до 0,0001 методами простых итераций и Ньютона:

$$\begin{cases} 2x_1^2 - x_1 x_2 - 5x_1 + 1 = 0, \\ x_1 + 3\lg x_1 - x_2^2 = 0. \end{cases}$$

Построим график функции, чтобы найти начальное приближение корня:



Из графика видно, что начальное приближение для верхнего корня можно взять x = 3.5, y = 2.2

Проверим выполнение условий сходимости. Будем рассматривать окрестность найденной точки $x^{(0)}$:

$$G = \{|x_1 - 3{,}5| \leqslant 0{,}1;\ |x_2 - 2{,}2| \leqslant 0{,}1\}.$$

Тогда

$$\left|\frac{\partial \varphi_1}{\partial x_1}\right| \leqslant \frac{2{,}3 + 5}{4\sqrt{\frac{3{,}4 \cdot (2{,}1 + 5) - 1}{2}}} = 0{,}536 < 0{,}54; \qquad \left|\frac{\partial \varphi_1}{\partial x_2}\right| \leqslant \frac{3{,}6}{4\sqrt{\frac{3{,}4 \cdot (2{,}1 + 5) - 1}{2}}} = 0{,}265 < 0{,}27;$$

$$\left|\frac{\partial \varphi_2}{\partial x_1}\right| \leqslant \frac{1 + \frac{3 \cdot 0{,}43429}{3{,}4}}{2\sqrt{3{,}4 + 3\lg 3{,}4}} = 0{,}309 < 0{,}31; \qquad \left|\frac{\partial \varphi_2}{\partial x_2}\right| = 0.$$

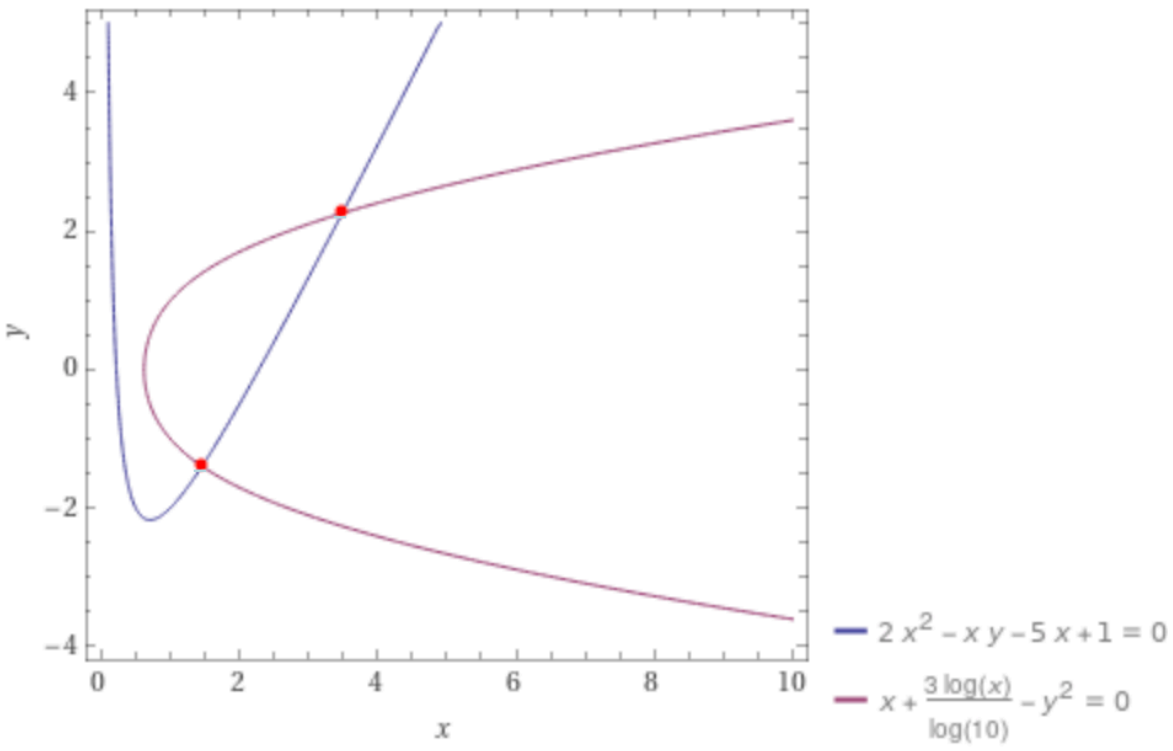
Решим систему для заданного начального приближения:

```
        Начальное приближение: (3.5, 2.2)
        *** Метод простой итерации: ***
Корни уравнения: ['3.4873', '2.2616']
Количество итераций: 9
--------------------
--------------------
        Начальное приближение: (3.5, 2.2)
        *** Метод Ньютона: ***
Корни уравнения: ['3.4874', '2.2616']
Количество итераций: 3
```

Тестовый пример 2

Решим ту же систему но с приблизительным корнем в другом интервале

$$\begin{cases} 2x_1^2 - x_1 x_2 - 5x_1 + 1 = 0, \\ x_1 + 3\lg x_1 - x_2^2 = 0. \end{cases}$$



Второй корень как видно примерно имеет координаты:

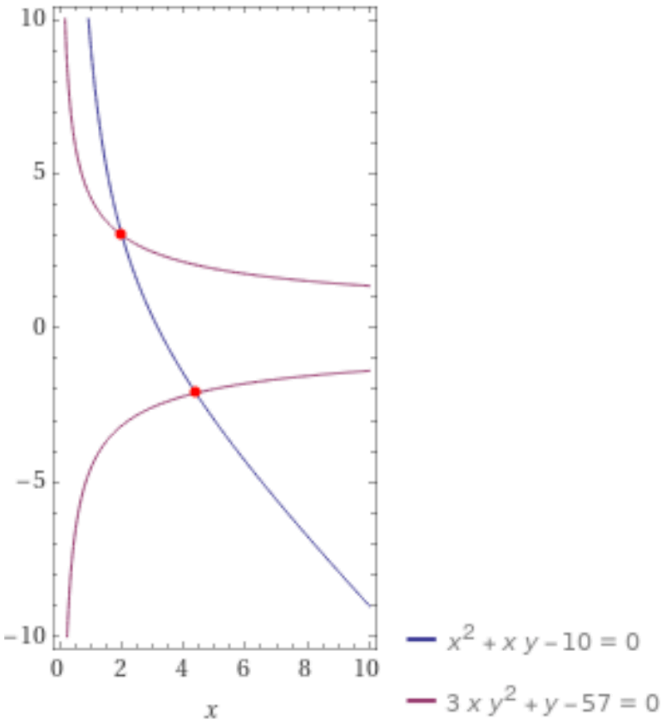
x = 1.8, y = -1.7

```
        Начальное приближение: (1.8, -1.7)
        *** Метод простой итерации: ***
Корни уравнения: ['3.4873', '2.2616']
Количество итераций: 21
--------------------
--------------------
        Начальное приближение: (1.8, -1.7)
        *** Метод Ньютона: ***
Корни уравнения: ['1.4589', '-1.3968']
Количество итераций: 4
```

Как видно метод простой итерации дал не тот ответ. Это связано с тем, что сжимающее отображение вокруг к которому идет метод простой итерации стягивается именно к [3.4873, 2.2616].

Тестовый пример 3

Решить систему нелинейных уравнений с точностью до 0,0001 методами простых итераций и Ньютона:



$$\begin{cases} x^2 + x * y - 10 = 0 \\ y + 3x * y^2 - 57 = 0 \end{cases}$$

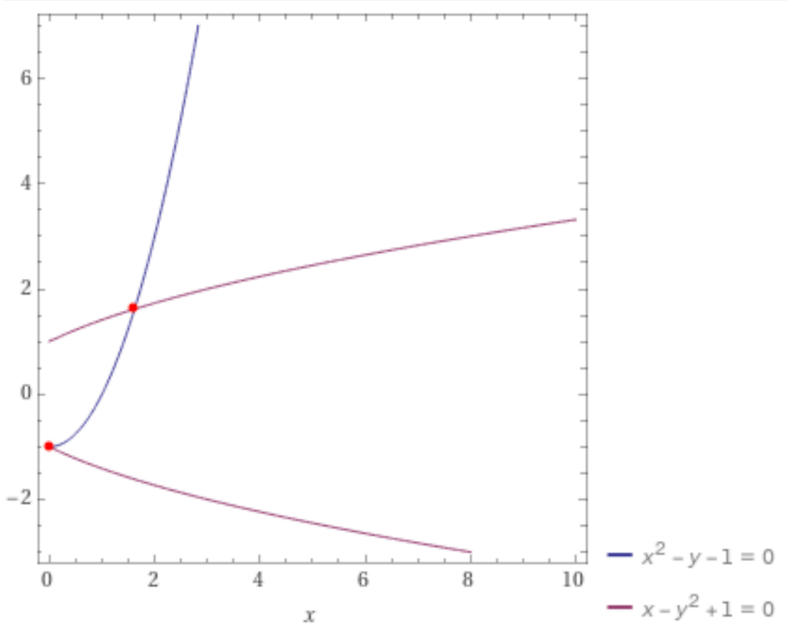
Из графика видно, что начальное приближение для верхнего корня можно взять x = 1.5, y = 3.5

```
        Начальное приближение: (1.5, 3.5)
        *** Метод простой итерации: ***
Корни уравнения: ['2.0000', '3.0000']
Количество итераций: 10
--------------------
--------------------
        Начальное приближение: (1.5, 3.5)
        *** Метод Ньютона: ***
Корни уравнения: ['2.0000', '3.0000']
Количество итераций: 4
```

Тестовый пример 4

Решить систему нелинейных уравнений с точностью до 0,0001 методами простых итераций и Ньютона:

$$\begin{cases} x^2 - y - 1 = 0 \\ x - y^2 + 1 = 0 \end{cases}$$



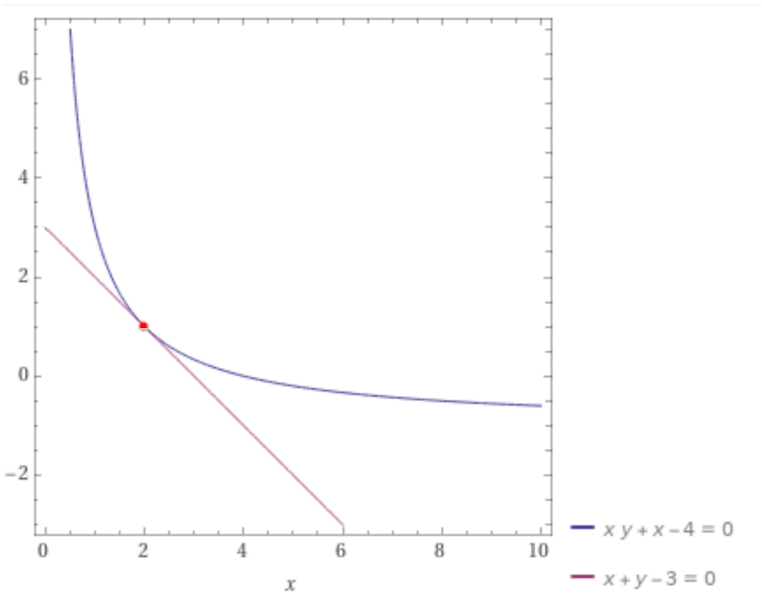
Из графика видно, что начальное приближение для верхнего корня можно взять x = 1.9, y = 1.8

```
        Начальное приближение: (1.9, 1.8)
        *** Метод простой итерации: ***
Корни уравнения: ['1.6180', '1.6180']
Количество итераций: 5
--------------------
--------------------
        Начальное приближение: (1.9, 1.8)
        *** Метод Ньютона: ***
Корни уравнения: ['1.6180', '1.6180']
Количество итераций: 4
```

Тестовый пример 5

Решить систему нелинейных уравнений с точностью до 0,0001 методам Ньютона:

$$\begin{cases} x + x * y - 4 = 0 \\ x + y - 3 = 0 \end{cases}$$



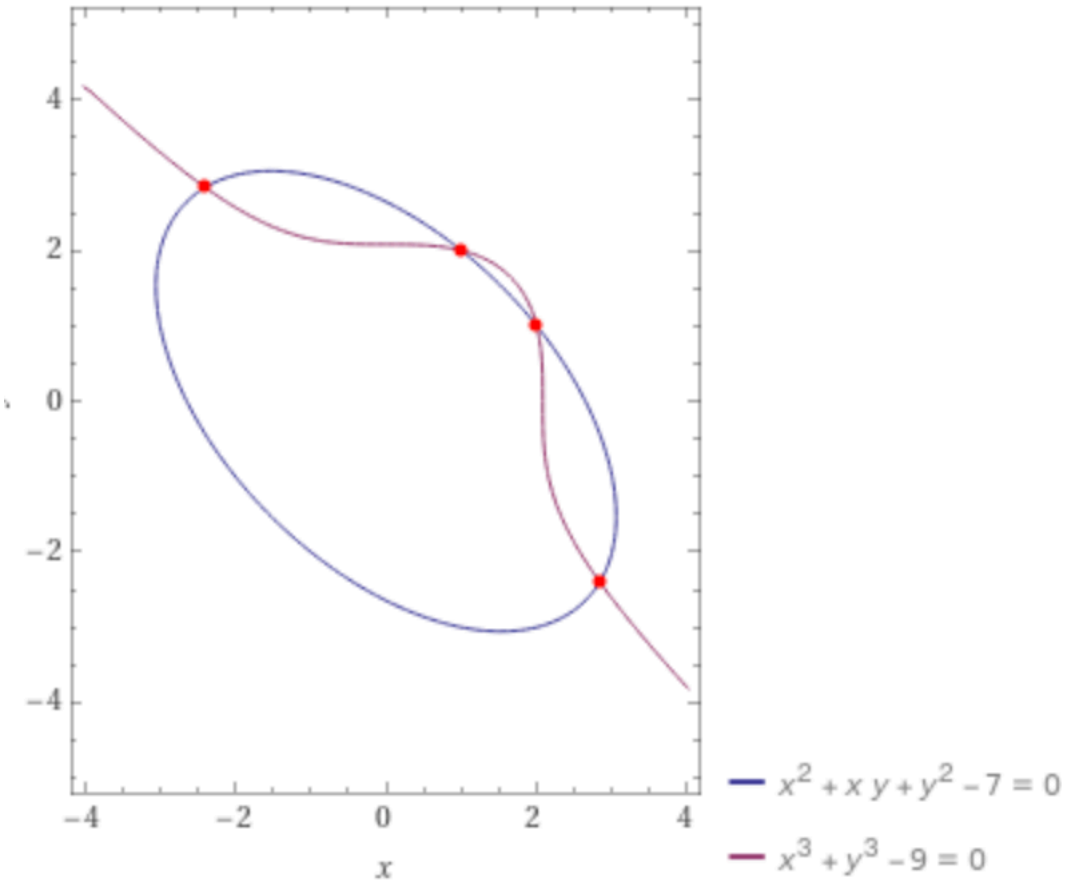
Из графика видно, что начальное приближение для верхнего корня можно взять x = 1.9, y = 0.6

```
        Начальное приближение: (1.9, 0.6)
        *** Метод Ньютона: ***
Корни уравнения: ['1.9999', '1.0001']
Количество итераций: 12
```

Тестовый пример 6

Решить систему нелинейных уравнений с точностью до 0,0001 методам Ньютона:

$$\begin{cases} x^2 + x * y + y^2 - 7 = 0 \\ x^3 + y^3 - 9 = 0 \end{cases}$$



```
        Начальное приближение: (1.5, 0.9)
        *** Метод Ньютона: ***
Корни уравнения: ['2.0000', '1.0000']
Количество итераций: 5
```

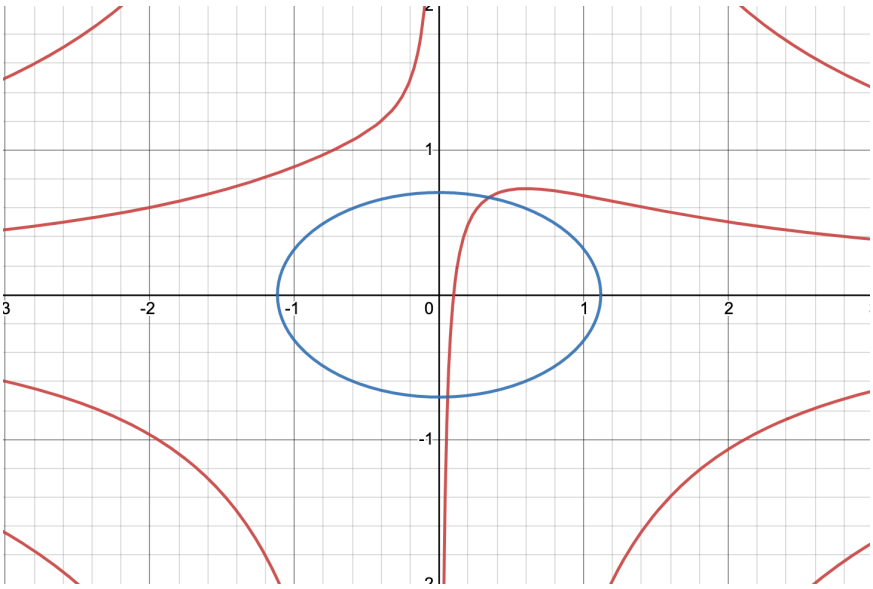
4. **Решение задания**
Вариант 10

Решить систему нелинейных уравнений с точностью до 0,0001 методами простых итераций и Ньютона:

m = 0.1

a = 0.8

$$\begin{cases} tg(xy + 0.1) = x \\ 0.8x^2 + 2y^2 = 1 \end{cases}$$



Из графика видно, что начальное приближение можно взять:
x = 0.3
y = 0.7

```
        Начальное приближение: (0.3, 0.7)
        *** Метод простой итерации: ***
Корни уравнения: ['0.3443', '0.6727']
Количество итераций: 14
--------------------
--------------------
        Начальное приближение: (0.3, 0.7)
        *** Метод Ньютона: ***
Корни уравнения: ['0.3444', '0.6727']
Количество итераций: 3
```

Найдем методом Ньютона второй корень, выбрав начальное приближение:
x = 0.05
y = -0.7

```
        Начальное приближение: (2.3, 0.7)
        *** Метод Ньютона: ***
Корни уравнения: ['2.0001', '0.9999']
Количество итераций: 12
```

5. Выводы

Таким образом, в ходе выполнения лабораторной работы были изучены методы численного решения систем нелинейных уравнений (метод простой итерации, метод Ньютона), составлена программа численного решения нелинейных уравнений методами простой итерации и Ньютона, проверена правильность работы программы на тестовых примерах, численно решено нелинейное уравнение заданного варианта, сравнено число итераций, необходимого для достижения заданной точности вычисления разными методами.

Как можно заметить, метод Ньютона более практичен, так как в решенных системах уравнений быстрее сходился к корню с заданной точностью, а также не требует нахождения сжимающего отображения, которого может быть найдено почти только вручную.

6. Программная реализация

Метод Ньютона

```python
def newton_solve(system_equations: np.array, approx, tol=0.0001):
    n = system_equations.shape[0]
    x = symbols(f'x:{n}')

    J = get_jacobi(system_equations)

    error = tol * 10000
    iteration = 0
    roots = approx
    while error > tol:
        iteration += 1

        roots_d = roots_to_dict(roots, x)
        jacobi_values = np.zeros(shape=(n, n))
        for i in range(n):
            for j in range(n):
                jacobi_values[i, j] = J[i, j].subs(roots_d)

        F = np.zeros(shape=(n, ))
        for i in range(0, n):
            F[i] = system_equations[i].subs(roots_d)

        delta_x = np.zeros(shape=(n, ), dtype=float)
        delta_x = np.linalg.solve(jacobi_values, -1 * F)

        roots = delta_x + roots
        error = np.amax(abs(delta_x))

    return roots, iteration
```

Метод простой итерации

```
def iteration_solve(system_equations: np.array, approx, tol=0.0001):
    n = system_equations.shape[0]
    x = symbols(f'x:{n}')
    fi_equations = system_equations[1]
    prev_roots = np.zeros(shape=(n, ))
    curr_roots = list(approx)
    errors = np.zeros(shape=(n, ))
    error = tol * 10000
    J = get_jacobi(system_equations[0])
    jacabi_values = np.zeros(shape=(n, n))
    roots_d = roots_to_dict(curr_roots, x)
    for i in range(n):
        for j in range(n):
            jacabi_values[i, j] = J[i, j].subs(roots_d)
    iteration = 0
    while error > tol:
        prev_roots = curr_roots.copy()
        roots_d = roots_to_dict(curr_roots, x)
        for i in range(n):
            try:
                curr_roots[i] = float(fi_equations[i].subs(roots_d))
            except TypeError:
                print("some complex numbers")
            errors[i] = abs(prev_roots[i] - curr_roots[i])
            roots_d = roots_to_dict(curr_roots, x)
        error = np.amax(errors)
        iteration += 1
    return curr_roots, iteration
```

Функции также использованные в работе

```python
def get_jacobi(system_equations: np.array):

    n = system_equations.shape[0]
    x = symbols(f'x:{n}')
    J = np.empty(shape=(n, n), dtype=core.add.Add)
    for i in range(n):
        for j in range(n):
            J[i, j] = system_equations[i].diff(x[j])

    return J

def check_norm(matrix):
    n = matrix.shape[0]
    max_value_norm = 0
    for i in range(n):
        sum = 0
        for j in range(r
            sum += abs(matrix[i, j])
        if sum > max_value_norm:
            max_value_norm = sum
    return max_value_norm

def roots_to_dict(roots, x):
    dict_roots = dict()
    for i in range(len(roots)):
        dict_roots[x[i]] = roots[i]

    return dict_roots
```

class int