Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей
Кафедра информатики
Дисциплина: Методы численного анализа

ОТЧЁТ

к лабораторной работе

на тему

Решение систем нелинейных уравнений

Выполнил: студент группы 153502
Богданов Алесандр Сергеевич

Проверил: Анисимов Владимир Яковлевич

Минск 2022

Содержание

Вариант 3

Цели выполнения задания

1. Изучить методы численного решения систем нелинейных уравнений (метод простых итераций, метод Ньютона)
2. Составить программу численного решения нелинейных уравнений методами простых итераций и Ньютона
3. Составить алгоритм решения нелинейных уравнений методами простых итераций и Ньютона
4. Проверить правильность работы программы на тестовых примерах
5. Численно решить нелинейное уравнение заданного варианта
6. Сравнить число итераций, необходимого для достижения заданной точности вычисления разными методами
7. Сравнить скорость вычисления систем нелинейных уравнений разными методами
8. Напечать отчёт
9. Принести отчёт на сдачу

Краткие теоритические 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:

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

Или в векторной форме

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

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

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

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

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

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

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:

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

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.

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 $x^{(k+1)} = \Phi(x^{(k)})$, $k = 0, 1, 2, \ldots$ *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*

функции $\varphi(x)$.

Теорема1. Пусть φ – непрерывно дифференцируемая векторная функция, для которой выполняются следующие условия:

1) $\left\|\varphi(x^0) - x^0\right\| \le \delta(1-q)$;

2) $\left\|\frac{\partial \varphi}{\partial x}(x)\right\| \le q, \qquad (q<1), \qquad \forall x \in U_\delta(x^0)$.

Тогда система (6.7) имеет решение $x^* \in U_\delta(x^0)$, причем единственное, и итерационная последовательность $x^k = \varphi(x^{k-1}), \quad k = 1,2,\ldots$, с начальным приближением x^0 сходится к решению x^* системы (6.7) . При этом справедлива следующая оценка скорости сходимости:

$$\left\|x^k - x^*\right\| \le \frac{q^n}{1-q}\left\|x^1 - x^0\right\|.$$

Доказательство. Из условия (6.7) вытекает сжимаемость этого отображения. Проверим, что

$\varphi : U_\delta(x^0) \to U_\delta(x^0)$.

Разложив $\varphi(x)$ по формуле Тейлора, получаем для $\forall x \in U_\delta(x^0)$

$$\varphi(x) = \varphi(x^0) + \frac{\partial \varphi}{\partial x}(\xi)(x - x^0),$$

где $\xi \in U_\delta(x^0)$. Тогда

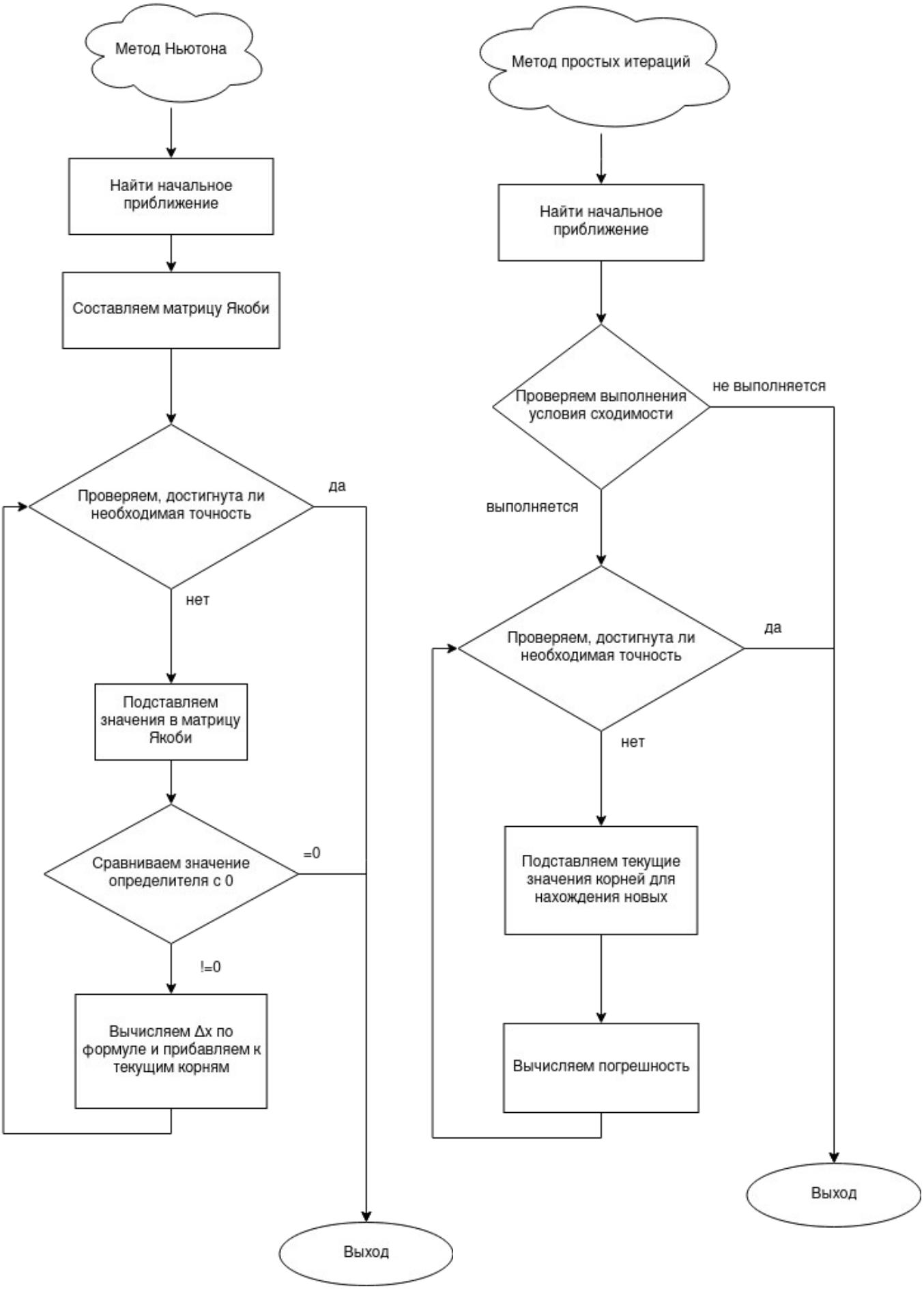
$$\left\|\varphi(x) - x^0\right\| = \left\|\varphi(x^0) + \frac{\partial \varphi}{\partial x}(\xi)(x - x^0) - x^0\right\| \le$$

$$\le \left\|\varphi(x^0) - x^0\right\| + \left\|\frac{\partial \varphi}{\partial x}(\xi)\right\| \cdot \left\|x - x^0\right\| < \delta(1-q) + q\delta = \delta,$$

т. е. $\varphi(x) \in U_\delta(x^0)$.

Следовательно, мы можем применить принцип сжимающих отображений, в силу которого получаем результат теоремы. При этом из принципа сжимающих отображений следует, что

Алгоритм программы



Программная реализация.

Метод простых итераций

```
def get_q(fi_equation, approx, e=0.1):
    x_0 = approx[0]
    x_fi = approx[1]
    x = symbols('x:2')
    e = 0.1
    x_1 = random.uniform(x_0 - e, x_0 + e)
    x_2 = random.uniform(x_0 - e, x_0 + e)

    q = (abs(fi_equation.subs({x[0]: x_1, x[1]: x_fi}) -
             fi_equation.subs({x[0]: x_2, x[1]: x_fi}))) / (abs(x_1 - x_2))
    return q
```

```
def iteration_solve(system_equations: np.array, approx, tol=0.00001):
    print('\nIteration method computing...')

    n = system_equations.shape[0]
    x = symbols(f'x:{n}')

    fi_equations = system_equations[1]

    curr_roots = list(approx)

    errors = np.zeros(shape=(n, ))
    error = tol * 10000

    try:
        q_1 = float(get_q(fi_equations[0], (approx[0], approx[1]), 0.01))
        q_2 = float(get_q(fi_equations[1], (approx[1], approx[0]), 0.01))
    except:
        print("Q is not real, cant solve")
        return None, None
    print(f'q_1 = {q_1:.4f}')
    print(f'q_2 = {q_2:.4f}')
    if q_1>=1 or q_2>=1:
        print("q is greater than 1")
        return None, None

    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


    print('stopped iteration method\n')

    return curr_roots, iteration
```

Вспомогательная функция

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

    return dict_roots
```

Якобиан

```
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
```

Метод Ньютона

```
import numpy as np
from sympy import symbols
from staff_matrix import get_jacobi, roots_to_dict


def newton_solve(system_equations: np.array, approx, tol=0.00001):
    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)

        jacobi_det = np.linalg.det(jacobi_values)
        print(f"Jacobi det = {jacobi_det:.4f}")
        if not jacobi_det:
            print("det equal 0. Can't solve system")
            exit(0)

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

        delta_x = np.linalg.solve(jacobi_values, -1 * F)

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

    return roots, iteration
```

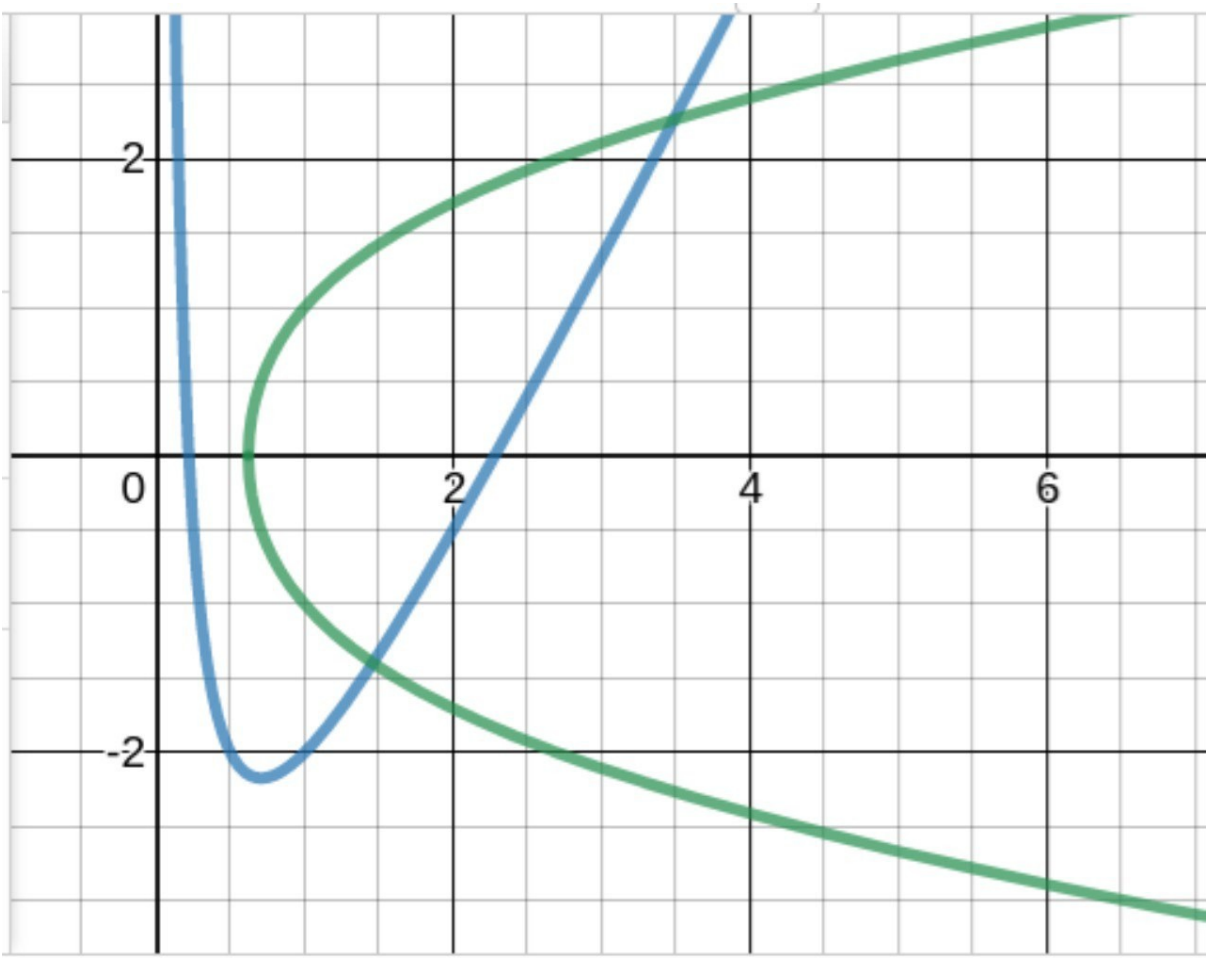
Полученные результаты

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

Решим систему для заданного начального приближения:

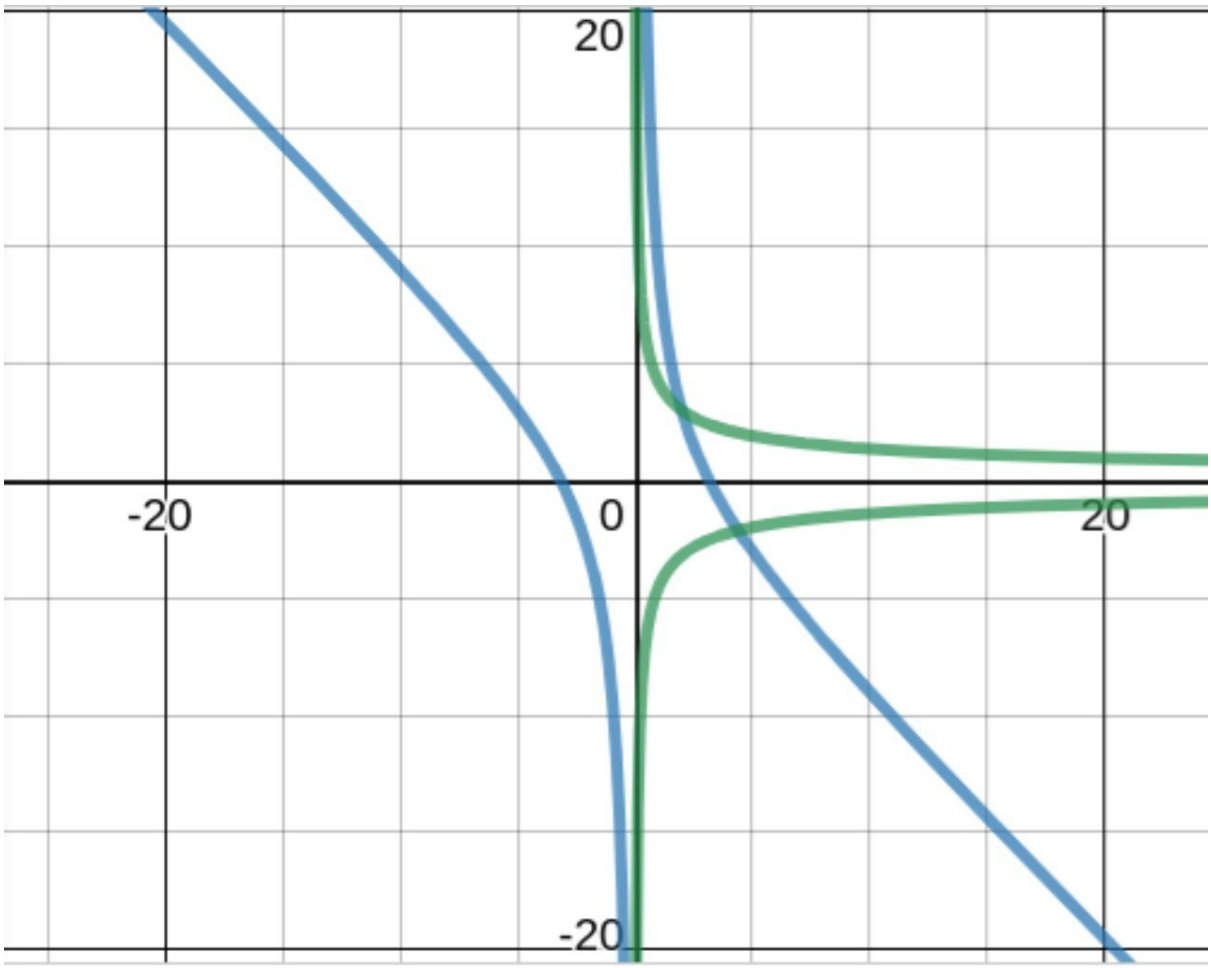
```
                Начальное приближение: (3.5, 2.2)

Iteration method computing...
q_1 = 0.5149
q_2 = 0.4416
stopped iteration method

Время выполнения метода итераций 0.14765 секунд
                *** Метод простой итерации: ***
3.4874 2.2616 Корни уравнения:
Количество итераций: 14
--------------------
--------------------
                Начальное приближение: (3.5, 2.2)
Jacobi det = -25.1171
Jacobi det = -25.4838
Jacobi det = -25.4619
Время выполнения метода ньютона 0.04758 секунд
                *** Метод Ньютона: ***
Корни уравнения:
3.4874 2.2616 Количество итераций: 3
```

Тестовый пример 2.

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

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

Построим график функции, чтобы найти начальное приближение корня:



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

Решим систему для заданного начального приближения:

```
                    Начальное приближение: (1.5, 3.5)

Iteration method computing...
q_1 = 0.8117
q_2 = 0.3213
stopped iteration method

Время выполнения метода итераций 0.12273 секунд
                  *** Метод простой итерации: ***
2.0000 3.0000 Корни уравнения:
Количество итераций: 12
--------------------
--------------------
                    Начальное приближение: (1.5, 3.5)
Jacobi det = 156.1250
Jacobi det = 197.7843
Jacobi det = 204.9696
Jacobi det = 204.9999
Время выполнения метода ньютона 0.03314 секунд
                  *** Метод Ньютона: ***
Корни уравнения:
2.0000 3.0000 Количество итераций: 4
```

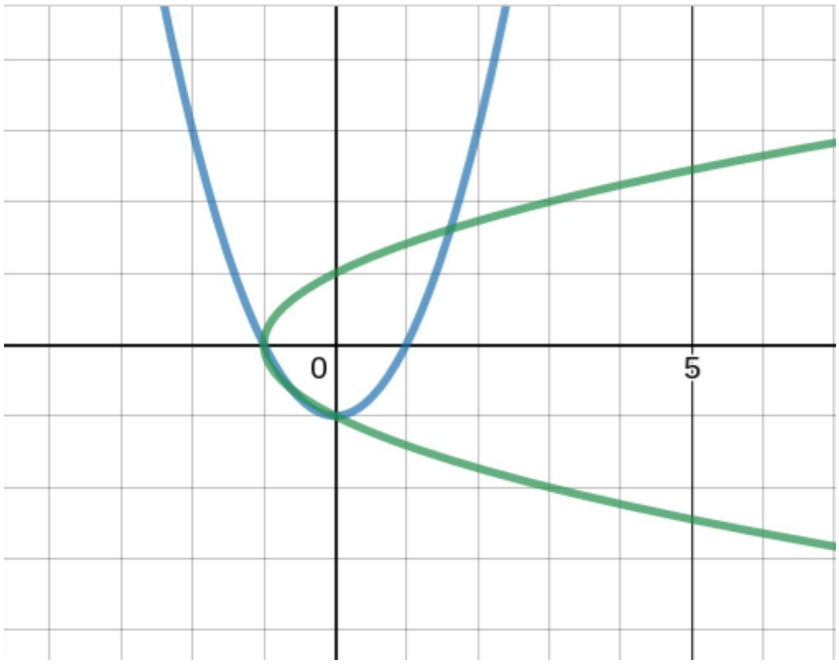
Тестовый пример 3.

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

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

Построим график функции, чтобы найти начальное приближение корня:

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



Решим систему для заданного начального приближения:

```
                    Начальное приближение: (1.9, 1.8)

Iteration method computing...
q_1 = 0.0000
q_2 = 0.2999
stopped iteration method

Время выполнения метода итераций 0.01028 секунд
                    *** Метод простой итерации: ***
1.6180 1.6180 Корни уравнения:
Количество итераций: 6
--------------------
--------------------
                    Начальное приближение: (1.9, 1.8)
Jacobi det = -12.6800
Jacobi det = -9.7416
Jacobi det = -9.4747
Jacobi det = -9.4721
Время выполнения метода ньютона 0.01993 секунд
                    *** Метод Ньютона: ***
Корни уравнения:
1.6180 1.6180 Количество итераций: 4
```

Можно заметить, что в этом примере время выолнения метода ньютона больше, чем в методе итераций. Это связано с тем, что количество итераций мало, а метод ньютона имеет большие накладные расчёты на вычисление

Решение задания:

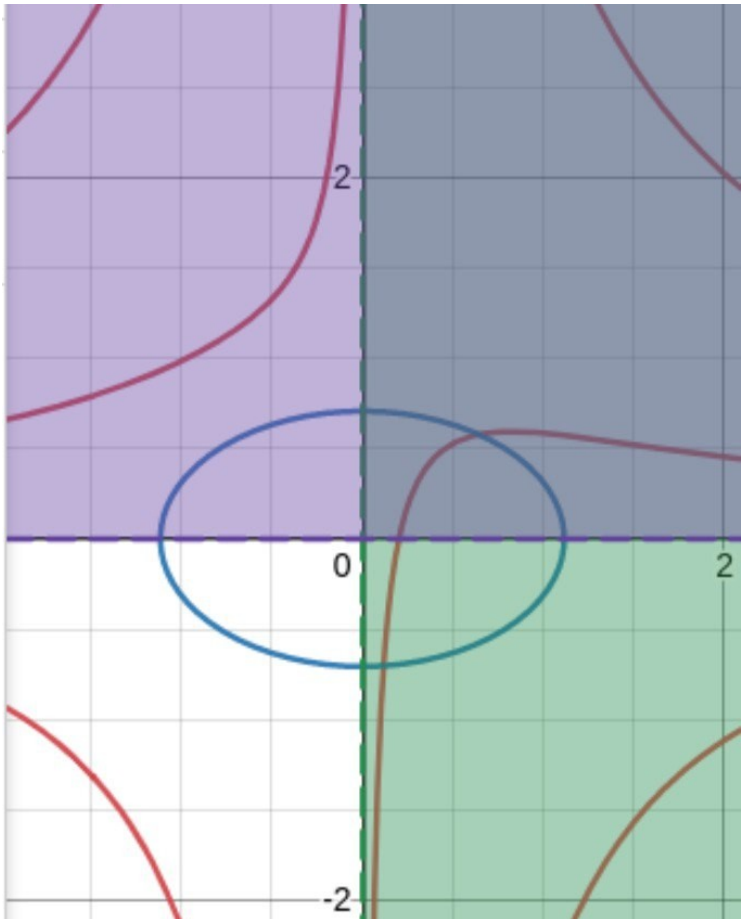
Вариант 3.

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

$$\begin{cases} tg(xy+m) = x \\ ax^2 + 2y^2 = 1, \end{cases}$$ где x>0, y>0,

```
m = 0.1
a = 0.7

approx = (0.6, 0.6)
```

Построим график функции, чтобы найти начальное приближение корня:



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

Решим систему для заданного начального приближения:

```
                    Начальное приближение: (0.6, 0.6)

Iteration method computing...
q_1 = 0.7349
q_2 = 0.3588
stopped iteration method

Время выполнения метода итераций 0.27196 секунд
                  *** Метод простой итерации: ***
0.3499 0.6761 Корни уравнения:
Количество итераций: 25
----------------------
----------------------
                    Начальное приближение: (0.6, 0.6)
Jacobi det = -1.2342
Jacobi det = -0.8602
Jacobi det = -0.8408
Jacobi det = -0.8444
Время выполнения метода ньютона 0.05677 секунд
                  *** Метод Ньютона: ***
Корни уравнения:
0.3499 0.6761 Количество итераций: 4
```

В данном задании метод ньютона показал себя ГОРАЗДО лучше чем метод итераций.

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, однако он очень сильно зависим от выбора начального приближения. В общем случае, результат всех итерационных методов зависит от выбора начального приближения.