Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет

информатики и радиоэлектроники»

Кафедра информатики

**Лабораторная работа № 5**

**«Численные методы решения систем нелинейных уравнений»**по учебной дисциплине «Методы численного анализа»

**Выполнили:**

студент гр. 153504 Климкович Н. В. студент гр. 153504 Тиханёнок И. А.  
студент гр. 153504 Тарасенко Ф. П.

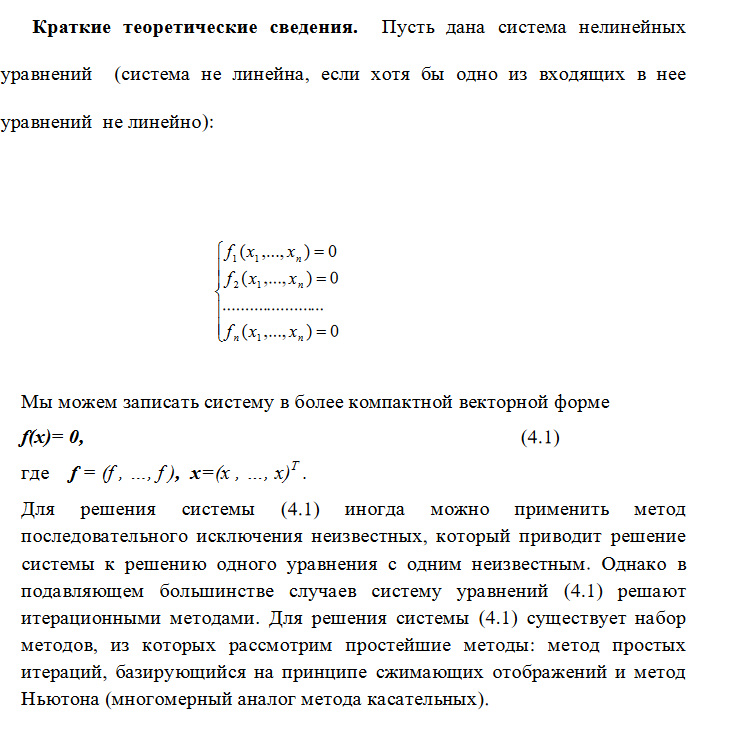
**Проверила:**

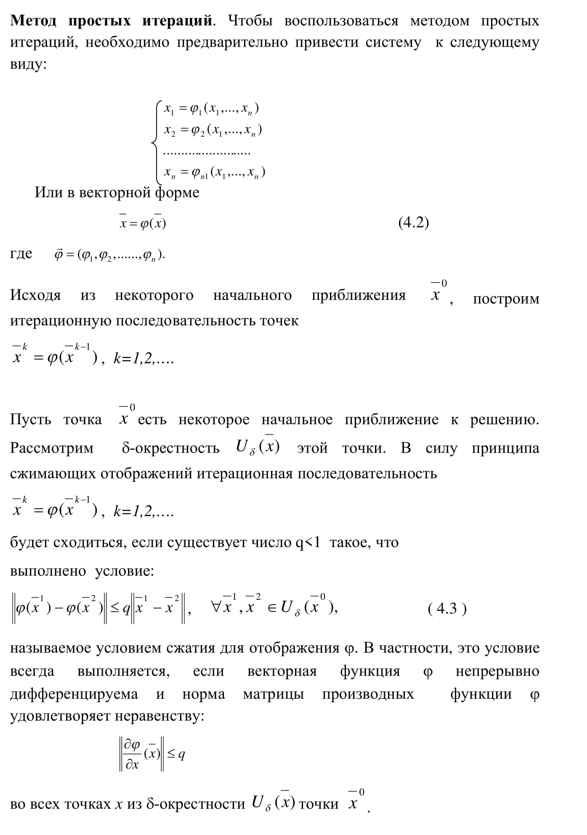
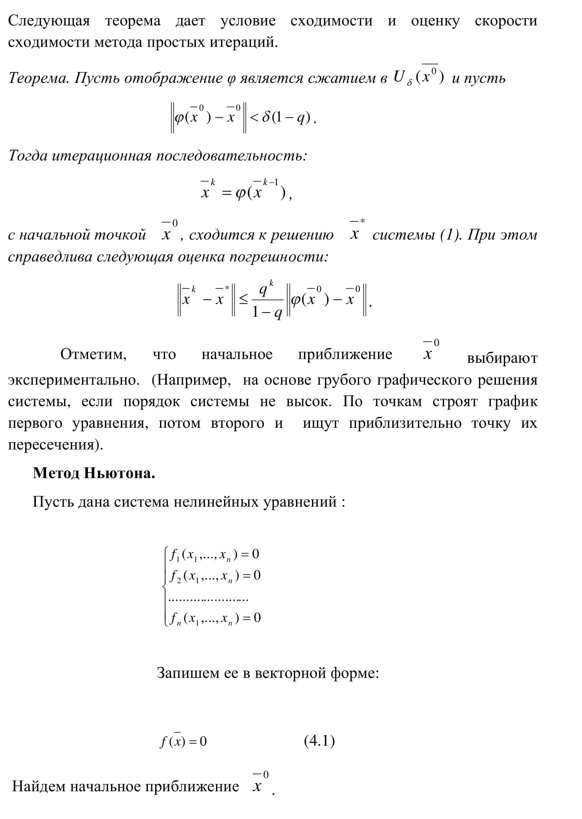
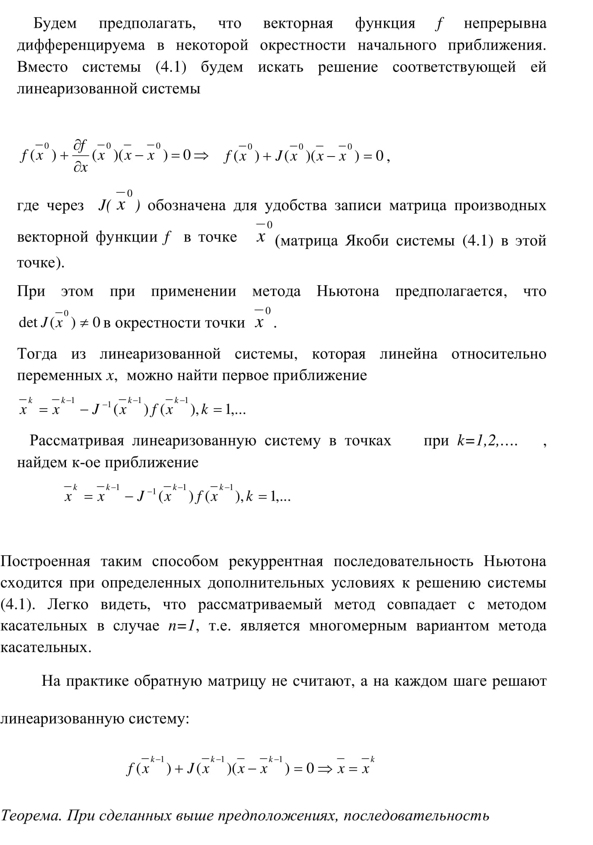
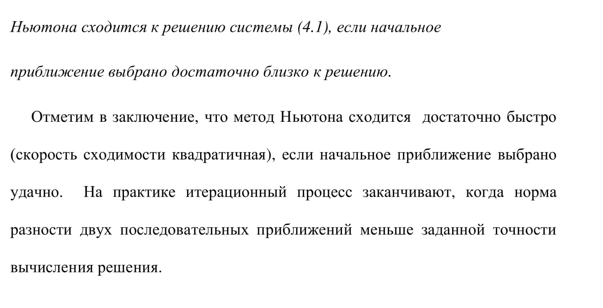
ст. преподаватель   
кафедры информатики Стройникова Е. Д.

Минск 2022

**Цели выполнения задания**

Изучить численные методы решения систем нелинейных уравнений простых итераций и Ньютона. Провести отделение решений, построить и запрограммировать итерационные алгоритмы методов, численно решить тестовое задание, сравнить трудоемкость методов.



**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, verbose=0):   
 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])  
 jacobi\_values = np.zeros(shape=(n, n))  
  
 roots\_d = roots\_to\_dict(curr\_roots, x)  
  
 for i in range(n):  
 for j in range(n):  
 jacobi\_values[i, j] = J[i, j].subs(roots\_d)  
  
 # compute q  
 if verbose == 1:  
 q\_1 = float(get\_q(fi\_equations[0], (approx[0], approx[1]), 0.1))  
 q\_2 = float(get\_q(fi\_equations[1], (approx[1], approx[0]), 0.1))  
 print(f'q\_1 = {q\_1}')  
 print(f'q\_2 = {q\_2}')  
 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  
 return curr\_roots, iteration

**#Метод Ньютона**

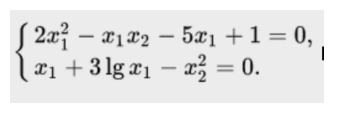
def newton\_solve(system\_equations: np.array, approx, tol=0.00001):  
 n = system\_equations.shape[0]  
 x = symbols(f'x:{n}')  
  
 J = get\_jacobi(system\_equations)  
  
 error = tol \* 10000  
 iteration = 0  
 roots = approx  
 while error > tol:  
 iteration += 1  
  
 roots\_d = roots\_to\_dict(roots, 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}")  
 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.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

**#Якоби**

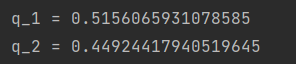
def get\_jacobi(system\_equations: np.array):  
 n = system\_equations.shape[0]  
 x = symbols(f'x:{n}')  
 J = np.empty(shape=(n, n), dtype=core.add.Add)  
 for i in range(n):  
 for j in range(n):  
 J[i, j] = system\_equations[i].diff(x[j])  
 return J

**Полученные результаты**

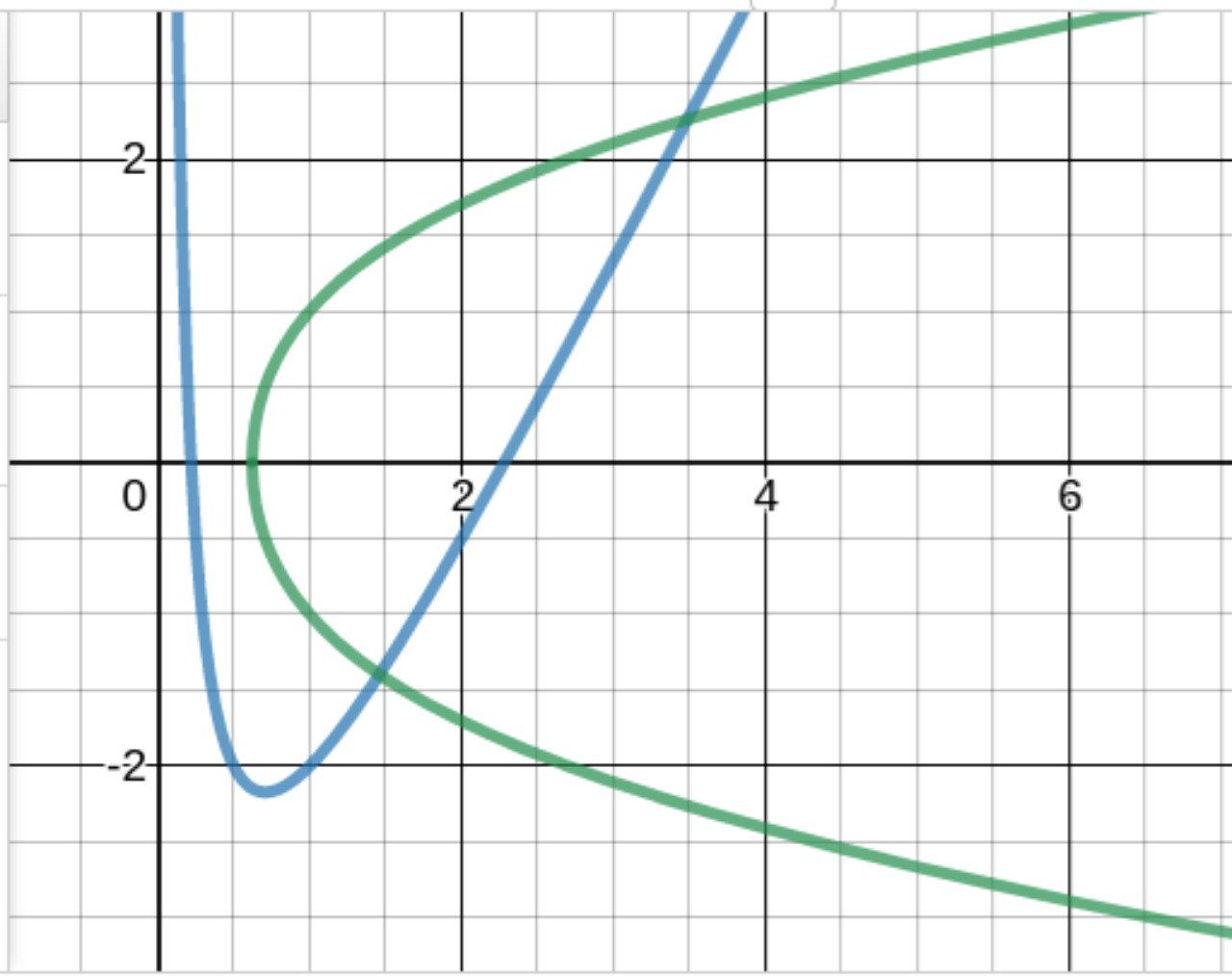
**Тестовый пример 1.**

****

Коэффициент сжатия q:

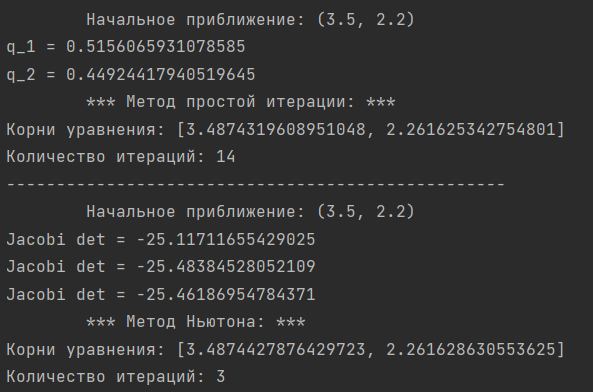


Построим график функции, чтобы найти начальное приближение корня:



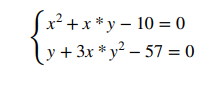
Из графика видно, что начальное приближение для верхнего корня можно взять x = 3.5, y = 2.2

Решим систему для заданного начального приближения:

****

|  |  |
| --- | --- |
| Метод простых итераций | Метод Ньютона |
| 3.4874319608951048  2.261625342754801 | 3.4874427876429723  2.26162863055362 |
| Количество итераций | |
| 14 | 3 |

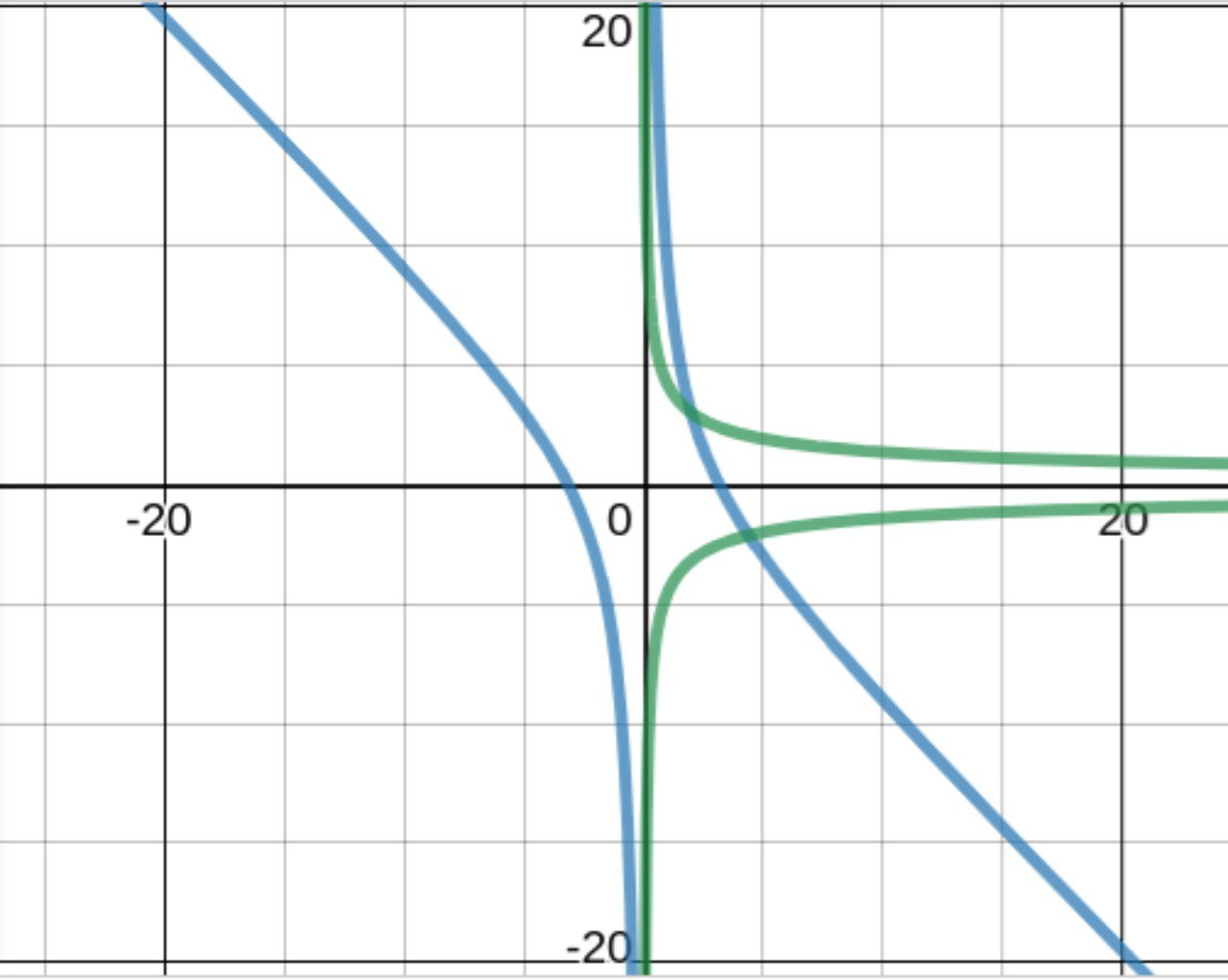
**Тестовый пример 2.**

****

Коэффициент сжатия q:

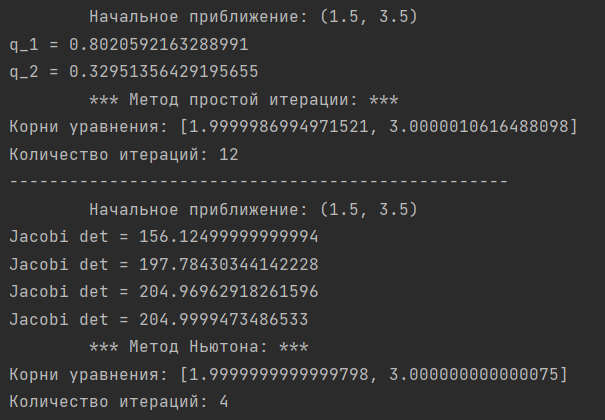
****

Построим график функции, чтобы найти начальное приближение корня:



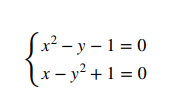
Из графика видно, что начальное приближение для верхнего корня можно взять x = 1.5, y = 3.5

Решим систему для заданного начального приближения:

****

|  |  |
| --- | --- |
| Метод простых итераций | Метод Ньютона |
| 1.9999986994971521,  3.0000010616488098 | 1.9999999999999798,  3.000000000000075 |
| Количество итераций | |
| 12 | 4 |

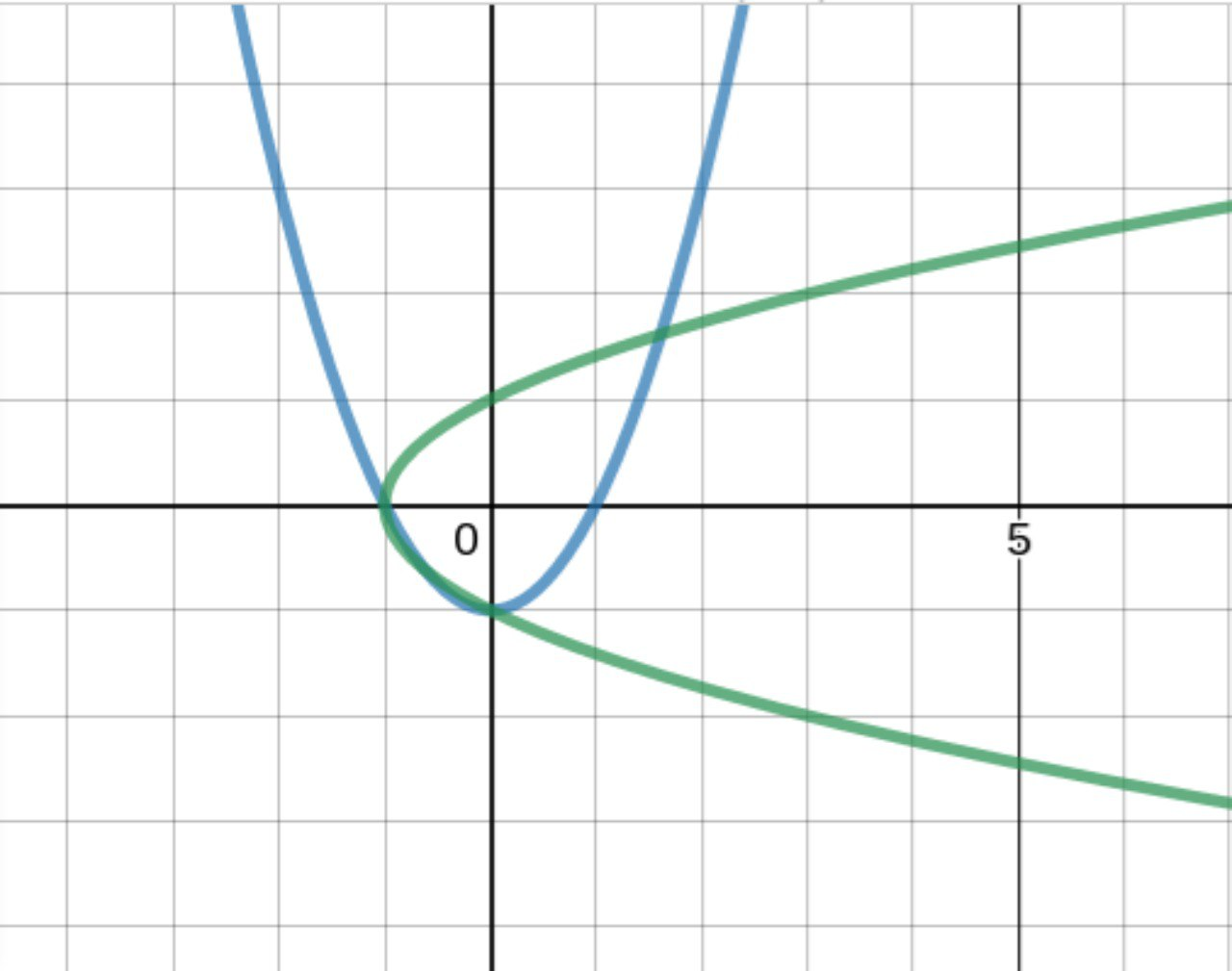
**Тестовый пример 3.**

****

Коэффициент сжатия q:

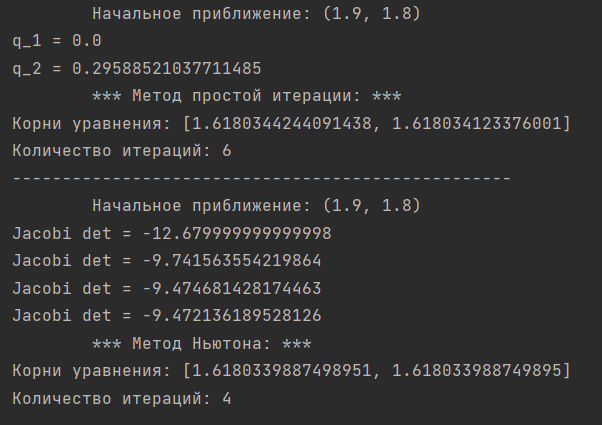
****

Построим график функции, чтобы найти начальное приближение корня:



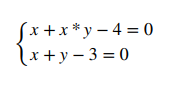
Из графика видно, что начальное приближение для верхнего корня можно взять x = 1.9, y = 1.8

Решим систему для заданного начального приближения:

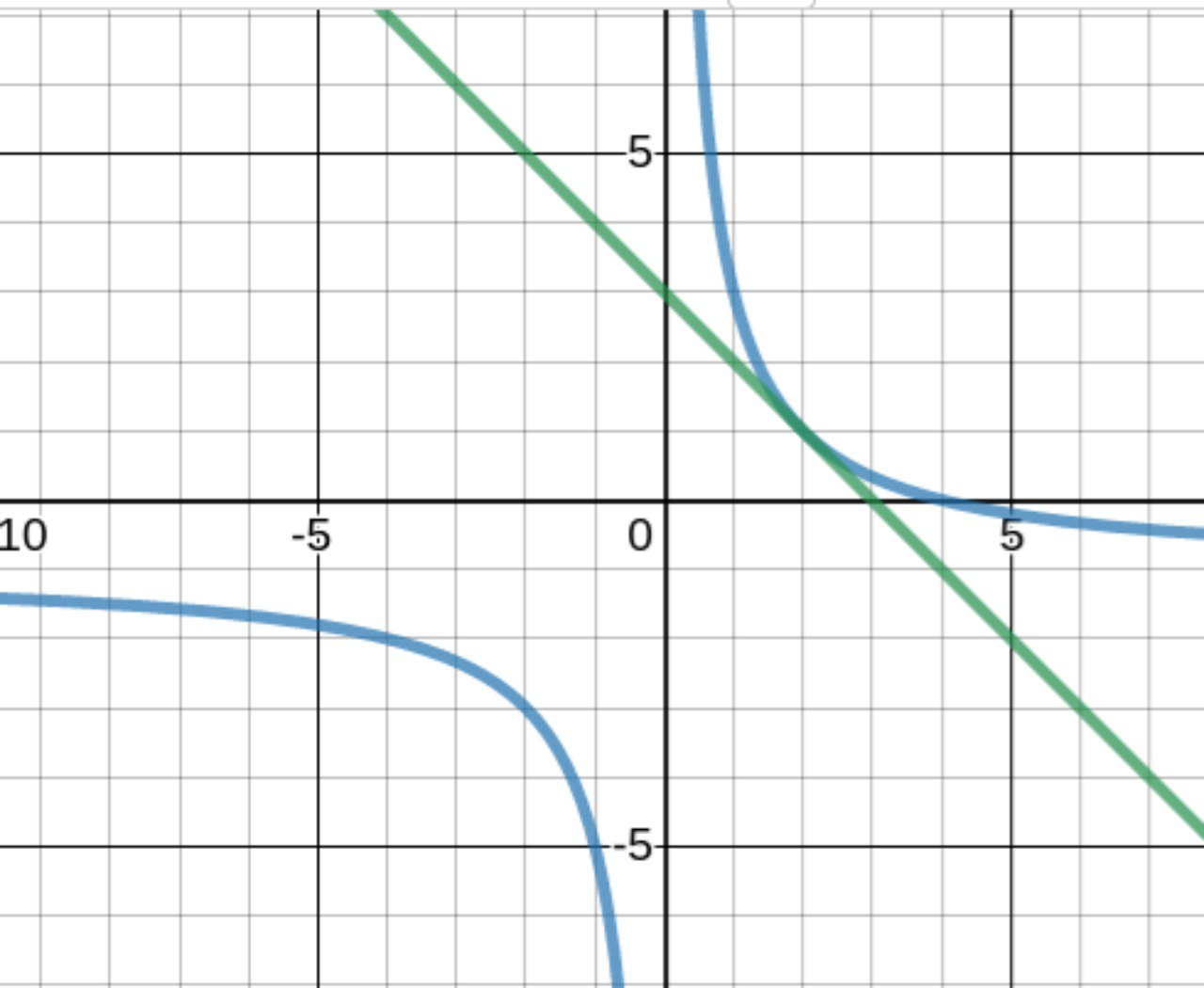
****

|  |  |
| --- | --- |
| Метод простых итераций | Метод Ньютона |
| 1.6180344244091438,  1.618034123376001 | 1.6180339887498951,  1.618033988749895 |
| Количество итераций | |
| 6 | 4 |

**Тестовый пример 4.**

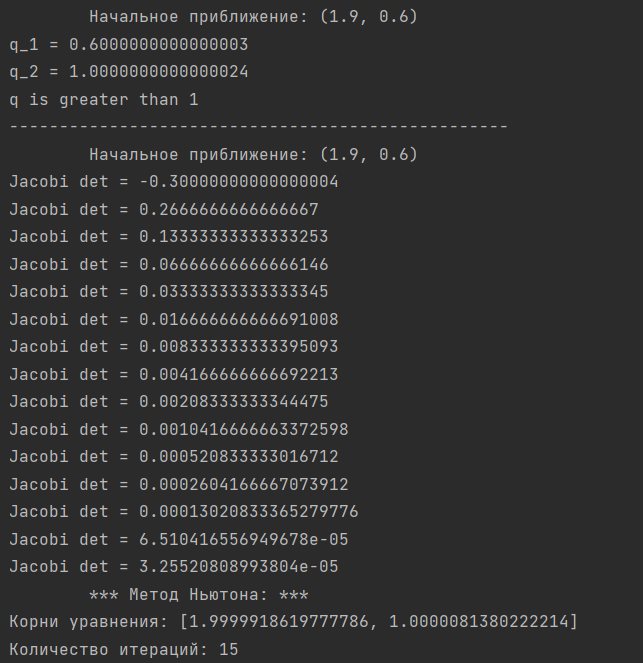
****

Построим график функции, чтобы найти начальное приближение корня:



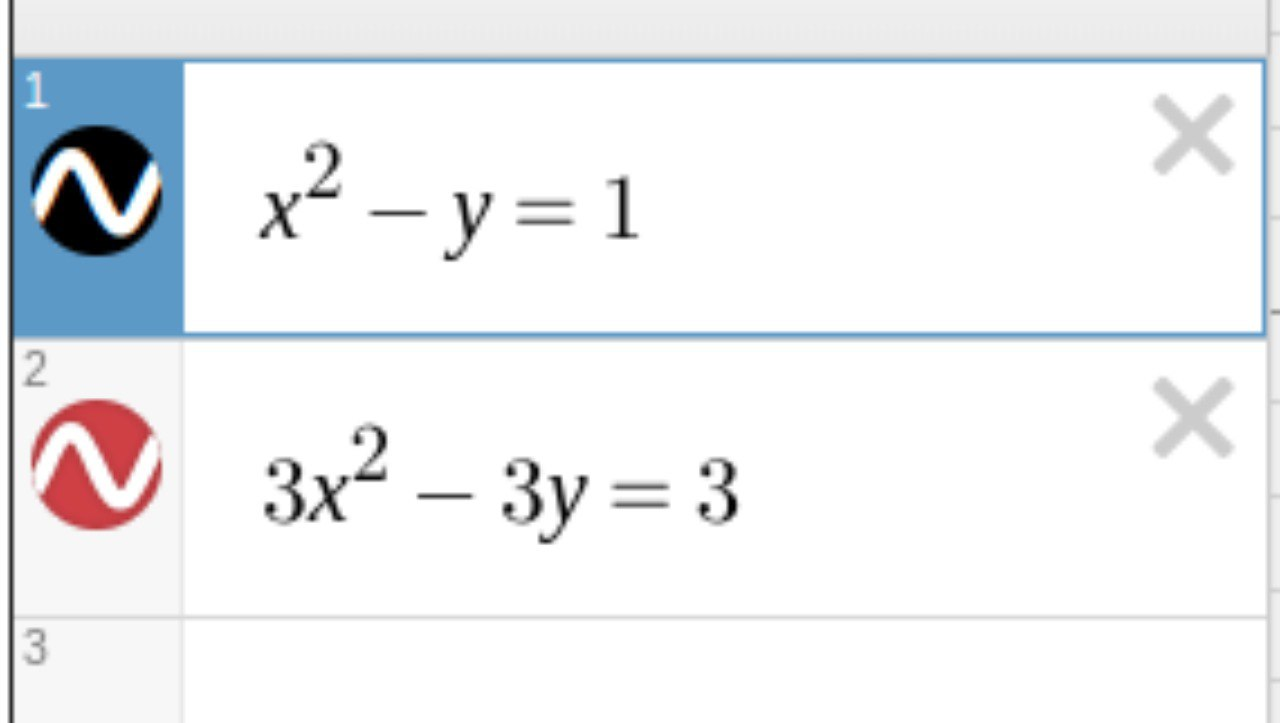
Из графика видно, что начальное приближение для верхнего корня можно взять x = 1.9, y = 0.6

Решим систему для заданного начального приближения:

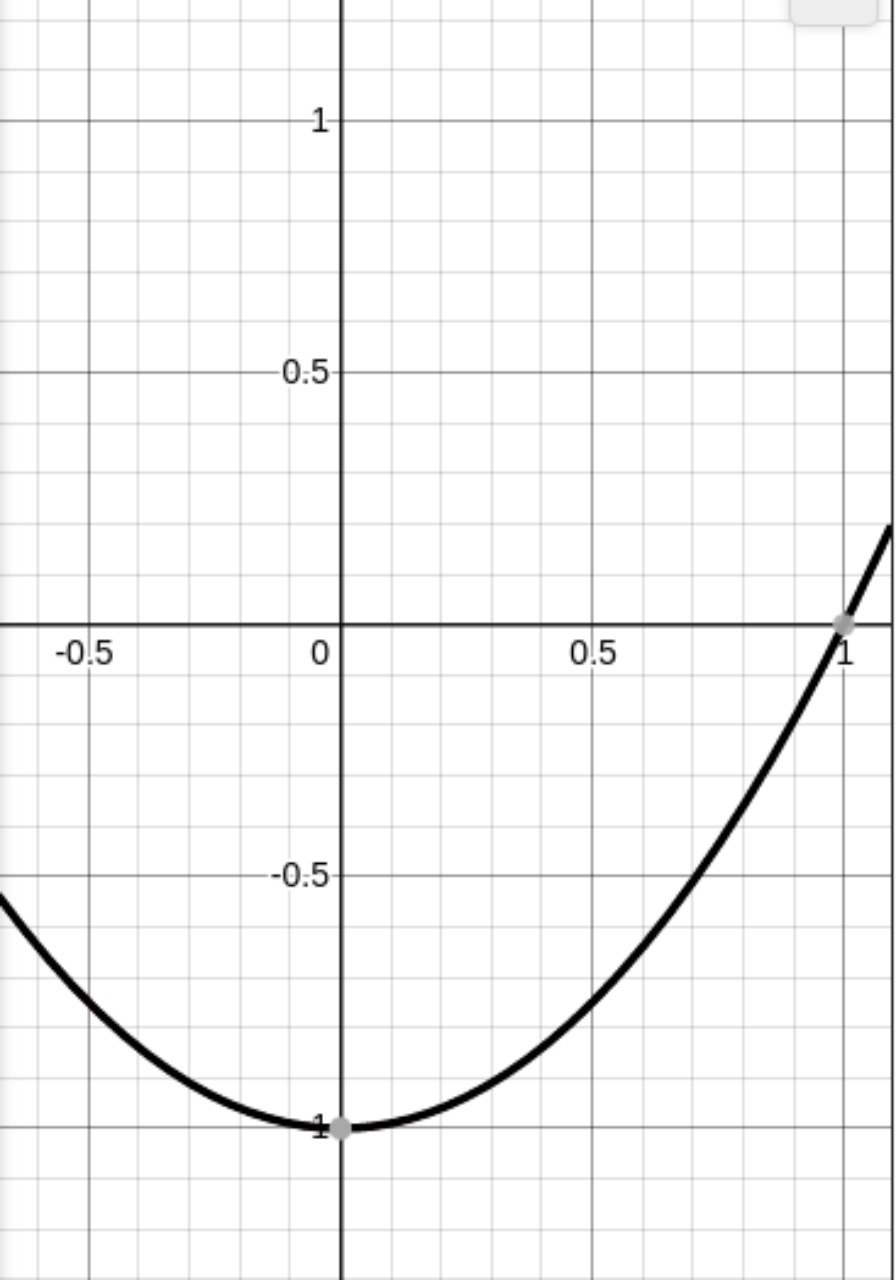
****

Метод простых итерации не дает ответ, так как q>1

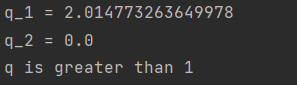
**Тестовый пример 5.**



Построим график функции, чтобы найти начальное приближение корня:



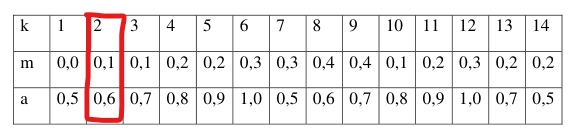
Из графика видно, что функции зависимы, а графики совпадают.

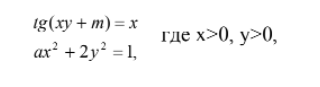


Определитель Якобиана равен 0, поэтому метод Ньютона не дает ответ.

**Решение задания:**

**Вариант 4.**

****



Построим график функции, чтобы найти начальное приближение корня:

****

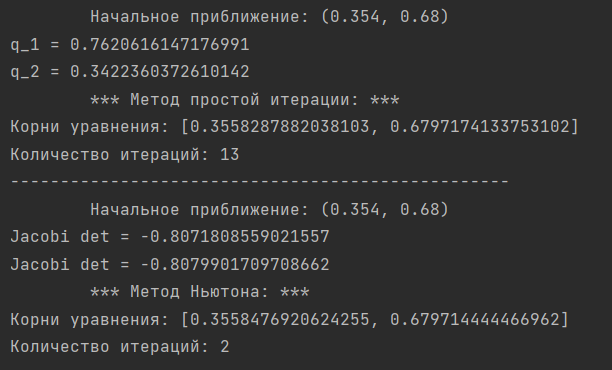
Из графика видно, что начальное приближение для верхнего корня можно взять x = 0.354, y = 0.68

Коэффициент сжатия q:

q\_1 = 0.7620616147176991

q\_2 = 0.3422360372610142

Решим систему для заданного начального приближения:

****

|  |  |
| --- | --- |
| Метод простых итераций | Метод Ньютона |
| 0.3558287882038103 0.6797174133753102 | 0.3558476920624255 0.679714444466962 |
| Количество итераций | |
| 13 | 2 |

# 

# **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.