Федеральное государственное автономное
образовательное учреждение высшего образования
«Санкт-Петербургский национальный исследовательский университет
ИТМО»

ФАКУЛЬТЕТ ПРОГРАММНОЙ ИНЖЕНЕРИИ И КОМПЬЮТЕРНОЙ
ТЕХНИКИ

ЛАБОРАТОРНАЯ РАБОТА № 2

по дисциплине

‘ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА’

Вариант №14

Выполнила
Студентка группы P32151
Ярусова Анна Александровна
Преподаватель:
Машина Екатерина Алексеевна

Санкт-Петербург,
2023

Цель

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

Задание

1. № варианта определяется как номер в списке группы согласно ИСУ.
2. **Вычислительная часть:**
   - Отделить корни заданного нелинейного уравнения графически ($2,3x^3 + 5,75x^2 - 7,41x - 10,6$)
   - Определить интервалы изоляции корней.
   - Уточнить корни нелинейного уравнения с точностью ε=10^{-2}
   - Используемые методы для уточнения каждого из 3-х корней многочлена - метод Ньютона, метод простой итерации, метод половинного деления.
   - Вычисления оформить в виде соответствующих таблиц. Для всех значений в таблице удержать 3 знака после запятой.
3. **Программная часть:**

   <u>Для нелинейных уравнений:</u>
   - Все численные методы (метод хорд, метод секущих, метод простой итерации) должны быть реализованы в виде отдельных подпрограмм/методов/классов.
   - 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), 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:

- 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.
- Организовать вывод вектора неизвестных: x_1, x_2.
- Организовать вывод количества итераций, за которое было найдено решение.
- Организовать вывод вектора погрешностей $|x_i^{(k)} - x_i^{(k-1)}|$
- Проверить правильность решения системы нелинейных уравнений.

Описание метода, расчетные формулы

<u>Метод Ньютона:</u>

Идея метода: функция $y = f(x)$ на отрезке [a, b] заменяется касательной и в качестве приближенного значения корня $x^* = x_n$ принимается точка пересечения касательной с осью абсцисс.

Перед расчётом корня необходимо проверить достаточное условие сходимости:

- функция y= f(x) определена и непрерывна на отрезке $[a; b]$;
- f(a)·f(b) < 0 (на концах отрезка [a;b] функция имеет разные знаки);
- производные f'(x) и f''(x) сохраняют знак на отрезке [a;b];
- производная f'(x)≠0.

Также в качестве начального приближение для быстрой сходимости выбирается тот конец интервала, для которого знаки функции и второй производной совпадают.

После этого x_i рассчитывается по формуле

$x_i = x_{i-1} + f(x_{i-1}) / f'(x_{i-1})$, $x_1 = x_0 - h_0 = x_0 - f(x_0) / f'(x_0)$

Критерии окончания итерационного процесса:

$|x_n - x_{n-1}|$, $|f(x_n) / f'(x_n)|$, $|f(x_n)|$

Приближенное значение корня: $x^* = x_n$

<u>Метод простой итерации:</u>

Идея метода: уравнение $f(x) = 0$ приводим к эквивалентному виду: $x = \varphi(x)$, выразив x из исходного уравнения.

Зная начальное приближение: $x_0 \in [a, b]$, найдем очередные приближения:

x_i рассчитывается по формуле $x_i = \varphi(x_{i-1})$

Также необходимо проверить достаточное условие сходимости: $|\varphi'(x)| \leq q < 1$.

Критерии окончания итерационного процесса: $|x_n - x_{n-1}|$

<u>Метод половинного деления:</u>

Идея метода: начальный интервал изоляции корня делим пополам, получаем начальное приближение к корню: $x_0 = (a_0 + b_0) / 2$.

Вычисляем f(x_0). В качестве нового интервала выбираем ту половину отрезка, на концах которого функция имеет разные знаки: $[a_0, x_0]$ либо $[b_0, x_0]$. Другую половину отрезка $[a_0, b_0]$, на которой функция f(x) знак не меняет, отбрасываем. Новый интервал вновь делим пополам, получаем очередное приближение к корню: $x_1 = (a_1 + b_1) / 2$ и т.д

x_i рассчитывается по формуле $x_i = (a_i + b_i) / 2$.

Критерии окончания итерационного процесса: $|b_n - a_n|, |f(x_n)|$.

Приближенное значение корня: $x^* = (a_n + b_n) / 2$ или $x^* = a_n$ или $x^* = b_n$.

<u>Метод хорд:</u>

Идея метода: функция $y = f(x)$ на отрезке [a, b] заменяется хордой и в качестве приближенного значения корня принимается точка пересечения хорды с осью абсцисс.

Уравнение хорды, проходящей через точки $A(a, f(a))$ и $B(b, f(b))$:

$(y - f(a)) / (f(b) - f(a)) = (x - a) / (x - b)$.

Точка пересечения хорды с осью абсцисс ($y = 0$):

$x_0 = a_0 - (b_0 - a_0) * f(a_0) / (f(b_0) - f(a_0)))$.

Вычисляем f(x_0), в качестве нового интервала выбираем ту половину отрезка, на концах которого функция имеет разные знаки $[a_0, x_0]$ либо $[b_0, x_0]$.

x_i рассчитывается по формуле

$x_i = (a_i f(b_i) - b_i f(a_i)) / (f(b_i) - f(a_i))$.

Критерии окончания итерационного процесса:

$|b_n - a_n|, |f(x_n)|, |x_n - x_{n-1}|$.

Приближенное значение корня: $x^* = x_n$.

<u>Метод секущих:</u>

Метод является упрощением метода Ньютона - заменой f'(x) разностным приближением: $f'(x_i) \approx (\ f(x_i)\ -\ f(x_{i-1}))\ /\ x_i\ -\ x_{i-1})$

x_i рассчитывается по формуле

$x_{i+1}\ = x_i\ +\ f(x_i)\ *\ (x_i\ -\ x_{i-1})\ /\ (f\ (x_i)\ -\ f\ (x_{i-1})\ )$, x_0 как в методе Ньютона, x_1 рядом с начальным самостоятельно.

Метод секущих является двухшаговым, т.е. новое приближение x_{i+1} определяется двумя предыдущими итерациями x_i и x_{i-1}.

Критерии окончания итерационного процесса: $|x_n\ -\ x_{n-1}|$, $|f(x_n)|$.

Приближенное значение корня: $x^* = x_n$

Метод простой итерации для систем:

Метод является наложением метода простой итерации только для систем. Приведем систему уравнений к эквивалентному виду:

$$\begin{cases} F_1(x_1, x_2, \dots, x_n) = 0 \\ F_2(x_1, x_2, \dots, x_n) = 0 \\ \dots\dots\dots\dots\dots\dots\dots. \\ F_n(x_1, x_2, \dots, x_n) = 0 \end{cases} \qquad \begin{cases} x_1 = \varphi_1(x_1, x_2, \dots, x_n) \\ x_2 = \varphi_2(x_1, x_2, \dots, x_n) \\ \dots\dots\dots\dots\dots\dots\dots. \\ x_n = \varphi_n(x_1, x_2, \dots, x_n) \end{cases}$$

Или, в векторной форме: $\boldsymbol{X} = \boldsymbol{\varphi}(\boldsymbol{X})$ $\quad \boldsymbol{\varphi}(\boldsymbol{X}) = \begin{pmatrix} \varphi_1(\boldsymbol{X}) \\ \varphi_2(\boldsymbol{X}) \\ \dots\dots \\ \varphi_n(\boldsymbol{X}) \end{pmatrix}$

Если выбрано начальное приближение: $X^{(0)} = x_1^{(0)}, x_2^{(0)}, \dots, x_n^{(0)}$, последующие приближения находятся по формулам:

$$\begin{cases} x_1^{(k+1)} = \varphi_1(x_1^k, x_2^k, \dots, x_n^k) \\ x_2^{(k+1)} = \varphi_2(x_1^k, x_2^k, \dots, x_n^k) \\ \dots\dots\dots\dots\dots\dots\dots. \\ x_n^{(k+1)} = \varphi_n(x_1^k, x_2^k, \dots, x_n^k) \end{cases} \qquad k = 0, 1, 2, \dots$$

При этом также необходимо проверить достаточное условие сходимости:

$$\max_{[x \in G]} |\varphi'(x)| \leq q < 1 \text{ или } \max_{[x \in G]} \max_{[i]} \sum_{j=1}^{n} \left| \frac{\partial \varphi_i(X)}{\partial x_j} \right| \leq q < 1$$

$$\varphi'(x) = \begin{bmatrix} \frac{\partial \varphi_1}{\partial x_1} & \frac{\partial \varphi_1}{\partial x_2} & \cdots & \frac{\partial \varphi_1}{\partial x_n} \\ \frac{\partial \varphi_2}{\partial x_1} & \frac{\partial \varphi_2}{\partial x_2} & \cdots & \frac{\partial \varphi_n}{\partial x_n} \\ \cdots & \cdots & \cdots & \\ \frac{\partial \varphi_n}{\partial x_1} & \frac{\partial \varphi_n}{\partial x_2} & \cdots & \frac{\partial \varphi_n}{\partial x_n} \end{bmatrix}$$

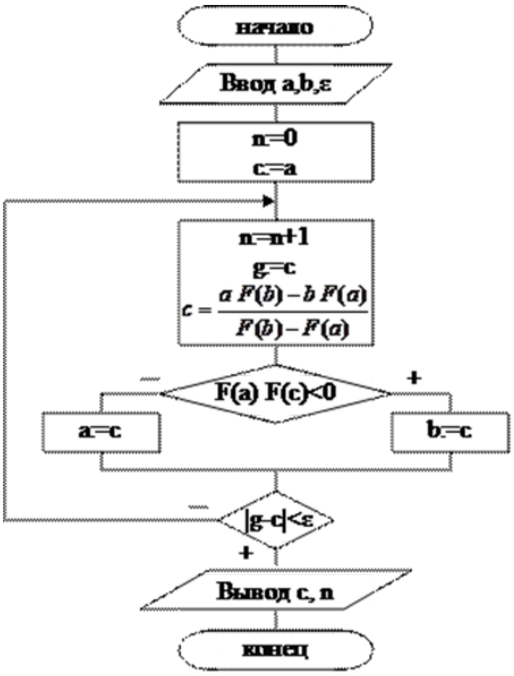
Если $X^{(0)} \in G$ и все последовательные приближения: $X^{(k+1)} = \boldsymbol{\varphi}(X^{(k)})$, k = 0, 1 , 2 …

также содержатся в ограниченной замкнутой области G, тогда итерационный процесс сходится к единственному решению уравнения $\boldsymbol{X}=\boldsymbol{\varphi}(\boldsymbol{X})$

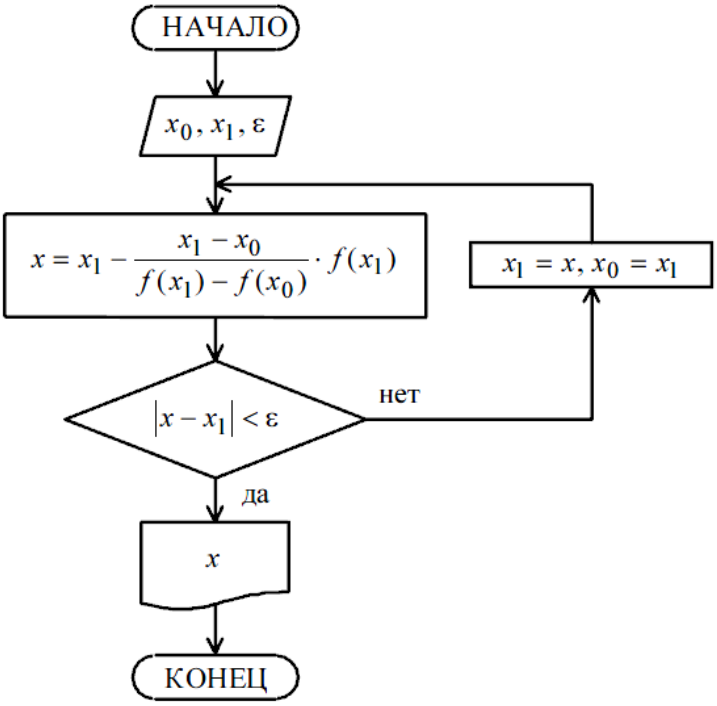
Критерии окончания итерационного процесса: $max\ |x_i^{(k+1)} - x_i^{(k)}|$

Блок-схема

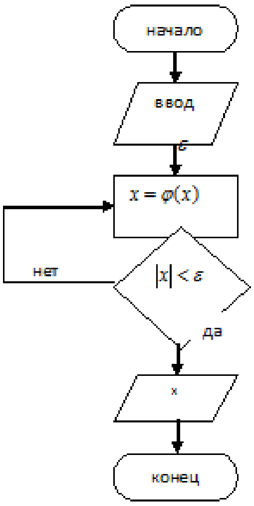
Метод хорд:



Метод секущих:



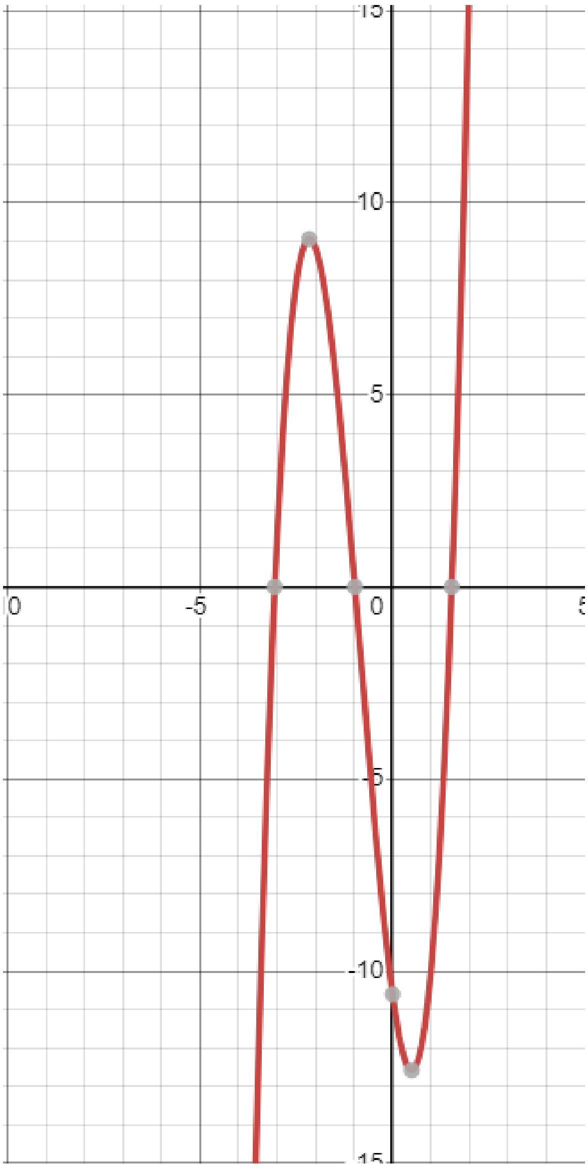
Метод простой итерации:



Вычислительная часть

$2,3x^3 + 5,75x^2 - 7,41x - 10,6$

График:



Корни и интервалы изоляции корней:

$x_1 = -3,06 \in (-4, -2)$; $x_2 = -0,98 \in (-2, 0)$; $x_3 = 1,54 \in (0, 2)$;

Метод Ньютона(Таблица 3):

$(-4, -2)$

№ итерации	x_k	$f(x_k)$	$f'(x_k)$	x_{k+1}	$\lvert x_{k+1} - x_k \rvert$
1	-4	-36,16	56,99	-3,366	0,634
2	-3,366	-8,225	32,058	-3,109	0,257
3	-3,109	-1,101	23,531	-3,062	0,046
4	-3,062	-0,03	22,07	-3,061	0,001

Результат: -3,061

Метод простой итерации(Таблица 5):

$(-\ 2,\ 0)$

$f(x)\ =\ 2,3x^3 + 5,75x^2 - 7,41x\ -\ 10,6$

$f'(x)\ =\ 6,9x^2 + 11,5x - 7,41$

$max\,|f'(x)|\ = 7321/600$

$\lambda\ =-\ 600/7321$

$\varphi(x)\ = x\ -\ 600/7321\ *\ (2,3x^3 + 5,75x^2 - 7,41x\ -\ 10,6)$

$\varphi'(x)\ = 1\ -\ 600/7321\ *\ (\ 6,9x^2 + 11,5x - 7,41)$

$q\ =\ max\,|\varphi'(x)|\ =\ 2\ >\ 1$

Значит, достаточное условие не выполняется.

№ итерации	x_k	x_{k+1}	$f(x_{k+1})$	$\lvert x_{k+1}\ -\ x_k\rvert$

Метод половинного деления(Таблица 1):

$(0,\ 2)$

№ шага	a	b	x	f(a)	f(b)	f(x)	|a - b|
1	0	2	1	-10,6	15,98	-9,96	2
2	1	2	1,5	-9,96	15,98	-1,015	1
3	1,5	2	1,75	-1,015	15,98	6,368	0,5
4	1,5	1,75	1,625	-1,015	6,368	2,411	0,25
5	1,5	1,625	1,5625	-1,015	2,411	0,634	0,125
6	1,5	1,5625	1,53125	-1,015	0,634	-0,207	0,063

7	1,53125	1,5625	1,546875	-0,207	0,634	0,21	0,031
8	1,53125	1,546875	1,5390625	-0,207	0,21	0,001	0,016
9	1,53125	1,5390625	1,53515625	-0,207	0,001	-0,103	0,007
10	1,53515625	1,5390625	1,537109375	-0,103	0,001	-0,051	0,003
11	1,537109375	1,5390625	1,5380859375	-0,051	0,001	-0,025	0,002
12	1,5380859375	1,5390625	1,53857421875	-0,025	0,001	-0,012	0,001
13	1,53857421875	1,5390625	1,538818359375	-0,012	0,001	-0,006	0,001

Результат: 1,538818359375 ≈ 1,54

Листинг численного метода

Метод хорд:

```
def chord_method(equation, a, b, accuracy, max_iterations):
    iterations = 0
    x = (a * equation(b) - b * equation(a)) / (equation(b) - 
equation(a))
    prev = a
    while (iterations < max_iterations) and (abs(equation(x)) 
> accuracy) and (abs(b - a) > accuracy) and (abs(x - prev) > 
accuracy):
        iterations += 1
        if equation(a) * equation(x) < 0:
            b = x
        else:
            a = x
        prev = x
        x = (a * equation(b) - b * equation(a)) / (equation(b) 
- equation(a))
    return x, iterations
```

Метод секущих:

```
def secant_method(f, a, b, accuracy, max_iterations):
    if f(a) * derivative(f, a, dx=0.01, n=2) > 0:
        x_prev = a
    else:
        x_prev = b
    iterations = 0
```

```
    x = x_prev - f(x_prev) / derivative(f, x_prev, dx=0.01, n=1)
    while (iterations < max_iterations) and (abs(f(x)) > accuracy) and (abs(x_prev - x) > accuracy):
        iterations += 1
        buf = x
        x = x - f(x) * (x - x_prev) / (f(x) - f(x_prev))
        x_prev = buf
    return x, iterations
```

<u>Метод простой итерации:</u>

```
def simple_iteration_method(equation, a, b, accuracy, max_iterations):
    max_derivative = abs(derivative(equation, a, n=1))
    for i in np.arange(a, b, accuracy):
        if max_derivative < abs(derivative(equation, i, n=1)):
            max_derivative = abs(derivative(equation, i, n=1))
    l = -1 / max_derivative
    phi_f = lambda x: l * equation(x) + x
    iterations = 0
    q = abs(derivative(phi_f, a, n=1))
    for i in np.arange(a, b, accuracy):
        if abs(derivative(phi_f, i, n=1)) > q:
            q = abs(derivative(phi_f, i, n=1))
    if q >= 1:
        print("The convergence condition is not met")
        return None, None
    x_prev = a
    x = phi_f(a)
    while (iterations < max_iterations) and abs(x - x_prev) >= accuracy:
        iterations += 1
        buf = x
        x = phi_f(x_prev)
        x_prev = buf
    return x, iterations
```

<u>Метод простой итерации для систем:</u>

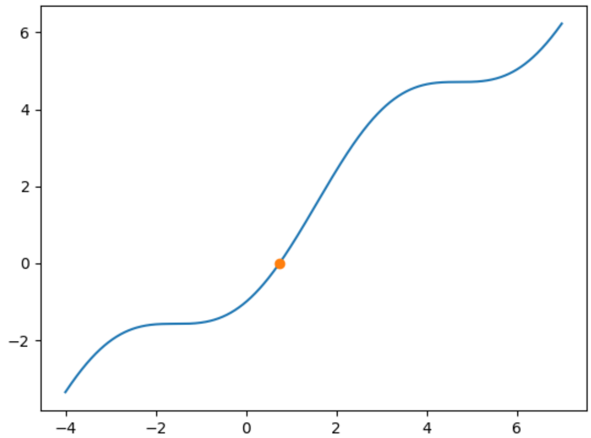
```
x, y = symbols('x y')
phi = {
    1: (4 - y ** 2) ** (1 / 2),
    2: x ** 2,
    3: 0.3 - 0.1 * x ** 2 - 0.2 * y ** 2,
```

```
    4: 0.7 - 0.2 * x ** 2 - 0.1 * x * y,
}
def simple_iteration_method_for_systems(f, g, a_1, b_1, a_2,
b_2, x0, y0, accuracy, max_iterations):
    phi_f = lambdify([x, y], phi[f])
    phi_g = lambdify([x, y], phi[g])
    iterations = 0
    q = abs(lambdify([x, y], Derivative(phi[f],
x).doit())(a_1, a_2)) + abs(lambdify([x, y],Derivative(phi[f],
y).doit())(a_1, a_2))
    vector = [[], []]
    df_x = lambdify([x, y], Derivative(phi[f], x).doit())
    df_y = lambdify([x, y], Derivative(phi[f], y).doit())
    dg_x = lambdify([x, y], Derivative(phi[g], x).doit())
    dg_y = lambdify([x, y], Derivative(phi[g], y).doit())
    for i in np.arange(a_1, b_1, accuracy):
        for j in np.arange(a_2, b_2, accuracy):
            if abs(df_x(i, j)) + abs(df_y(i, j)) > q:
                q = abs(df_x(i, j)) + abs(df_y(i, j))
            if abs(dg_x(i, j)) + abs(dg_y(i, j)) > q:
                q = abs(dg_x(i, j)) + abs(dg_y(i, j))
            if q >= 1:
                print("The convergence condition is not met")
                return None, None, None
    xi, yi = x0, y0
    x_prev, y_prev = xi + accuracy + 1, yi + accuracy + 1
    while (iterations < max_iterations) and (abs(x_prev - xi)
> accuracy) and (abs(y_prev - yi) > accuracy):
        iterations += 1
        buf_x, buf_y = xi, yi
        xi, yi = phi_f(buf_x, buf_y), phi_g(buf_x, buf_y)
        x_prev, y_prev = buf_x, buf_y
        vector[0].append(abs(xi - x_prev))
        vector[1].append(abs(yi - y_prev))
    return xi, yi, iterations, vector
```

Пример работы

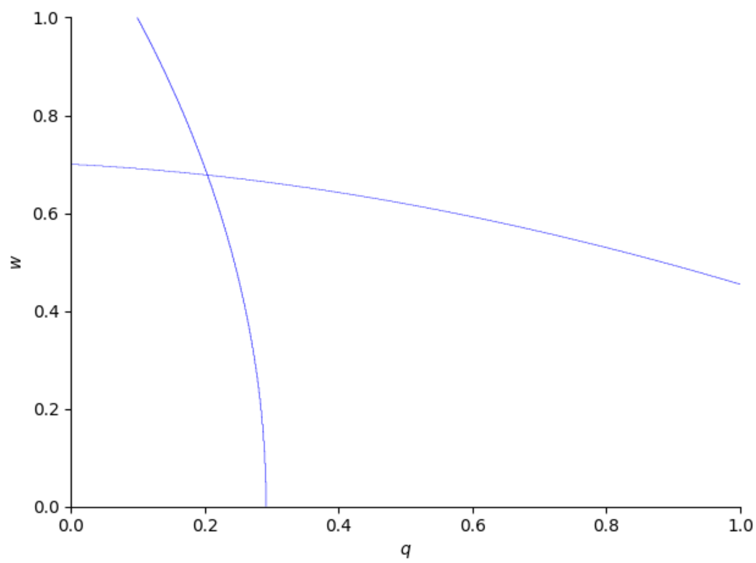
<u>Пример 1:</u>

```
Choose the method:
1. Chord method (default)
2. Secant method
3. Simple iteration method
4. Simple iteration method for systems
1
Choose the equation:
1. x^3 - 3x^2 + 2 = 0 (default)
2. x - cos(x) = 0
3. x^3 - 2x^2 - 5x + 6 = 0
4. 2,3x^3 + 5,75x^2 - 7,41x - 10,6 = 0
5. x^2 - 2x - 3 = 0
2
Choose the input format:
1. Console (default)
2. File
1
Enter the left border of the interval: -4
Enter the right border of the interval: 7
Enter the accuracy: 0.01
Enter the max iterations: 69
Choose the output format:
1. Console (default)
2. File
1
x =  0.7365335456910418
Iterations:  3
```



<u>Пример 2:</u>

```
Choose the method:
1. Chord method (default)
2. Secant method
3. Simple iteration method
4. Simple iteration method for systems
4
Choose the system of equations:
1. {x^2 + y^2 = 4; y = 3x^2} (default)
2. {0,1x^2 + x + 0,2y^2 = 0,3; 0,2x^2 + y + 0,1xy = 0,7}
2
Choose the input format:
1. Console (default)
2. File
1
Enter the left border of the interval: 0
Enter the right border of the interval: 1
Enter the left border of the interval: 0
Enter the right border of the interval: 1
Enter the initial approximation for x: 1
Enter the initial approximation for y: 1
Enter the accuracy: 0.01
Enter the max iterations: 35
Choose the output format:
1. Console (default)
2. File
1
x =  0.2033827465513719  y =  0.6773270394906847
Iterations:  5
x vector:  [1.0, 0.26800000000000007, 0.07318240000000004, 0.01244250379801598, 0.0038773572466440642]
y vector:  [0.6000000000000001, 0.30000000000000004, 0.03312479999999954, 0.012542117949695952, 0.002090278459011219]
Check: -0.0007264156041069669 -0.0006243888298468336
```



Вывод

В ходе выполнения лабораторной работы я узнала о различных численных методах решения нелинейных уравнений и систем из них, научилась решать их, написала для этого код на языке Python.