НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Факультет программной инженерии и компьютерной техники

Направление подготовки 09.03.04 - Программная инженерия

Дисциплина – Вычислительная математика

Лабораторная работа №2

Вариант №11

Выполнил: Мухсинов С.П

Группа: P3217

Преподаватель: Малышева Т.А

Санкт-Петербург

2024

Цель работы:

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

Задание:

1. Вычислительная реализация задачи:

1. Решение нелинейного уравнения:

1. Отделить корни заданного нелинейного уравнения графически

$$4{,}45x^3 + 7{,}81x^2 - 9{,}62x - 8{,}17 = 0$$

2. Определить интервалы изоляции корней

3. Уточнить корни нелинейного уравнения с точностью $\varepsilon = 10^{-2}$

   3.1 Уточнить крайний правый корень нелинейного уравнения методом половинного деления

   3.2 Уточнить крайний левый корень нелинейного уравнения методом хорд

   3.3 Уточнить центральный корень нелинейного уравнения методом простой итерации

4. Вычисления оформить в виде таблиц. Для всех значений в таблице удержать 3 знака после запятой.

2. Решение системы нелинейных уравнений

1. Отделить корни заданной системы нелинейных уравнений графически

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

2. Используя метод Ньютона, решить систему нелинейных уравнений с точностью до 0,01.

3. Подробные вычисления привести в отчете.

2. Программная реализация задачи:

Для нелинейных уравнений:

1. Все численные методы (см. табл. 9) должны быть реализованы в виде отдельных подпрограмм/методов/классов.
2. Пользователь выбирает уравнение, корень/корни которого требуется вычислить (3-5 функций, в том числе и трансцендентные), из тех, которые предлагает программа.

3. Предусмотреть ввод исходных данных (границы интервала/начальное приближение к корню и погрешность вычисления) из файла или с клавиатуры по выбору конечного пользователя.
4. 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.
5. Для методов, требующих начальное приближение к корню (методы Ньютона, секущих, хорд с фиксированным концом, простой итерации), выбор начального приближения (а или b) вычислять в программе.
6. Для метода простой итерации проверять достаточное условие сходимости метода на введенном интервале.
7. Предусмотреть вывод результатов (найденный корень уравнения, значение функции в корне, число итераций) в файл или на экран по выбору конечного пользователя.
8. Организовать вывод графика функции, график должен полностью отображать весь исследуемый интервал (с запасом).

Для систем нелинейных уравнений:

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

Выбор метода для программной реализации задачи:

Решение нелинейных уравнений: <u>Метод половинного деления</u>, <u>Метод секущих</u>, <u>Метод простой итерации</u>

Решение систем нелинейных уравнений: – <u>Метод простой итерации</u>

Используемые формулы и методы

Метод половинного деления

Идея метода: начальный интервал изоляции корня делим пополам, получаем начальное приближение к корню:

$$x_0 = \frac{a_0+b_0}{2}$$

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.

Рабочая формула метода: $\boxed{x_i = \frac{a_i + b_i}{2}}$

Приближенное значение корня: $x^* = \frac{a_n+b_n}{2}$ или $x^* = a_n$ или $x^* = b_n$

Критерии окончания итерационного процесса

Сходимость итерационного процесса фиксируется следующими способами:

1. Сходимость по аргументу: $|x_n - x_{n-1}| \leq \varepsilon$
2. Сходимость по функции: $|f(x_n)| \leq \varepsilon$

Для метода половинного деления можно рассматривать еще один критерий окончания итерационного процесса:
$|a_n - b_n| \leq \varepsilon$

Метод хорд

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 $B(b, f(b))$:

$$\frac{y-f(a)}{f(b)-f(a)} = \frac{x-a}{b-a}$$

Точка пересечения хорды с осью абсцисс ($y = 0$): $x = a - \frac{b-a}{f(b)-f(a)} f(a)$

Алгоритм метода:

0 шаг: Находим интервал изоляции корня $[a_0, b_0]$

1 шаг: Вычисляем x_0: $x_0 = a_0 - \frac{b_0-a_0}{f(b_0)-f(a_0)} f(a_0)$

2 шаг: Вычисляем $f(x_0)$.

3 шаг: В качестве нового интервала выбираем ту половину отрезка, на концах которого функция имеет разные знаки: $[a_0,x_0]$ либо $[b_0,x_0]$.

4 шаг: Вычисляем x_1 и т.д (повторяем 1-3 шаги).

Рабочая формула метода: $\boxed{x_i = \frac{a_i f(b_i) - b_i f(a_i)}{f(b_i) - f(a_i)}}$

Критерии окончания итерационного процесса: $|x_n - x_{n-1}| \leq \varepsilon$ или $|a_n - b_n| \leq \varepsilon$ или $|f(x_n)| \leq \varepsilon$

Приближенное значение корня: $x^* = x_n$

Метод секущих

Упростим метод Ньютона, заменив $f'(x)$ разностным приближением:

$$f'(x_i) \approx \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}}$$

Рабочая формула метода:

$$\boldsymbol{x_{i+1} = x_i - \frac{x_i - x_{i-1}}{f(x_i) - f(x_{i-1})} f(x_i)} \qquad i = 1,2 \ldots$$

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

Выбор x_0 определяется как и в методе Ньютона, x_1- выбирается рядом с начальным самостоятельно.

Критерий окончания итерационного процесса:

$$|x_n - x_{n-1}| \leq \varepsilon \text{ или } |\boldsymbol{f(x_n)}| \leq \boldsymbol{\varepsilon}$$

Приближенное значение корня: $x^* = x_n$

Метод простой итерации

Уравнение $f(x) = 0$ приведем к эквивалентному виду: $x = \varphi(x)$, выразив x из исходного уравнения.

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

$x_1 = \varphi(x_0) \rightarrow x_2 = \varphi(x_1) \ldots$

Рабочая формула метода: $\boldsymbol{x_{i+1} = \varphi(x_i)}$

Условия сходимости метода простой итерации определяются следующей теоремой.

Теорема. Если на отрезке локализации $[a, b]$ функция $\varphi(x)$ определена, непрерывна и дифференцируема и удовлетворяет неравенству:

$|\varphi'(x)| < q$, где $0 \leq q < 1$, то независимо от выбора начального приближения $x_0 \in [a, b]$ итерационная последовательность $\{x_n\}$ метода будет сходится к корню уравнения.

<u>Достаточное</u> условие сходимости метода:

$|\varphi'(x)| \leq q < 1$, где q – некоторая константа (коэффициент Липшица или коэффициент сжатия)

$q = \max\limits_{[a,b]} |\varphi'(x)|$

При $q \approx 0$ - скорость сходимости высокая,

При $q \approx 1$ - скорость сходимости низкая,

При $q > 1$ - нет сходимости.

Чем меньше q, тем выше скорость сходимости.

Критерий окончания итерационного процесса:

$|x_n - x_{n-1}| \leq \varepsilon$ (при $0 < q \leq 0{,}5$)

$|x_n - x_{n-1}| < \frac{1-q}{q} \varepsilon$ (при $0{,}5 < q < 1$)

Можно ограничиться: $|x_n - x_{n-1}| \leq \varepsilon$

РЕШЕНИЕ СИСТЕМЫ НЕЛИНЕЙНЫХ УРАВНЕНИЙ. МЕТОД НЬЮТОНА.

К основе метода лежит использование разложения функций $F_i(x_1, x_2, \dots, x_n)$ в ряд Тейлора в окрестности некоторой фиксированной точки, причем члены, содержащие вторые (и более высоких порядков) производные, отбрасываются.

Пусть начальные приближения неизвестных системы (1) получены и равны соответственно $a_1, a_2, \dots, a_n$. Задача состоит в нахождении приращений (поправок) к этим значениям $\Delta x_1, \Delta x_2, \dots, \Delta x_n$, благодаря которым решение системы запишется в виде

$$x_1 = a_1 + \Delta x_1, \ x_2 = a_2 + \Delta x_2, \dots, x_n = a_n + \Delta x_n \qquad (2)$$

Проведем разложение левых частей уравнений (1) с учетом (2) в ряд Тейлора, ограничиваясь лишь линейными членами относительно приращений:

$$\begin{cases} F_1(x_1, x_2, \dots, x_n) \approx F_1(a_1, a_2, \dots, a_n) + \dfrac{\partial F_1}{\partial x_1}\Delta x_1 + \cdots \dfrac{\partial F_1}{\partial x_n}\Delta x_n \\ F_2(x_1, x_2, \dots, x_n) \approx F_2(a_1, a_2, \dots, a_n) + \dfrac{\partial F_2}{\partial x_1}\Delta x_1 + \cdots \dfrac{\partial F_2}{\partial x_n}\Delta x_n \\ \dots\dots\dots\dots\dots\dots\dots\dots. \\ F_n(x_1, x_2, \dots, x_n) \approx F_n(a_1, a_2, \dots, a_n) + \dfrac{\partial F_n}{\partial x_1}\Delta x_1 + \cdots \dfrac{\partial F_n}{\partial x_n}\Delta x_n \end{cases}$$

РЕШЕНИЕ СИСТЕМЫ НЕЛИНЕЙНЫХ УРАВНЕНИЙ. МЕТОД НЬЮТОНА.

Поскольку в соответствии с (1) левые части этих выражений должны обращаться в нуль, то приравняем к нулю и правые части. Получим следующую систему линейных алгебраических уравнений относительно приращений:

$$\begin{cases} \dfrac{\partial F_1}{\partial x_1}\Delta x_1 + \dfrac{\partial F_1}{\partial x_2}\Delta x_2 + \cdots + \dfrac{\partial F_1}{\partial x_n}\Delta x_n = -F_1 \\ \dfrac{\partial F_2}{\partial x_1}\Delta x_1 + \dfrac{\partial F_2}{\partial x_2}\Delta x_2 + \cdots \dfrac{\partial F_2}{\partial x_n}\Delta x_n = -F_2 \\ \dots\dots\dots\dots\dots\dots\dots. \\ \dfrac{\partial F_n}{\partial x_1}\Delta x_1 + \dfrac{\partial F_n}{\partial x_2}\Delta x_2 + \cdots \dfrac{\partial F_n}{\partial x_n}\Delta x_n = -F_n \end{cases} \qquad (3)$$

Значения $F_1, F_2, \dots, F_n$ и их производные вычисляются при $x_1 = a_1,\ x_2 = a_2, \dots, x_n = a_n$.

Определителем системы (3) является ***якобиан***:

$$J = \begin{vmatrix} \dfrac{\partial F_1}{\partial x_1} & \dots\dots & \dfrac{\partial F_1}{\partial x_n} \\ \dfrac{\partial F_2}{\partial x_1} & \dots\dots & \dfrac{\partial F_2}{\partial x_n} \\ \dots\dots & \dots\dots & \dots\dots \\ \dfrac{\partial F_n}{\partial x_1} & \dots\dots & \dfrac{\partial F_n}{\partial x_n} \end{vmatrix}$$

РЕШЕНИЕ СИСТЕМЫ НЕЛИНЕЙНЫХ УРАВНЕНИЙ. МЕТОД НЬЮТОНА.

Итерационный процесс решения систем нелинейных уравнений методом Ньютона состоит в определении приращений $\Delta x_1, \Delta x_2, \dots, \Delta x_n$ к значениям неизвестных на каждой итерации.

Критерий окончания итерационного процесса: $max|\Delta x_i \leq \varepsilon|$.

В методе Ньютона:

1. Важен удачный выбор начального приближения для обеспечения хорошей сходимости.
2. Сходимость ухудшается с увеличением числа уравнений системы.

РЕШЕНИЕ СИСТЕМЫ НЕЛИНЕЙНЫХ УРАВНЕНИЙ. МЕТОД ПРОСТОЙ ИТЕРАЦИИ.

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: $\boldsymbol{X}^{(0)} = x_1^{(0)}, x_2^{(0)}, \dots, x_n^{(0)}$,
получим первые приближения к корням:

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

РЕШЕНИЕ СИСТЕМЫ НЕЛИНЕЙНЫХ УРАВНЕНИЙ. МЕТОД ПРОСТОЙ ИТЕРАЦИИ.

Последующие приближения находятся по формулам:

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

Сходятся ли эти последовательности?

Пусть задача отделения корней уже решена и определена достаточно малая область изоляции G, в которой находится подлежащий уточнению корень.

Пусть в этой окрестности функции $\varphi_i\left(x_1^k, x_2^k, \dots, x_n^k\right)$ дифференцируемы:

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

РЕШЕНИЕ СИСТЕМЫ НЕЛИНЕЙНЫХ УРАВНЕНИЙ. МЕТОД ПРОСТОЙ ИТЕРАЦИИ.

Достаточное условие сходимости итерационного процесса:

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

$$\begin{gathered} \left|\frac{\partial\varphi_1}{\partial x_1}\right| + \left|\frac{\partial\varphi_1}{\partial x_2}\right| + \dots + \left|\frac{\partial\varphi_1}{\partial x_n}\right| < 1 \\ \left|\frac{\partial\varphi_2}{\partial x_1}\right| + \left|\frac{\partial\varphi_2}{\partial x_2}\right| + \dots + \left|\frac{\partial\varphi_2}{\partial x_n}\right| < 1 \\ \dots\dots\dots \\ \left|\frac{\partial\varphi_n}{\partial x_1}\right| + \left|\frac{\partial\varphi_n}{\partial x_2}\right| + \dots + \left|\frac{\partial\varphi_n}{\partial x_n}\right| < 1 \end{gathered}$$

Если $\boldsymbol{X}^{(0)}$ и все последовательные приближения: $\boldsymbol{X}^{(k+1)} = \boldsymbol{\varphi}\left(\boldsymbol{X}^{k}\right)$, $k = 0, 1, 2 \dots$ принадлежат ограниченной замкнутой области G, тогда итерационный процесс сходится к единственному решению уравнения $\boldsymbol{X} = \boldsymbol{\varphi}(\boldsymbol{X})$

Критерий окончания итерационного процесса:

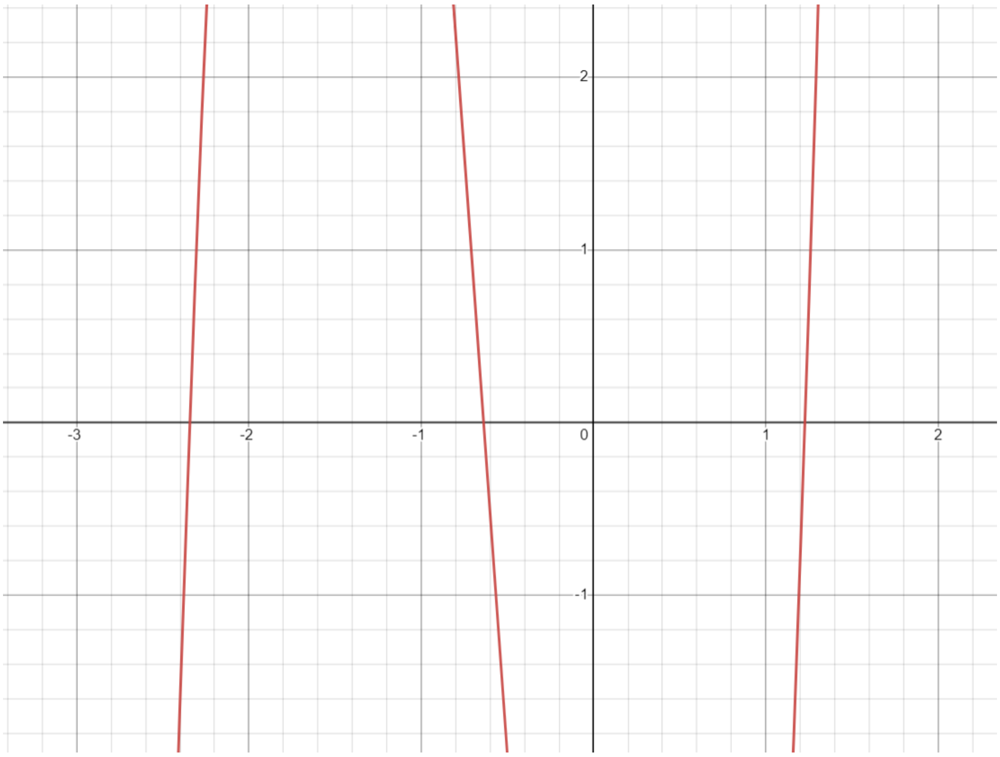
$$\max_{1<i<n}\left|x_i^{(k+1)} - x_i^k\right| \le \varepsilon$$

Вычислительная реализация

1 часть. Решение нелинейного уравнения.

Исследуемое уравнение:

$$4{,}45x^3 + 7{,}81x^2 - 9{,}62x - 8{,}17 = 0$$



Интервалы изоляции:

[-2.5, -2] – для крайнего левого корня

[-1; 0] – для центрального корня

[1; 2] – для крайнего правого корня

Уточнение корня уравнения методом половинного деления

№ шага	a	b	x	$f(a)$	$f(b)$	$f(x)$	$\|a-b\|$
1	1	2	1.5	-5.53	39.43	9.99125	1
2	1	1.5	1.25	-5.53	9.99125	0.69953125	0.5
3	1	1.25	1.125	-5.53	0.69953125	-2.7719335	0.25
4	1.125	1.25	1.1875	-2.7719335	0.69953125	-1.1286352	0.125
5	1.1875	1.25	1.21875	-1.1286352	0.69953125	-0.2380679	0.0625
6	1.21875	1.25	1.234375	-0.2380679	0.69953125	0.22480175	0.03125
7	1.21875	1.234375	1.2265625	-0.2380679	0.22480175	-0.0081092	0.015625
8	1.2265625	1.234375	1.23046875	-0.0081092	0.22480175	0.107976450	0.0078125

Уточнение корня уравнения методом хорд

№ шага	a	b	x	$f(a)$	$f(b)$	$f(x)$	$\|x_{k+1} - x_k\|$
1	-2.5	-2	-2.290507	-4.83875	6.71	1.36371675	0.209492369
2	-2.5	-2.290507	-2.336568	-4.83875	1.36371675	0.179954404	0.046060424
3	-2.5	-2.336568	-2.3424281	-4.83875	0.179954404	0.022287492	0.0058601375

Уточнение корня уравнения методом простой итерации

$$f(x) = 4{,}45x^3 + 7{,}81x^2 - 9{,}62x - 8{,}17$$

$$4{,}45x^3 + 7{,}81x^2 - 9{,}62x - 8{,}17 = 0$$

Преобразуем уравнение к виду $x = \varphi(x)$

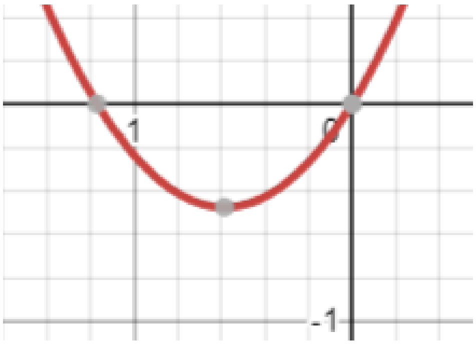
$$9.62x = 4{,}45x^3 + 7{,}81x^2 - 8{,}17$$

$$x = 0.4626x^3 + 0.8119x^2 - 0.8493$$

$$\varphi(x) = 0.4626x^3 + 0.8119x^2 - 0.8493$$

$$\varphi'(x) = 1.3878x^2 + 1.6238x$$

$|\varphi'(x)| < 1$ на отрезке $[-1; 0]$

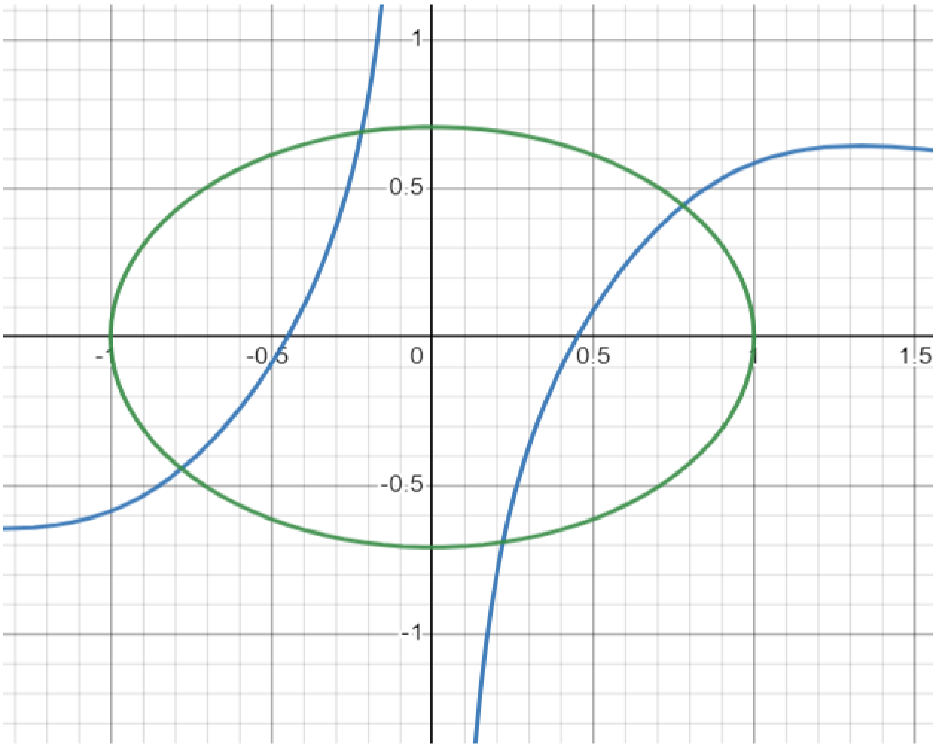


№ итерации	x_k	x_{k+1}	$\varphi(x_{k+1})$	$f(x_{k+1})$	$\|x_{k+1} - x_k\|$
1	-1.0	-0.5	-0.7042	-1.9638	0.5
2	-0.5	-0.7042	-0.6082	0.9234	0.2042
3	-0.7042	-0.6082	-0.6530	-0.4313	0.096
4	-0.6082	-0.6530	-0.6319	0.2030	0.0448
5	-0.6530	-0.6319	-0.6418	-0.0954	0.0211
6	-0.6319	-0.6418	-0.6372	0.0447	0.0099
7	-0.6418	-0.6372	-0.6393	-0.0203	0.0046

2 часть. Решение системы нелинейных уравнений.

Исследуемая система уравнений:

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



$$f(x, y) = tg(xy + 0.2) - x^2$$

$$g(x, y) = x^2 + 2y^2 - 1$$

Построим матрицу Якоби

$$\frac{\partial f}{\partial x} = \frac{y}{\cos^2(xy + 0.2)} - 2x$$

$$\frac{\partial f}{\partial y} = \frac{x}{\cos^2(xy + 0.2)}$$

$$\frac{\partial g}{\partial x} = 2x$$

$$\frac{\partial g}{\partial y} = 4y$$

$$\begin{cases} \left(\dfrac{y}{\cos^2(xy+0.2)} - 2x\right)\Delta x + \dfrac{x}{\cos^2(xy+0.2)}\Delta y = x^2 - tg(xy+0.2) \\ 2x\Delta x + 4y\Delta y = 1 - x^2 - 2y^2 \end{cases}$$

Найдем крайний правый корень методом Ньютона

Выбираем $x_0 = 0.75$, $y_0 = 0.5$ в качестве начального приближения.

На первой итерации система имеет вид:

$$\begin{cases} -0.7900185\Delta x + 1.0649722\Delta y = -0.0855454 \\ 1.5\Delta x + 2\Delta y = -0.0625 \end{cases}$$

Решаем полученную систему:

$$\Delta x = 0{,}032897050385$$

$$\Delta y = -0{,}055922787789$$

Вычисляем очередное приближение:

$$x_1 = x_0 + \Delta x = 0.782897050385$$

$$y_1 = y_0 + \Delta y = 0.444077212211$$

Проверяем критерий окончания итерационного процесса при $\varepsilon = 0.01$:

$$|x_1 - x_0| = 0.03289705038 > \varepsilon$$

$$|y_1 - y_0| = 0.0559227877 > \varepsilon$$

На следующей итерации система имеет вид:

$$\begin{cases} -0.956530\Delta x + 1.07411702\Delta y = 0.0030283 \\ 1.565794\Delta x + 1.77630884\Delta y = -0.00733693 \end{cases}$$

Решаем полученную систему:

$$\Delta x = -0{,}003921977356$$

$$\Delta y = -0{,}000673263157$$

Вычисляем очередное приближение:

$$x_2 = x_1 + \Delta x = 0.778975073029$$

$$y_2 = y_1 + \Delta y = 0.443403949054$$

Проверяем критерий окончания итерационного процесса при $\varepsilon = 0.01$:

$$|x_2 - x_1| = 0.003921977355999 < \varepsilon$$

$$|y_2 - y_1| = 0.00067326315700000036 < \varepsilon$$

Завершаем итерационный процесс.

Найденный ответ:

$$x = 0.778975073029$$

$$y = 0.443403949054$$

Программная реализация

Метод половинного деления:

```
    @Override
    public Solution solve() {
        double a = start;
        double b = end;
        double x;
        int n = 0;
        while (n < MAX_ITERATION){
            x = (a + b) / 2;
            if(f.at(a)*f.at(x) > 0){
                a = x;
            }else{
                b = x;
            }
            n++;
            if( (Math.abs(a - b) <= eps) & (Math.abs(f.at(x)) <= eps)){
                x = (a+b) / 2;
                return new Solution(x, n, Math.abs(f.at(x)));
            }
        }
        throw new IllegalArgumentException("Не удалось достичь указанной
точности");
    }

}
```

Метод секущих:

```
    @Override
    public Solution solve() {
        double x0 = end;
        double x1 = (start + end) / 2;
        int n = 0;
        double x2;
        while (n < MAX_ITERATION){
            x2 = x1 - ((x1 - x0)/(f.at(x1) - f.at(x0))) * f.at(x1);
            if((Math.abs(f.at(x2)) <= eps)){
                return new Solution(x2, n, Math.abs(f.at(x2)));
            }
            n++;
            x0 = x1;
            x1 = x2;
        }
```

```
        throw new IllegalArgumentException("Не удалось достичь указанной точности");
    }
}
```

Метод простой итерации:

```
    public Iteration(double start, double end, double eps, MathFunction f) {
        super(start, end, eps, f);
        double fS = f.derivativeAt(start);
        double fE = f.derivativeAt(end);
        double m = Math.max(Math.abs(fS), Math.abs(fE));
        if(Math.abs(fS) > Math.abs(fE)){
            if(fS > 0){
                phi = new MathFunction(x -> (x + (-1/m) * f.at(x)));
            }else{
                phi = new MathFunction(x -> (x + (1/m) * f.at(x)));
            }
        }else{
            if(fE > 0){
                phi = new MathFunction(x -> (x + (-1/m) * f.at(x)));
            }else{
                phi = new MathFunction(x -> (x + (1/m) * f.at(x)));
            }
        }
    }

    @Override
    public Solution solve() throws IllegalIntervalException {
        double x0 = f.derivativeSecondAt(start)*f.at(start) > 0?start : end;
        int n = 0;
        double xn = x0;
        double x_next;
        System.out.println("Вывод значения производных функции: ");
        System.out.println(phi.derivativeAt(start) + " " + phi.derivativeAt(end));
        while(n < MAX_ITERATION){
            if(Math.abs(phi.derivativeAt(xn)) > 1){
                System.out.println(CONVERGENCE_EXCEPTION.getMessage());
            }
            x_next = phi.at(xn);
            n++;
            if((Math.abs(x_next - xn) < eps) && Math.abs(f.at(x_next)) < eps){
                return new Solution(x_next, n, Math.abs(f.at(x_next)));
            }
            xn = x_next;
        }
        throw new IllegalArgumentException("Не удалось достичь указанной точности");
    }


}
```

Метод простой итерации для системы:

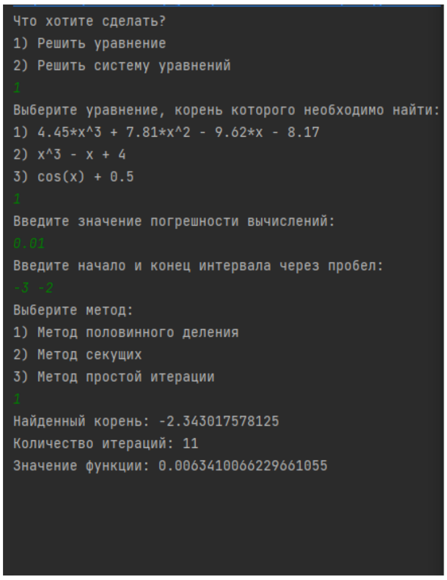
```
public SystemIteration(double eps, double x0, double y0, boolean system) {
    this.eps = eps;
    this.x = x0;
    this.y = y0;
```

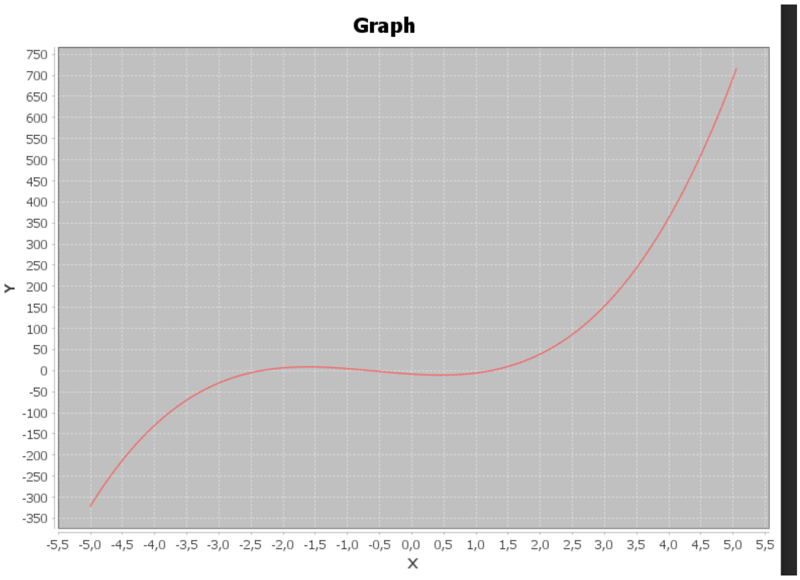
```java
    if(!system){
        f1 = new MathBinaryFunction((x,y)->(Math.pow(x, 3) + Math.pow(y, 3) +
3) / 6);
        f2 = new MathBinaryFunction((x,y)->(Math.pow(x, 3) - Math.pow(y, 3) +
2) / 6);
    }else {
        f1 = new MathBinaryFunction((x,y)->(Math.cbrt(y)));
        f2 = new MathBinaryFunction((x,y)->(x*x - 1));
    }
}

public SystemSolution solve(){
    if(!ifConvergence(x,y)){
        System.out.println(CONVERGENCE_EXCEPTION.getMessage());
    }
    int n = 0;
    double xn;
    double yn;
    while (n < MAX_ITERATION){
        xn = f1.at(x,y);
        yn = f2.at(x,y);
        n++;
        if(Math.abs(xn - x) < eps && Math.abs(yn - y) < eps){
            return new SystemSolution(xn, yn, n, f.at(xn,yn), g.at(xn,yn));
        }
        x = xn;
        y = yn;
    }
    throw new IllegalArgumentException("Не удалось достичь указанной
точности");
}
```

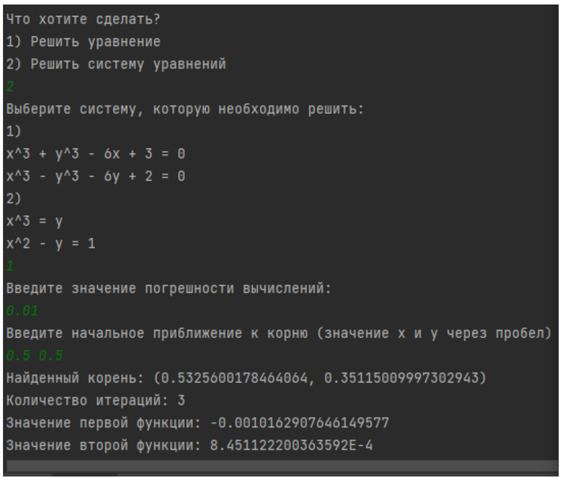
Примеры работы программы

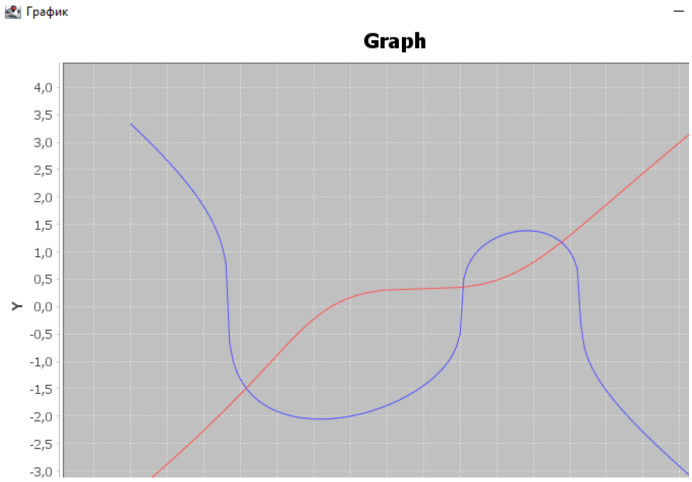
№1





№2





Вывод

В ходе выполнения лабораторной работы я познакомился с несколькими способами решения нелинейных уравнений и систем нелинейных уравнений. Была реализована программа, которая вычисляет приближенное решение нелинейного уравнения и системы нелинейных уравнений, используя методы секущих, половинного деления, Ньютона и простых итераций.