

Факультет программной инженерии и компьютерной техники

Дисциплина: Вычислительная математика

Лабораторная работа №2

«Численное решение нелинейных уравнений и систем»

Вариант № 2

Выполнил: Васильев А. Ю.

№ группы: P3215

Преподаватель: Малышева Т. А.

2022 год, 4 семестр

**Цель работы.**

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

**Задание лабораторной работы**

1. № варианта определяется как номер в списке группы согласно ИСУ. .

2. Отделить корни заданного нелинейного уравнения графически (см. табл. 5)

3. Определить интервалы изоляции корней.

4. Вычислительная реализация задачи (в отчет):

Уточнить корни нелинейного уравнения (см. табл.5) с точностью ε=10-2. Вы-числения оформить в виде таблиц (1-4), удержать 3 знака после запятой.

Представить в отчете заполненные таблицы (1-4). В таблице 6 указаны методы для каждого из 3-х корней многочлена.

4.1 Для метода половинного деления или метода хорд заполнить таблицу 1.

4.2 Для метода Ньютона или метода секущих заполнить таблицу 2.

4.3 Для метода секущих заполнить таблицу 3.

4.4 Для метода простой итерации заполнить таблицу 4.

**Для прямых методов должно быть реализовано:**

1. Вычисление определителя
2. Вывод треугольной матрицы (включая преобразованный столбец В)
3. Вывод вектора неизвестных: x1, x2, …, xn
4. Вывод вектора невязок: r1, r, …, rn
5. Программная реализация задачи:
6. Для нелинейных уравнений:
7. Все численные методы (см. табл. 7) должны быть реализованы в виде отдельных подпрограмм или классов.
8. Пользователь выбирает уравнение, корень/корни которого требуется вычислить (3-5 функций, в том числе и трансцендентные), из тех, которые предлагает программа.
9. Предусмотреть ввод исходных данных (границы интервала/начальное приближение к корню и погрешность вычисления) из файла или с клавиатуры по выбору конечного пользователя.
10. Выполнить верификацию исходных данных. Для метода половинного деления (метода хорд) анализировать наличие корня на введенном интервале. Для метода Ньютона (метода секущих) – выбор начального приближения (а или b). Для метода простой итерации – достаточное условие сходимости метода. Программа должна реагировать на некорректные введенные данные.
11. Предусмотреть вывод результатов (найденный корень уравнения, значение функции в корне, число итераций) в файл или на экран по выбору конечного пользователя.
12. Организовать вывод графика функции, график должен полностью отображать весь исследуемый интервал (с запасом).
13. Для систем нелинейных уравнений:
14. Рассмотреть систему двух уравнений.
15. Организовать вывод графика функций.
16. Для метода простой итерации проверить достаточное условие сходимости.
17. Вывод вектора неизвестных: x\_1,x\_2
18. Вывод количества итераций, за которое было найдено решение.
19. Вывод вектора погрешностей: 〖|x〗\_i^((k))-x\_i^((k-1) ) |

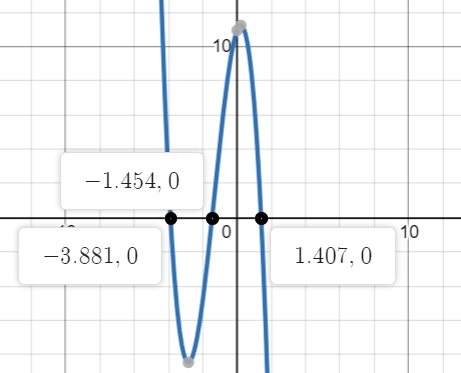
**Листинг программы:**

DataManager – класс, в котором происходит считывание данных с консоли и из файла

calculating.py – файл, в котором реализованы методы решения нелинейных урав-й

**Численные методы для функции**

|  |
| --- |
|  |



**Метод хорд**(крайний левый корень)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| № шага | a | b | x | f(a) | f(b) | f(x) | |a-b| |
| 1 | -4 | -3.5 | -3.5 | 2.27 | -5.272 | -5.272 | 0.02 |
| 2 | -4 | -3.85 | -3.85 | 2.27 | -0.539 | -0.539 | 0.35 |
| 3 | -4 | -3.878 | -3.878 | 2.27 | -0.038 | -0.038 | 0.029 |
| 4 | -4 | -3.880 | -3.880 | 2.27 | -0.003 | -0.003 | 0.002 |

Левый корень - -3.880

**Метод половинного деления**(центральный корень)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| № шага | a | B | x | f(a) | f(b) | f(x) | |a-b| |
| 1 | -2 | -1 | -1.5 | -4.83 | 4.34 | -0.4425 | 1 |
| 2 | -1.5 | -1 | -1.25 | -0.4425 | 4.34 | 1.96406 | -0.5 |
| 3 | -1.5 | -1.25 | -1.375 | -0.4425 | 1.96406 | 0.75652 | 0.25 |
| 4 | -1.5 | -1.375 | -1.4375 | -0.4425 | 0.75652 | 0.15494 | 0.125 |
| 5 | -1.5 | -1.4375 | -1.46875 | -0.4425 | 0.15494 | -0.14443 | 0.0625 |
| 6 | -1.46875 | -1.4375 | -1.45313 | -0.14443 | 0.15494 | 0.00511 | 0.03125 |
| 7 | -1.46875 | -1.45313 | -1.46094 | -0.14443 | 0.00511 | -0.06970 | 0.01563 |
| 8 | -1.46094 | -1.45313 | -1.45703 | -0.06970 | 0.00511 | -0.03230 | 0.00781 |

Центральный корень - -1.45703

**Метод простой итерации**(крайний правый корень)

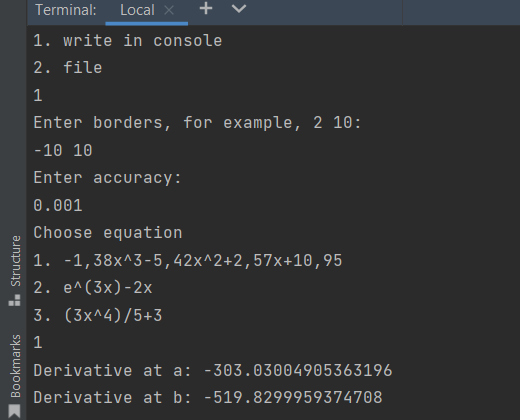
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № итерации | *xk* | *f*(*xk* ) | *xk*+1 |  | │*xk* − *xk*+1│ |
| 1 | 1.6 | -4.46568 | 1.3277 | 1.3277 | 0.2723 |
| 2 | 1.3277 | 1.5781 | 1.40172 | 1.42393 | 0.09623 |
| 3 | 1.42393 | -0.3641 | 1.40172 | 1.40172 | 0.0222 |
| 4 | 1.40172 | 0.10233 | 1.40796 | 1.40796 | 0.00624 |

Интервал изоляции [1.2;1.6]

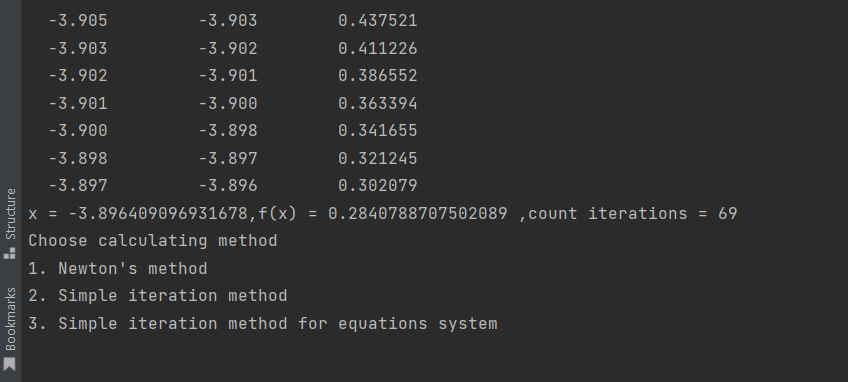
Правый корень - 1.40796

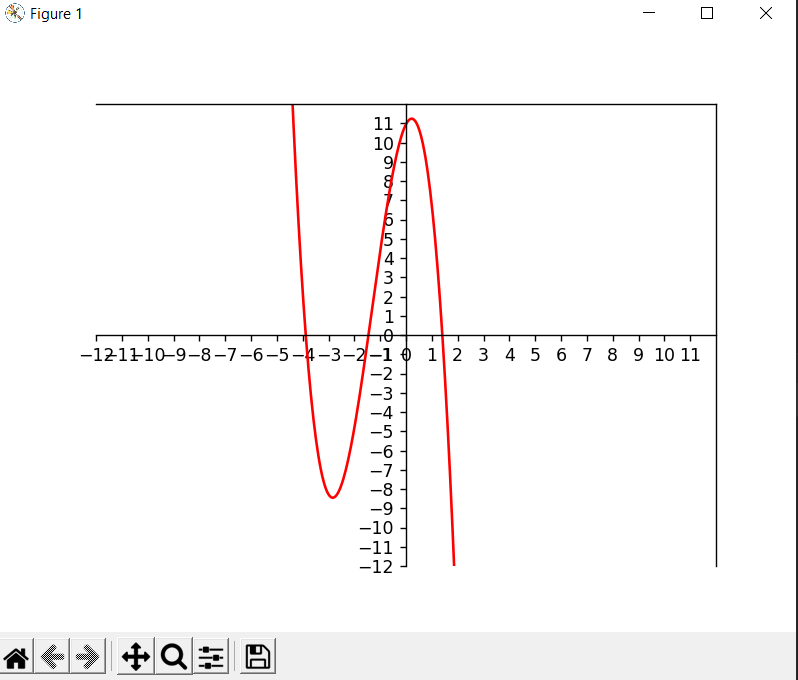
**Примеры и результаты работы программы.**

Входные данные:

****

Результат:





**Исходный код.**

[**https://github.com/wizarsi/calculative-math-lab2**](https://github.com/wizarsi/calculative-math-lab2)

**Выводы по работе.**

Изучил численный методы решения нелинейных уравнений. Реализовал метод простой итерации, метод Ньютона, метод простой итерации для системы из 2-х уравнений в программе на языке Python.