|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **Министерство науки и высшего образования Российской Федерации**  Калужский филиал  федерального государственного бюджетного  образовательного учреждения высшего образования  ***«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»***  ***(КФ МГТУ им. Н.Э. Баумана)*** |

**ФАКУЛЬТЕТ** ***ИУК «Информатика и управление»***

**КАФЕДРА** \_\_***ИУК4 «Программное обеспечение ЭВМ, информационные технологии»***

**ЛАБОРАТОРНАЯ РАБОТА №3**

**«Решение нелинейных уравнений»**

**ДИСЦИПЛИНА: «Вычислительные алгоритмы»**

|  |  |  |
| --- | --- | --- |
| Выполнил: студент гр. ИУК4-42Б | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ( Карельский М.К. )  (Подпись) |
| Проверил: | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ( Никитенко У.В. )  (Подпись) |
| Дата сдачи (защиты):  Результаты сдачи (защиты): | | |
|  | - Балльная оценка:  - Оценка: | |

Калуга, 2022

**Цель:** сформировать практические навыки описания и анализа используемых алгоритмов, создания программной реализации системы с заданными свойствами.

**Задачи:** изучение и исследование методов решения нелинейных уравнений и их систем, отделение корней, поиск корней методами с различной степенью сходимости.

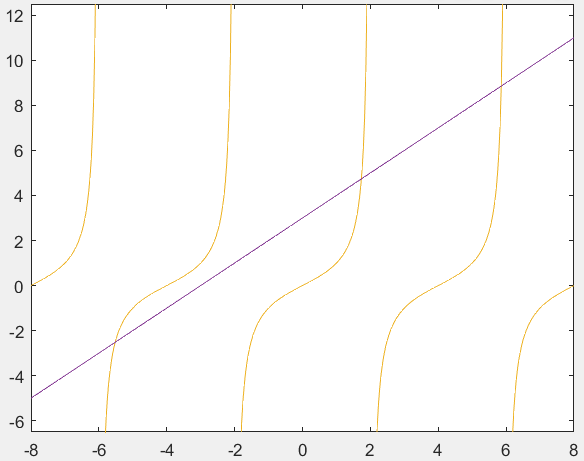
**Вариант №12**

Требуется:

1. Отделить все корни
2. Вычислить корни методом Ньютона с точностью ε = 0.000001
3. Используя интервалы из второго пункта, найти требуемые корни с точностью ε = 0.001 методом хорд
4. Сравнить результаты и количество итераций

**Решение:**

Корни уравнения можно найти как абсциссы точек пересечения графиков:

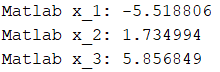


**Рис. 1.** Точки пересечения в интервале [-8, 8]

В интервале [-8, 8] находится 3 корня:

1. В диапазоне [-5.6, -5.5]
2. В диапазоне [1.7, 1.8]
3. В диапазоне [5.8, 5.9]

Найдем корни, используя встроенные средства Matlab:



**Рис. 2.** Корни, найденные встроенными средствами Matlab

Найдем корни, используя метод Ньютона:

|  |  |  |  |
| --- | --- | --- | --- |
| **Диапазон** | **[-5.6, -5.5]** | | |
| k | xk | xk – xk-1 | f(xk) |
| 0 | -5.533883 | 0.066117 | -0.074570 |
| 1 | -5.519345 | 0.014538 | -0.002574 |
| 2 | -5.518806 | 0.000538 | -0.000003 |
| 3 | -5.518806 | 0.000001 | 0 |
|  | **[1.7, 1.8]** | | |
| 0 | 1.739868 | 0.039868 | 0.086426 |
| 1 | 1.735088 | -0.004780 | 0.001623 |
| 2 | 1.734994 | -0.000093 | 0.000001 |
|  | **[5.8, 5.9]** | | |
| 0 | 5.879959 | 0.079959 | 1.695292 |
| 1 | 5.860611 | -0.019348 | 0.237296 |
| 2 | 5.856949 | -0.003662 | 0.006144 |
| 3 | 5.856849 | -0.000100 | 0.000004 |
| 4 | 5.856849 | 0 | 0 |

**Табл. 1.** Корни, найденные методом Ньютона

Найдем корни, используя методом хорд:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Диапазон** | **[-5.6, -5.5]** | | | |
| k | xk | xk – xk-1 | xk – x\* | f(xk) |
| 0 | -5.515225 | -0.015225 | 0.003581 | 0.016929 |
| 1 | -5.518126 | -0.002902 | 0.000679 | 0.003234 |
| 2 | -5.518677 | -0.000550 | 0.000129 | 0.000614 |
|  | **[1.7, 1.8]** | | | |
| 0 | 1.726103 | 0.026103 | -0.008892 | -0.149430 |
| 1 | 1.732742 | 0.006639 | -0.002252 | -0.038832 |
| 2 | 1.734424 | 0.001682 | -0.000570 | -0.009893 |
| 3 | 1.734850 | 0.000426 | -0.000144 | -0.002508 |
|  | **[5.8, 5.9]** | | | |
| 0 | 5.839512 | 0.039512 | -0.017337 | -0.948042 |
| 1 | 5.851574 | 0.012062 | -0.005275 | -0.312214 |
| 2 | 5.855245 | 0.003671 | -0.001604 | -0.097367 |
| 3 | 5.856361 | 0.001116 | -0.000488 | -0.029831 |
| 4 | 5.856700 | 0.000339 | -0.000148 | -0.009089 |

**Табл. 2.** Корни, найденные методом хорд

Таким образом, метод Ньютона оказался более точным, чем метод хорд. Среднее количество итераций оказалось одинаковым.

**Вывод:** в ходе работы были получены практические навыки описания и анализа используемых алгоритмов, создания программной реализации системы с заданными свойствами.

**ПРИЛОЖЕНИЯ**

**Листинг:**

ezplot ('tan(pi \* x / 4)', [-8 8]);

hold on

ezplot ('x + 3', [-8 8]);

hold on

syms x;

x\_1 = fzero("tan(pi \* x / 4) - x - 3", -5.5);

fprintf("Matlab x\_1: %f\n", x\_1);

x\_2 = fzero("tan(pi \* x / 4) - x - 3", 1.7);

fprintf("Matlab x\_2: %f\n", x\_2);

x\_3 = fzero("tan(pi \* x / 4) - x - 3", 5.8);

fprintf("Matlab x\_3: %f\n", x\_3);

fprintf("\n");

f = inline('tan(pi \* x / 4) - x - 3');

df = inline('pi / (4 \* cos(pi \* x / 4)^2) - 1');

x\_1 = newton(f, df, -5.6);

x\_2 = newton(f, df, 1.7);

x\_3 = newton(f, df, 5.8);

chord(f, -5.5, -5.6, 0.001, x\_1);

chord(f, 1.7, 1.8, 0.001, x\_2);

chord(f, 5.8, 5.9, 0.001, x\_3);

function [root] = newton(f, df, x0)

eps = 1e-6;

root = x0;

y0 = f(x0);

k = 0;

fprintf("Newton method table\n");

while abs(y0) > eps

old\_root = root;

root = old\_root - f(old\_root) / df(old\_root);

y0 = f(root);

fprintf("%i %f %f %f %f", k, root, root - old\_root, f(root));

fprintf("\n");

k = k + 1;

end

fprintf("\n");

end

function [root] = chord(f, a, b, eps, x\_i)

root = a;

old\_root = b;

k = 0;

fprintf("Chord method table\n");

while abs(root - old\_root) > eps

old\_root = root;

root = old\_root - (old\_root - b) \* f(old\_root) / (f(old\_root) - f(b));

fprintf("%i %f %f %f %f\n", k, root, root - old\_root, root - x\_i, f(root));

k = k + 1;

end

fprintf("\n");

end