МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский Авиационный Институт»

(Национальный Исследовательский Университет)

Институт№8: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

**КУРСОВОЙ ПРОЕКТ**

По курсу «Вычислительные системы»

I семестр

Тема:

«Процедуры и функции в качестве параметров»

|  |  |
| --- | --- |
| Группа: | М8О-106Б-22 |
| Студент: | Каримов А.А. |
| Преподаватель: | Дубинин А.В. |
| Оценка: |  |
| Дата: |  |

Москва, 2022

Оглавление

[Постановка задачи 2](#_Toc123844322)

[Численные методы 2](#_Toc123844323)

[Практическая часть 4](#_Toc123844324)

[Код программы 9](#_Toc123844325)

[Заключение 12](#_Toc123844326)

[Источники 12](#_Toc123844327)

# Постановка задачи

Составить программу на языке Си с процедурами решения трансцендентных алгебраических уравнений различными численными методами (итераций, Ньютона и половинного деления – дихотомии). Применить каждую процедуру к решению двух уравнений, заданных двумя строками таблицы, начиная с варианта с заданным номером. Дать графическое обоснование применимости.

# Численные методы

Метод дихотомии

Теорема:

Если непрерывная функция на концах некоторого интервала имеет значения разных знаков, то внутри этого интервала у нее есть хотя бы один корень.

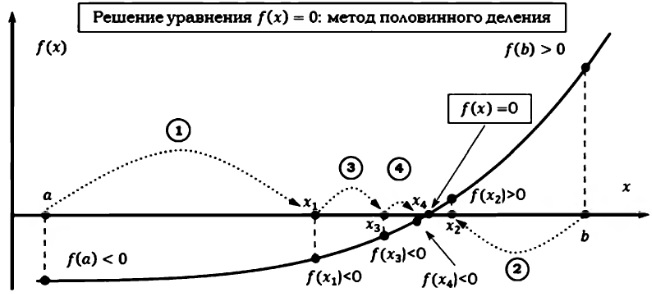
На этой теореме основан метод дихотомии. Алгоритм выглядит так:

1. Задается начальный отрезок [a, b]
2. Проверка разного знака на концах отрезка: f(a) \* f(b) < 0
3. a(k) = a(k-1) ; b(k) = (a(k-1) + b(k-1)) / 2 , если f((a(k-1) + b(k-1)) / 2) \* f(a(k-1)) < 0, иначе a(k) = (a(k-1) + b(k-1)) / 2 ; b(k) = b(k-1)
4. Повторять преобразование, пока результат не достигнет нужной точности, то есть пока не будет выполнятся b(k) – a(k) < ε

Тогда по окончанию выполнения алгоритма будет получено приближенное значение корня:

x\* ≈ (a(конечное) + b(конечное)) / 2

Рисунок 1 – Метод дихотомии



Метод итераций

Для реализации метода требуется привести функцию к виду . Для сходимости метода достаточно, чтобы

Так как функция может быть выражена неоднозначно, следует применить, в случае расхождения, вышеописанное условие ко всем вариантам выражения переменной.

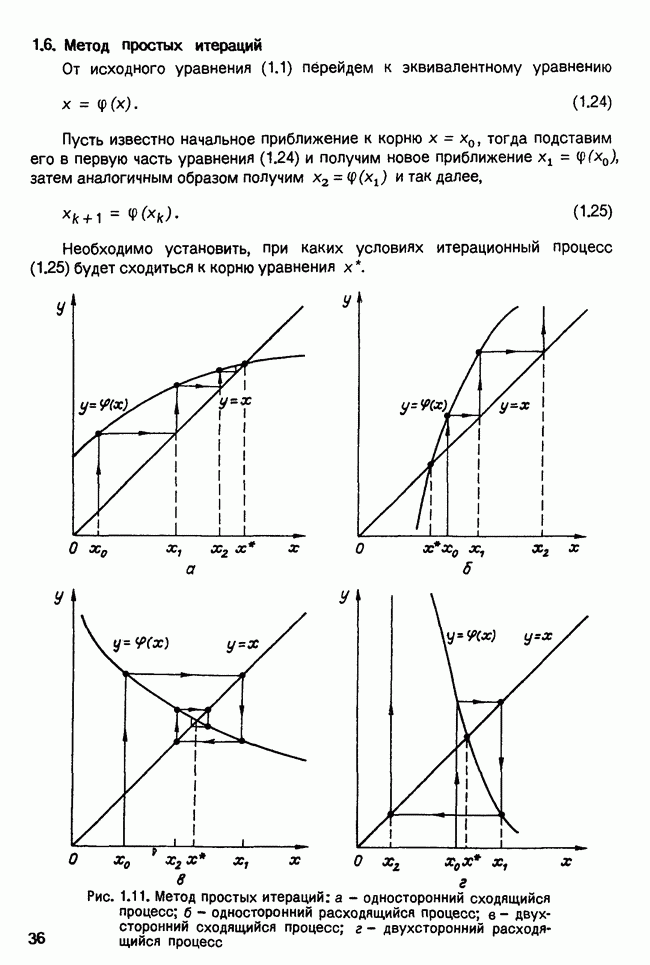
Начальное приближение корня: x(0) = (a + b) / 2 (середина [a, b])

Итерационный процесс: x(k+1) = f(x(k))

Условием выхода из процесса преобразований будет:

*Так, приближенное значение корня: x\** ≈ *x(конечное)*

Рисунок 2 - Метод итераций



Метод Ньютона

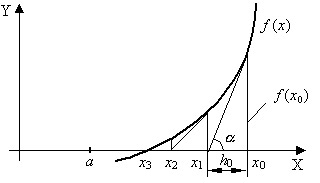
Метод Ньютона является частным случаем метода итераций. Его условие сходимости:

Итерационный процесс:

Условие выхода:

Таким образом, корень: *x\** ≈ *x(конечное)*

Рисунок 3 - Метод Ньютона



# Практическая часть

Вариант 15, 16

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Уравнение | Отрезок, содержащий корень | Базовый метод | Приближенное значение корня |
| 15 |  |  | Итераций | 1.2388 |
| 16 |  |  | Итераций | 2.2985 |

Графики функций

*Функция из 15-го варианта*

Функция

Рисунок 4 - График функции на [1;2]

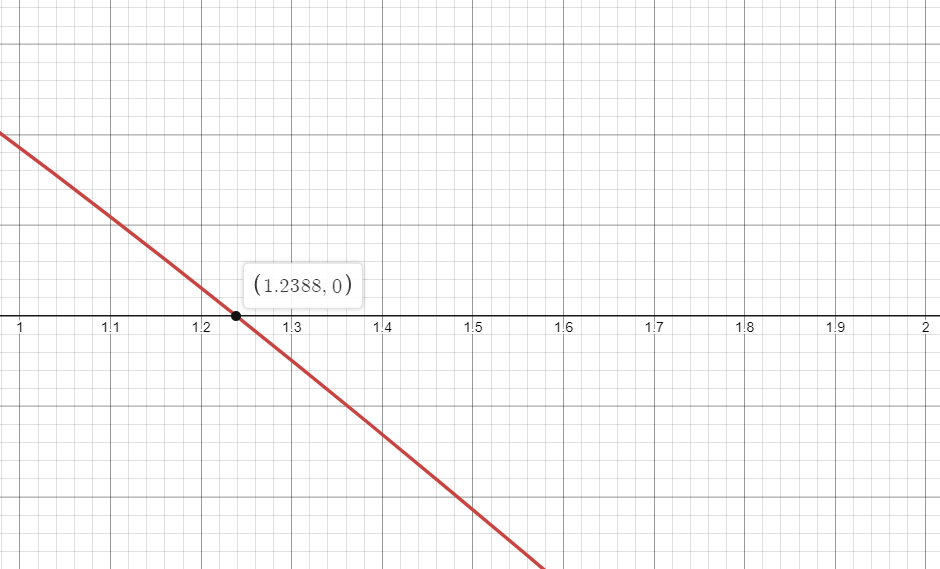
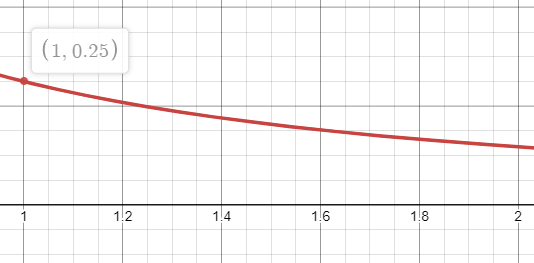


График производной

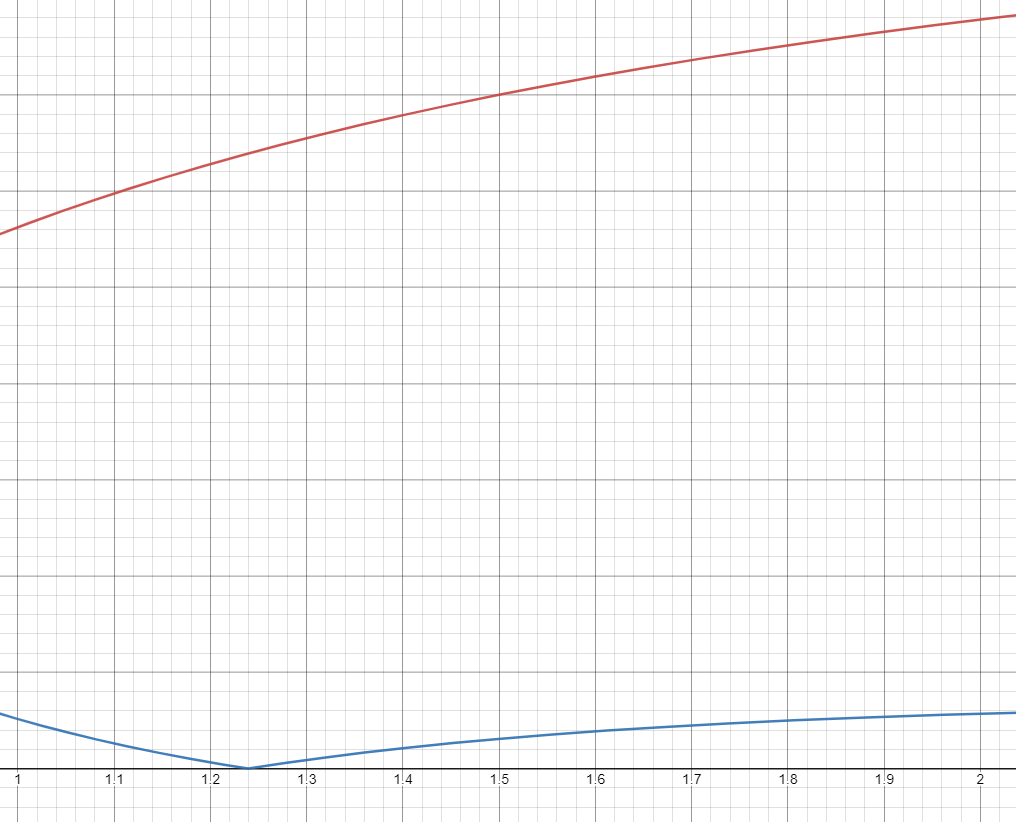
Рисунок 5 - График производной функции



Как видно из графика, на отрезке [1,2] производная преобразованной функции всегда меньше 1, при любом *x* из отрезка. Таким образом, применимость метода итераций доказана.

Графики второй производной функции, самой функции и её первой производной в квадрате , где левая часть – синяя линяя, а правая часть – красная.

Рисунок 6 - Сравнение графиков для метода Ньютона



Из графика видно, что метод Ньютона применим к данному уравнению на в каждой точке отрезка [1;2].

*Функция из 16-го варианта*

Функция F(x) =

Рисунок 7 - График функции на отрезке [2;3]

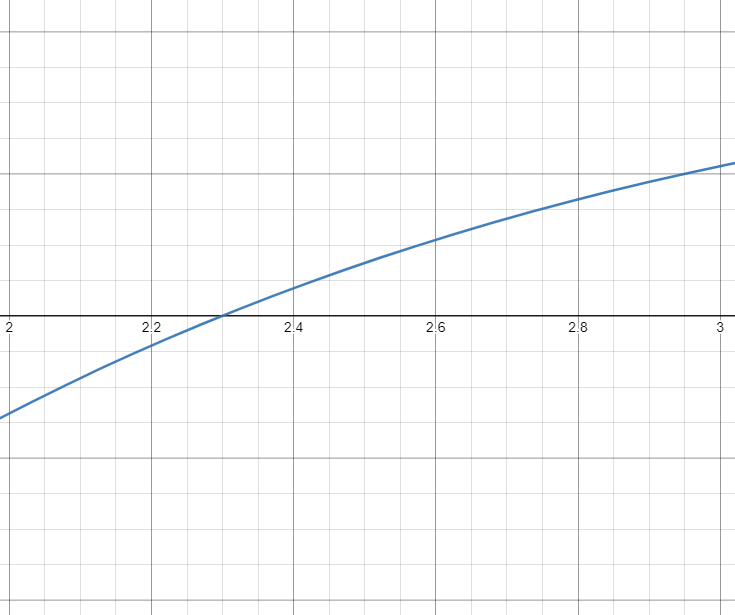
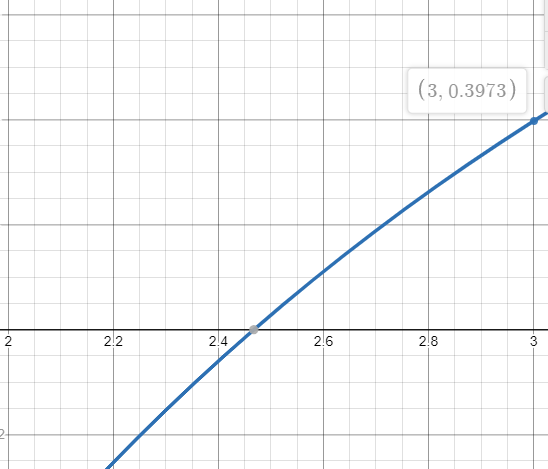


График производной функции :

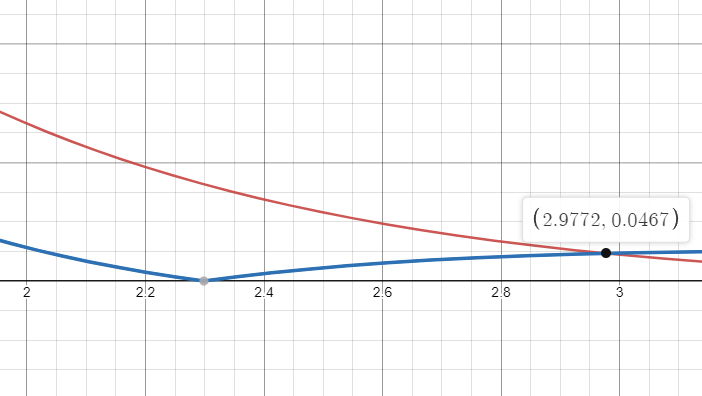
Рисунок 8 - График производной функции



Из графика видно, что наибольшее значение производной функции на отрезке [2;3] достигается в точке 3, причем само значение равно ≈0.4, поэтому метод итераций применим к данной функции на заданном отрезке.

Графики второй производной функции, самой функции и её первой производной в квадрате , где левая часть – синяя линяя, а правая часть – красная.

Рисунок 9 - Сравнение графиков для метода Ньютона



Из графика видно, что условие применимости метода Ньютона выполняется не на всем отрезке, тем не менее, так как начальное приближение корня равно 2.2985, можно утверждать о применимости метода, потому что корень все равно лежит в области, где условие выполняется.

# Код программы

#include <stdio.h>

#include <math.h>

#include <stdbool.h>

const double DBL\_EPSILON = 2.220446e-16;

double func(double x) {           // F(x) = 0

    return 0.4 + atan(sqrt(x)) - x;

}

double f(double x) {              // x = f(x)

    return 0.4 + atan(sqrt(x));

}

double func2(double x) {           // F(x) = 0

    return 3 \* sin(sqrt(x)) + 0.35 \* x - 3.8;

}

double f2(double x) {              // x = f(x)

    return (3.8 - 3 \* sin(sqrt(x))) / 0.35;

}

double der(double (\*f)(double), double x, double eps) {      // Производная

    return (f(x + eps) - f(x - eps)) / (2 \* eps);

}

double der2(double (\*f)(double), double x) {                // Вторая производная через предел с википедии

    double dx = pow(2, -18);

    return (func(x + dx) - 2 \* func(x) + func(x - dx)) / (dx \* dx);

}

typedef struct {

    double root;

    int i;

    bool is\_success;

} return\_helper;

return\_helper dichotomy(double (\*f)(double), double a, double b, double eps) {

    return\_helper res;

    double root, c;

    int i = 0;

    if (f(a) \* f(b) > 0) {

        res.is\_success = false;

        return res;

    } else {

        res.is\_success = true;

    }

    while (b - a > eps) {

        c = (a + b) / 2;

        if (f(a) \* f(c) <= 0) {

            b = c;

        } else {

            a = c;

        }

        i++;

    }

    res.i = i;

    root = (a + b) / 2;

    res.root = root;

    return res;

}

return\_helper iter(double (\*f)(double), double a, double b, double eps) {

    return\_helper res;

    double x0 = (a + b) / 2, x = f(x0);

    int i = 0;

    while(fabs(x - x0) > eps) {

        if (fabs(der(f, x, eps)) >= 1) {

            res.is\_success = false;

            return res;

        }

        x0 = x;

        x = f(x0);

        i++;

    }

    res.i = i;

    res.root = x;

    res.is\_success = true;

    return res;

}

return\_helper newton(double (\*f)(double), double a, double b, double eps) {

    return\_helper res;

    double x0, x = (a + b) / 2;

    int i = 0;

    do {

        x0 = x;

        if(fabs(f(x0) \* der2(f, x0)) >= der(f, x0, eps) \* der(f, x0, eps)) {

            res.is\_success = false;

            return res;

        }

        x = x0 - (f(x0) / der(f, x0, eps));

        i++;

    } while (fabs(x - x0) > eps);

    res.i = i;

    res.root = x;

    res.is\_success = true;

    return res;

}

int main() {

    double a = 1, b = 2;

    double eps = 65536 \* DBL\_EPSILON;

    printf("\n0.4 + atan(sqrt(x)) - x = 0    [1, 2]    x = 1.2388\n");

    return\_helper d1 = dichotomy(func, a, b, eps);

    printf("Dichotomy: root = %f iter\_num = %d is\_success = %d\n", d1.root, d1.i, d1.is\_success);

    return\_helper i1 = iter(f, a, b, eps);

    printf("Iterations: root = %f iter\_num = %d is\_success = %d\n", i1.root, i1.i, i1.is\_success);

    return\_helper n1 = newton(func, a, b, eps);

    printf("Newton: root = %f iter\_num = %d is\_success = %d\n", n1.root, n1.i, n1.is\_success);

    double a2 = 2, b2 = 3;

    printf("\n3sin(sqrt(x)) + 0.35x - 3.8 = 0    [2, 3]    x = 2.2985\n");

    return\_helper d2 = dichotomy(func2, a2, b2, eps);

    printf("Dichotomy: root = %f iter\_num = %d is\_success = %d\n", d2.root, d2.i, d2.is\_success);

    return\_helper i2 = iter(f2, a2, b2, eps);

    printf("Iterations: root = %f iter\_num = %d is\_success = %d\n", i2.root, i2.i, i2.is\_success);

    return\_helper n2 = newton(func2, a2, b2, eps);

    printf("Newton: root = %f iter\_num = %d is\_success = %d\n", n2.root, n2.i, n2.is\_success);

}

Выходные данные

|  |  |  |  |
| --- | --- | --- | --- |
|  | Root | Iter\_num | Is\_success |
| Dichotomy | 1.238840 | 37 | 1 |
| Iterations | 1.238840 | 15 | 1 |
| Newton | 1.238840 | 4 | 1 |
|  |  |  |  |
| Dichotomy | 2.298536 | 37 | 1 |
| Iterations | 2.298536 | 13 | 1 |
| Newton | 2.298536 | 5 | 1 |

# Заключение

В ходе выполнения курсового проекта я научился применять три численных метода для решения трансцендентных уравнений, а именно методы дихотомии, итерация и Ньютона. Я увидел, что метод Ньютона является самым быстрым по количеству итераций для получения корня в рамках заданной погрешности, тем не менее нельзя игнорировать его требования вычисления второй производной функции.

# Источники

1. Сайт построения графиков

[www.desmos.com](http://www.desmos.com)

1. Информационный портал, «Вычисление второй производной по одной переменной»

<http://www.machinelearning.ru/wiki/index.php?title=%D0%92%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D0%B2%D1%82%D0%BE%D1%80%D0%BE%D0%B9_%D0%BF%D1%80%D0%BE%D0%B8%D0%B7%D0%B2%D0%BE%D0%B4%D0%BD%D0%BE%D0%B9_%D0%BF%D0%BE_%D0%BE%D0%B4%D0%BD%D0%BE%D0%B9_%D0%BF%D0%B5%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%BD%D0%BE%D0%B9>

1. Информационный портал, «Метод Дихотомии»

[www.machinelearning.ru/wiki/index.php?title=%D0%9C%D0%B5%D1%82%D0%BE%D0%B4%D1%8B\_%D0%B4%D0%B8%D1%85%D0%BE%D1%82%D0%BE%D0%BC%D0%B8%D0%B8](http://www.machinelearning.ru/wiki/index.php?title=%D0%9C%D0%B5%D1%82%D0%BE%D0%B4%D1%8B_%D0%B4%D0%B8%D1%85%D0%BE%D1%82%D0%BE%D0%BC%D0%B8%D0%B8)