**Московский авиационный институт**

(национальный исследовательский университет)

**Факультет № 8 «Прикладная математика и информатика»**

**Кафедра 806 «Вычислительная математика и программирование»**

КУРСОВОЙ ПРОЕКТ

по дисциплине «Вычислительные системы»

1 семестр

на тему “Процедуры и функции в качестве параметров”

|  |  |
| --- | --- |
| Студент: | Соколов Д. В. |
| Группа: | М8О-107Б-20 |
| Преподаватель: | Найдёнов И. Е. |
| Подпись: |  |
| Оценка: |  |

Москва, 2020

**Постановка задачи**

Составить программы на языке Си с процедурами решения трансцендентных алгебраических уравнений различными численными методами (итераций, Ньютона и половинного деления — дихотомия). Нелинейные уравнения оформить как параметры-функции, разрешив относительно неизвестной величины в случае необходимости. Применить каждую процедуру к решению двух уравнений, заданных двумя строками таблицы, начиная с варианта с заданным номером. Если метод неприменим, дать математическое обоснование и графическую иллюстрацию, например, с использованием *gnuplot*.

Вариант 14

Уравнение

Отрезок [1, 2]

Приближённое значение корня 1.0769

Вариант 15

Уравнение

Отрезок [1, 2]

Приближённое значение корня 1.2388

**Теоретическая часть**

**Метод дихотомии (половинного деления)** — простейший численный метод для решения нелинейных уравнений вида f(x)=0. Он подразумевает только непрерывности функции на данном отрезке. Нa концах этого отрезка функция должна быть разных знаков: Из непрерывности следует, что на отрезке существует хотя бы один корень уравнения. Нужно найти значение xM середины отрезка Вычислим значение функции f(xM) в середине отрезка. Если значения функции в середине отрезка и на левой границе разные , то нужно переместить правую границу в середину отрезка, иначе левую границу в середину отрезка. Затем нужно повторить алгоритм начиная с вычисления значения xM  Алгоритм заканчивается тогда, когда f(xM)=0 либо xL=xR.

**Метод итераций** —ещё один простой численный метод решения уравнений. Метод основан на принципе сжимающего отображения, который применительно к численным методам в общем виде так же может называться методом простой итерации. Идея состоит в замене исходного уравнения f(x)=0 на эквивалентное ему x=φ(x). При чём должно выполнятся условие сходимости |φ’(x)|<1 на всём отрезке [a, b]. Итерации начинаются со значения xM середины отрезка. Однако φ(x) может выбрано неоднозначно. Сохраняет корни уравнения такое преобразование: Здесь λ0 – постоянная, которая не зависит от количества шагов. В данном случае мы возьмём , что приводит к простому методу одной касательной и имеет условие сходимости . Тогда итерационный процесс выглядит так: .

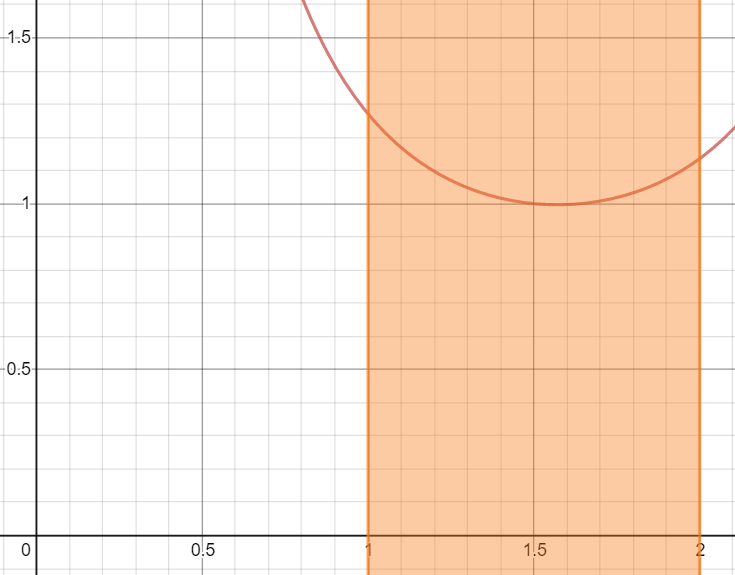
**Метод Ньютона** — итерационный численный метод нахождения корня заданной функции, который является частным случаем метода простых итераций. А именно за λ0 берётся значение производной в каждой новой точке. Тогда итерационный процесс имеет вид Условие окончания итераций и начальное значение абсолютно такие же, как и в методе итерации. Условие сходимости:

**ПРОВЕРКА НА УСЛОВИЕ СХОДИМОСТИ:**

Функции непрерывны на данном отрезке [1;2], следовательно, метод дихотомии подходит для решения двух уравнений.

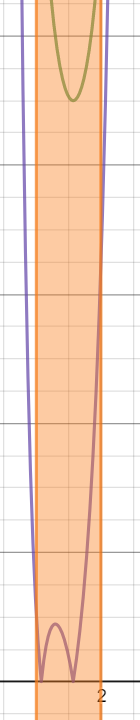
1) Возьмем производную от функции +1. λ=f'(xM)=3.010057

Проверим условия сходимости для метода итераций, построив графики производных сжимающих отображений:



На графике видно , что условие сходимости функция удовлетворяет методу итераций, потому что на интервале [1;2] функция .

Проверим условия сходимости для метода Ньютона, построив графики левых и правых частей неравенства .



- синий цвет

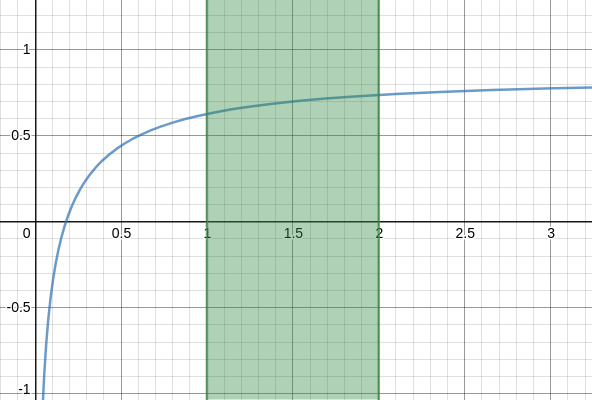
- зелёный цвет

Уравнение удовлетворяет условию сходимости метода Ньютона.

2) Возьмём производную от второй функции

λ=f'(xM)= -0.8367

Проверим условия сходимости для метода итераций, построив графики производных сжимающих отображений:

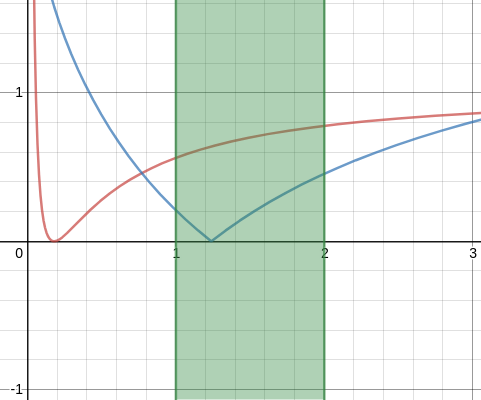


На графике видно , что условие сходимости функция удовлетворяет методу итераций, потому что на интервале [1;2] функция .

Проверим условия сходимости для метода Ньютона, построив графики левых и правых частей неравенства :

- синий цвет

- красный цвет

****

Из графика следует , что Метод Ньютона применим к этому уравнению.

**АЛГОРИТМ ВЫПОЛНЕНИЯ ПРГРАММЫ**

Первым делом программа считает машинный эпсилон. Вычисление эпсилона ЭВМ было расписано в 3-ей части КП. В этом курсовом проекте я дал эпсилон тип double для максимальной точности. Вся программа выполняется через функции , поэтому я создаю функции для расчёта уравнений по аругменту х:

double f1(double x) {

return tan(x/2.0) - 1.0 / tan(x/2.0) + x;

}

double f2(double x) {

return 0.4 + atan(sqrt(x)) - x;

}

Чтобы эти функции можно было использовать в другой функции, то надо в качестве аргумента подать нужную функцию. Так для нахождения корня уравнения по методу дихотомии нужно брать среднее значение от суммы на концах отрезка , и если функция от левого края отрезка умноженная на функцию от среднего значения между краями отрезками:

double Dihotomia\_Method(double f(double), double a, double b) {

double c = 0;

while (f(c)!=0 && fabs(b-a) > eps()) {

c = (a+b)/2;

if (f(c)\*f(a)>0) a=c;

else b = c;

}

return c;

}

Для метода итераций надо было посчитать производную от уравнения и не зависящую от шага переменную, которая равняется 1/f“(x). Алгоритм , по которому ищется корень, очень простой: . Алгоритм выполняется пока выражение не станет истинной. Для этой функции тоже придётся использоваться в качестве аргумента функцию с параметром:

double Iterations\_Method(double f(double), double a, double b) {

double x = (a + b) / 2;

double x1 = x+1;

while (fabs(x-x1)>=eps()) {

x1 = x;

x = f(x1);

}

return x;

}

Для метода Ньютона я использую формулу из прошлого метода с одним различием: вместо используется отношение функции уравнение и функции производной. Тут уже нужны две функции-параметра:

double Newton\_Method(double f(double), double fd(double), double a, double b)

{

double x = (a + b) / 2;

while (fabs(f(x)/fd(x))>=eps()) {

x -= f(x)/fd(x);

}

return x;

}

**КОД ПРОГРАММЫ**

#include <stdio.h>

#include <math.h>

double eps()

{

double e = 1;

while (e /2.0 + 1.0 > 1.0) {

e = e / 2.0;

}

return e;

}

double f1(double x) {

return tan(x/2.0) - 1.0 / tan(x/2.0) + x;

}

double proiz\_1(double x) {

return 1.0 / (2 \* cos(x / 2.0) \* cos(x / 2.0)) + 1.0 / (2 \* sin(x / 2.0) \* sin(x / 2.0)) + 1;

}

double F1(double x) {

return x - 0.3333 \* f1(x);

}

double f2(double x) {

return 0.4 + atan(sqrt(x)) - x;

}

double proiz\_2(double x) {

return (1.0 / (1 + x)) \* (1.0 / (2 \* sqrt(x))) - 1;

}

double F2(double x) {

return x + 0.8367 \* f2(x);

}

double Newton\_Method(double f(double), double fd(double), double a, double b)

{

double x = (a + b) / 2;

while (fabs(f(x)/fd(x))>=eps()) {

x -= f(x)/fd(x);

}

return x;

}

double Dihotomia\_Method(double f(double), double a, double b) {

double c = 0;

while (f(c)!=0 && fabs(b-a) > eps()) {

c = (a+b)/2;

if (f(c)\*f(a)>0) a=c;

else b = c;

}

return c;

}

double Iterations\_Method(double f(double), double a, double b) {

double x = (a + b) / 2;

double x1 = x+1;

while (fabs(x-x1)>=eps()) {

x1 = x;

x = f(x1);

}

return x;

}

void main() {

printf("epsilon = %.20lf\n", eps());

printf("--------------------------------------\n");

printf("Variant #14 --> 1.076874\n");

printf("Dihotomia Method: %10f\n", Dihotomia\_Method(f1, 1, 2));

printf("Iterations Method:%10f\n", Iterations\_Method(F1, 1, 2));

printf("Newton Method: %13f\n", Newton\_Method(f1, proiz\_1, 1, 2));

printf("Variant #15 --> 1.238840\n");

printf("Dihotomia Method: %10f\n", Dihotomia\_Method(f2, 1, 2));

printf("Iterations Method: %9f\n", Iterations\_Method(F2, 1, 2));

printf("Newton Method: %13f\n", Newton\_Method(f2, proiz\_2, 1, 2));

printf("--------------------------------------\n");

}

**ПРОТОКОЛ**

**epsilon = 0.00000000000000022204**

**--------------------------------------**

**Variant #14 --> 1.076874**

**Dihotomia Method: 1.076874**

**Iterations Method: 1.076874**

**Newton Method: 1.076874**

**Variant #15 --> 1.238840**

**Dihotomia Method: 1.238840**

**Iterations Method: 1.238840**

**Newton Method: 1.238840**

**--------------------------------------**

**Вывод**

В ходе курсового проекта я описал идеи и принципы трёх численных методов по нахождению корня уравнения: дихотомии, итераций и Ньютона. В начале КП я проверил условия сходимости данных уравнений методам и провел нужные вычисления для использования методов. Составил алгоритм решения уравнений, на основе которого составлена программа на языке Си.

**Список литературы**

1. Метод бисекции [Электронный ресурс] – URL: <https://ru.wikipedia.org/wiki/Метод_бисекции>
2. Метод простой итерации [Электронный ресурс] – URL: [https://ru.wikipedia.org/wiki/М](https://ru.wikipedia.org/wiki/Метод_бисекции)етод\_простой\_итерации
3. Метод Ньютона [Электронный ресурс] – URL: [https://ru.wikipedia.org/wiki/Метод\_](https://ru.wikipedia.org/wiki/Метод_бисекции)Ньютона