**Московский Авиационный Институт**

**(Национальный Исследовательский Университет)**

Институт № 8:

Кафедра 804

**«Информационные технологии и прикладная математика»**

**Задание №4**

**Тема: «**Решение задач на языке Си»

I семестр

|  |  |
| --- | --- |
| Студент:  Группа:  Руководители:  Подпись:  Оценка:  Дата: | Борисов С.П.  М8О-104Б-21  Безлуцкая Е.Н. |

Москва 2021 г.

**Задание**

Составить программу на языке Си с процедурами решения трансцендентных алгебраических уравнений различными численными методами (итерации, Ньютона и половинного деления – дихотомии). Нелинейные уравнения оформить как параметры-функции, разрешив относительно неизвестной величины в случае необходимости. Применить каждую процедуру к решению двух уравнений, начиная с варианта с заданным номером. Если метод неприменим, дать математическое обоснование и графическую иллюстрацию.

Вариант №19(20)

; отрезок, содержащий корень, –

; отрезок, содержащий корень, –

**Решение**

Описание методов:

1. **Метод дихотомии**

Численное нахождение приближенного значения корня функции строится на базе следствия из теоремы Больцано-Коши: «Если непрерывная функция на концах некоторого интервала имеет значение разных знаков, то внутри этого интервала у нее есть как минимум один корень».

Задача заключается в том, чтобы найти корень методом половинного деления, т.е. найти приближенное значение корня с заданной точностью .

Пусть функция непрерывна на отрезке , – единственный корень уравнения .

Поделим отрезок пополам. Получим точку и два отрезка .

Если , то корень найден (.

Если нет, то из двух полученных отрезков надо выбрать один такой, что , то есть

, если или

, если

Новый отрезок делим пополам. Получаем середину этого отрезка и так далее.

Для того, чтобы найти приближенное значение корня с точностью до , необходимо остановить процесс половинного деления на таком шаге , на котором и вычислить . Тогда можно взять .

Функция непрерывна на отрезке и на концах его принимает значения разных знаков: ;

Функция непрерывна на отрезке и на концах его принимает значения разных знаков: ;

1. **Метод итераций**

Суть метода в поиске по известному приближению искомой величины следующего, более точного приближения.

Пусть дана функция . Заменим исходное уравнение на эквивалентное . Выберем начальное приближение корня . Тогда получим некоторое число . Теперь подставляя вместо число получим . Повторяя этот процесс, будем иметь последовательность . Если эта последовательность сходящаяся, то существует предел , то данный предел является корнем уравнения и может быть вычислен по формуле .

Для преобразования в используется уравнение вида , где – некоторая постоянная, знак которой совпадает со знаком производной в некоторой окрестности корня.

Условие сходимости метода итераций:

Найдем для функции

Условие сходимости выполняется.

Найдем для функции

Условие сходимости выполняется.

1. **Метод Ньютона**

Метод Ньютона – частный случай метода итераций (). Суть метода состоит в разбиении отрезка на два отрезка с помощью касательной и выборе нового отрезка от точки пересечения касательной с осью абсцисс до неподвижной точки, на которой функция меняет знак и содержит решение.

Построение касательных продолжается до достижения необходимой точности .

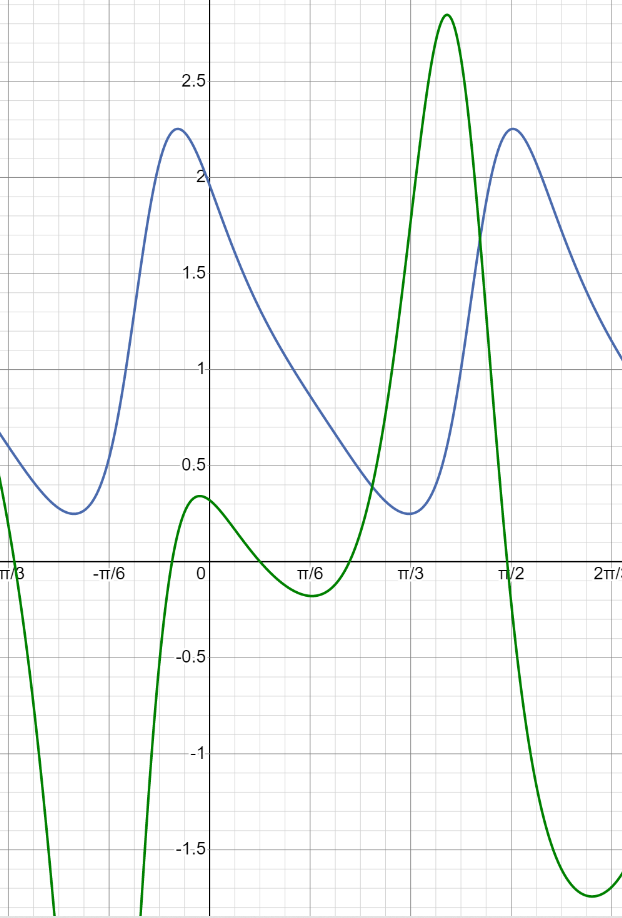
Метод итераций сходится тогда и только тогда, когда , подставим в условие выражение для и получим условие сходимости метода Ньютона:

Условие окончания: если , то значение считается приближенным значением корня уравнения

Найдем производные данных функций:

Вариант 19:

| на отрезке

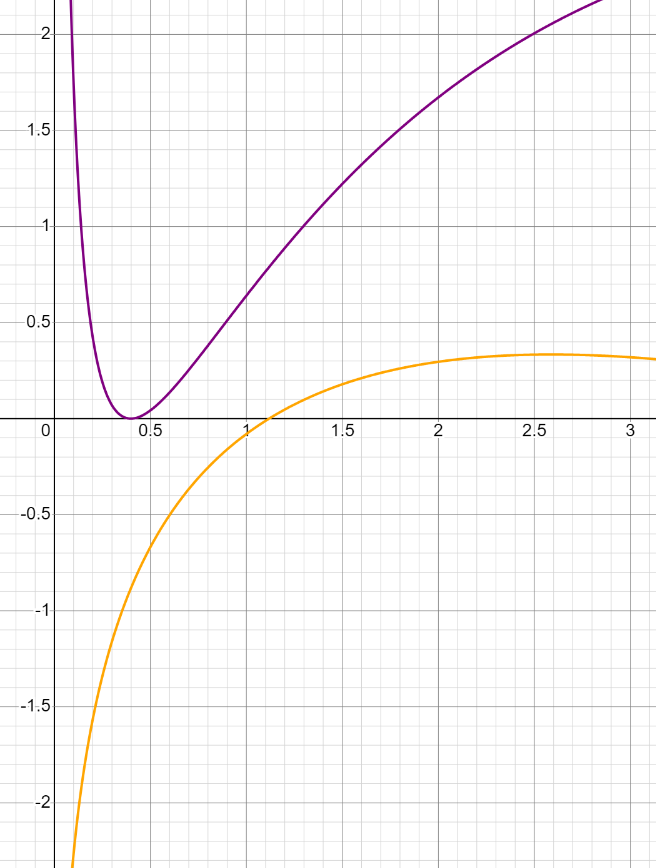
**

(синим - , зеленым - )

Условие сходимости выполняется.

Вариант 20:

| на отрезке



(фиолетовым - , оранжевым - )

Условие сходимости выполняется.

**Листинг программного кода**

#include <stdio.h>

#include <math.h>

#include <float.h>

double dichotomy(double(\*F)(double), double a, double b)

{

double x;

while (fabs(a - b) > DBL\_EPSILON) {

x = (a + b) / 2.0;

if ((\*F)(a) \* (\*F)(x) < 0.0) {

b = x;

} else {

a = x;

}

}

return x;

}

double iteration(double(\*f)(double), double a, double b)

{

double x\_pres = (a + b) / 2.0, x\_prev;

while (fabs(x\_pres - x\_prev) > DBL\_EPSILON) {

x\_prev = x\_pres;

x\_pres = (\*f)(x\_pres);

}

return x\_pres;

}

double newton(double(\*F)(double), double (\*fp)(double), double a, double b)

{

double x\_pres = (a + b) / 2.0, x\_prev;

while (fabs(x\_pres - x\_prev) > DBL\_EPSILON) {

x\_prev = x\_pres;

x\_pres -= (\*F)(x\_pres) / (\*fp)(x\_pres);

}

return x\_pres;

}

double F\_var19(double x)

{

return x - 1/(3 + sin(3.6\*x));

}

double f\_var19(double x)

{

return x - 0.71 \* (x - 1/(3 + sin(3.6\*x)));

}

double fp\_var19(double x)

{

return 1 + (3.6 \* cos(3.6\*x))/pow(3 + sin(3.6\*x), 2);

}

double F\_var20(double x)

{

return 0.1 \* pow(x, 2) - x \* log(x);

}

double f\_var20(double x)

{

return pow(exp(1.0), 0.1 \* x);

}

double fp\_var20(double x)

{

return 0.2 \* x - log(x) - 1;

}

int main(void)

{

printf("var 19: root\_dichotomy = %.20lf \nroot\_iteration = %.20lf \nroot\_newton = %.20lf \n", dichotomy(F\_var19, 0, 0.85), iteration(f\_var19, 0, 0.85), newton(F\_var19, fp\_var19, 0, 0.85));

printf("var 20: root\_dichotomy = %.20lf \nroot\_iteration = %.20lf \nroot\_newton = %.20lf \n", dichotomy(F\_var20, 1, 2), iteration(f\_var20, 1, 2), newton(F\_var20, fp\_var20, 1, 2));

return 0;

}

**Результат работы программы**

raccoonrocket@raccoonrocket-VirtualBox:~$ gcc kp4.c -lm -std=c99

raccoonrocket@raccoonrocket -VirtualBox:~$ ./a.out

var 19: root\_dichotomy = 0.26244146511922328280

root\_iteration = 0.26244146511922333831

root\_newton = 0.26244146511922328280

var 20: root\_dichotomy = 1.11832559158962951962

root\_iteration = 1.11832559158962974166

root\_newton = 1.11832559158962974166

**Выводы**

Нахождение корней трансцендентных уравнений является зачастую достаточно сложной задачей, не решаемой аналитически с помощью конечных формул. Кроме того, иногда на практике уравнение содержит коэффициенты, значения которых заданы приблизительно, так что говорить о точном решении уравнений в таких случаях вообще не имеет смысла. Поэтому задачи приближенного определения корней уравнения и соответствующей оценки их точности имеют большое значение.

В курсовом проекте были рассмотрены 3 численных метода решения трансцендентных уравнений – метод дихотомии, метод итераций и метод Ньютона.

Численные методы являются основным инструментом решения современных прикладных задач. Аналитическое решение той или иной задачи в виде отдельных формульных соотношений является скорее исключением, нежели правилом в силу сложного и приближенного характера исследуемых моделей. Вот почему численный анализ математических моделей является в настоящее время актуальным и наиболее эффективным аппаратом конструктивного исследования прикладных проблем.