МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский Авиационный Институт» (Национальный Исследовательский Университет)

Факультет: №8 «Информационные технологии и прикладная математика» Кафедра: 806 «Вычислительная математика и программирование»

Курсовой проект

по курсу «Вычислительные системы» I семестр

Задание 4

«Процедуры и функции в качестве параметров»

|  |  |
| --- | --- |
| Группа | М80-110Б-21 |
| Преподаватель | Никулин Сергей Петрович |
| Студент | Агеева Алиса Ивановна |
| Дата |  |
| Оценка |  |

**Содержание**

[Постановка задачи 3](#_bookmark0)

[Теоретическая часть 4](#_bookmark1)

[Метод половинного деления 4](#_bookmark2)

[Метод итераций 4](#_bookmark3)

[Метод Ньютона 4](#_bookmark4)

[Проверка условий сходимости 5](#_bookmark5)

[Описание алгоритма 7](#_bookmark6)

[Описание программы 8](#_bookmark7)

[Использованные в программе структуры данных 8](#_bookmark8)

[Использованные в программе переменные 8](#_bookmark9)

[Использованные в программе функции 8](#_bookmark10)

[Программа 10](#_bookmark11)

[Входные и выходные данные 14](#_bookmark12)

[Входные данные 14](#_bookmark13)

[Выходные данные 14](#_bookmark14)

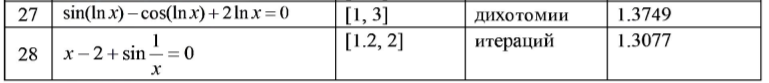
[Протокол исполнения 15](#_bookmark15)-16

[Вывод 17](#_bookmark19)

[Список литературы 18](#_bookmark20)

# 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*. В моем случае MathLib



# Теоретическая часть

**Метод половинного деления**

**Метод половинного деления** — простейший численный метод для решения нелинейных уравнений вида f(x)=0. Предполагается только непрерывность функции f(x). Для начала итераций необходимо знать отрезок [xL, xR] значений x, на концах которого функция принимает значения противоположных знаков. Это можно проверить так: 𝑓(𝑥𝐿) ∗ 𝑓(𝑥𝑅) < 0Из непрерывности следует, что на отрезке существует хотя бы один корень уравнения. Далее нужно найти

значение xM середины отрезка 𝑥𝑀

= 𝑥𝐿+𝑥𝑅Вычислим значение функции f(xM) в

2

середине отрезка. Если значения функции в середине отрезка и на левой границе разные 𝑓(𝑥𝑀) ∗ 𝑓(𝑥𝐿) < 0, 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**

**Метод итераций** — довольно 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

|f(1)(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: 𝜑(𝑥) = 𝑥 − 𝜆0 ∗ 𝑓(𝑥) Здесь λ0 – постоянная, которая не

зависит от количества шагов. В данном случае мы возьмём 𝜆 = 1 , что

0 𝑓(1)(𝑥𝑀)

приводит к простому методу одной касательной и имеет условие сходимости

𝜆0 ∗ 𝑓(1)(𝑥) > 0. Тогда итерационный процесс выглядит так: 𝑥𝑘+1 = 𝑥𝑘 − 𝜆0 ∗

𝑓(𝑥𝑘)Условием окончания итераций является достижение нужной точности между предыдущим и следующим значением.

**Метод Ньютона**

**Метод Ньютона** — итерационный численный метод нахождения корня заданной функции, который является частным случаем метода итераций. А именно за λ0 берётся значение производной в каждой новой точке. Тогда

итерационный процесс имеет вид 𝑥𝑘+1

= 𝑥𝑘

− 𝑓(𝑥𝑘) Условие окончания итераций

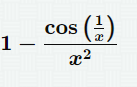
𝑓(1)(𝑥𝑘)

и начальное значение абсолютно такие же, как и в методе итерации. Условие

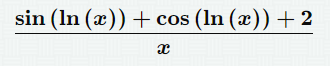
сходимости метода можно записать как |𝑓(𝑥) ∗ 𝑓(2)(𝑥)| < (𝑓(1)(𝑥))

**Проверка условий сходимости**

Пусть первое уравнение f1(x), а второе уравнение f2(x). Функции непрерывны на заданных промежутках, значит метод дихотомии применим к ним. Найдём производные f1(1)(x) и f2(1)(x) для заданных функций:



f1’ =



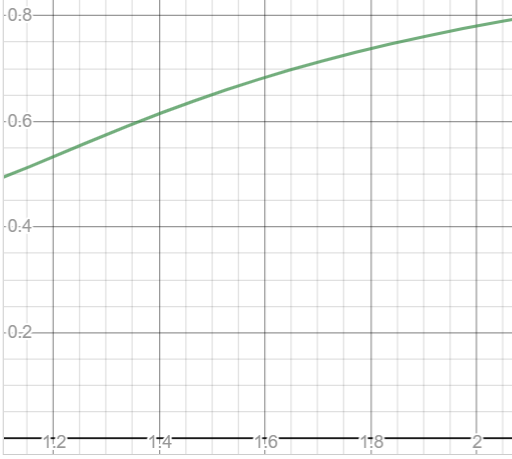
f2’ =

λ1 = 1,25602

λ2 = 1,000153

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

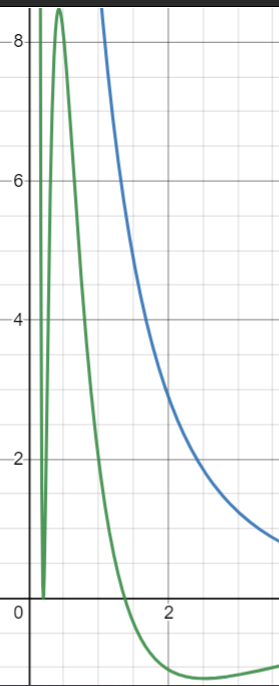
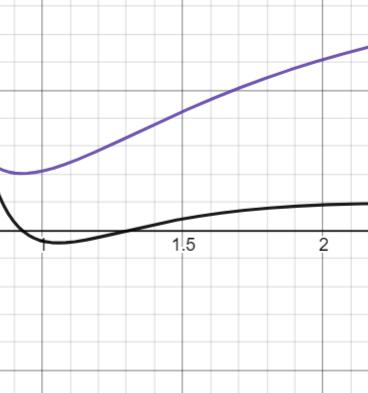
|f1’(x)|<1 на отрезке |f2’(x)|<1 на отрезке



Как видно, метод итераций **применим** для данных функций.

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

|f1’’(x)\*f1(x)|<(f1’(x))^2 |f2’’(x)\*f2(x)|<(f2’(x))^2



Оба уравнения **удовлетворяют условию сходимости**  метода Ньютона.

# Описание алгоритма

Рассмотрим алгоритм решения. Сперва нужно найти машинное эпсилон, на котором будет основываться точность вычисления. Это можно сделать просто деля 1 на 2 за O(log(1016))~O(1). Опишем каждую из функций решения уравнений разными методами. Для метода дихотомии достаточно просто выполнять итерации, пока xR-xL>ε. Корень с помощью такого метода рассчитывается примерно за O(log210k)~O(k\*log210). Все функции, их производные и сжимающие отображения зададим в виде функций-параметров. Алгоритмы для решения уравнений методом итерации и Ньютона реализованы так же, как и было описано в теории. Невозможно оценить из сложность, так как она зависит от самой функции. Будем сохранять все корни и сразу же из выводить, не затрачивая память на новые переменные.

# Описание программы

**Использованные в программе структуры данных**

|  |  |  |
| --- | --- | --- |
| Название структуры | Переменные в структуре | Смысл структуры |
| root\_x | int steps double x | steps – то самое N, число итераций, затраченное на вычисление корня.  x – то самое x0, искомое значение корня |

**Использованные в программе переменные**

|  |  |  |
| --- | --- | --- |
| Название переменной | Тип переменной | Смысл переменной |
| k | int | То самое число K, используемое для вычисления точности. Так же обозначает, что вывод должен быть с точность до K знаков после запятой |
| e0 | double | То самое машинное эпсилон. В случае с double ε =2.20 \* 10-16 |
| acc | double | Точность вычислений. Именно с этой переменной мы будем сравнивать ответы A1 и A2 |
| x0 | root | Значение корня, которое будут возвращать функции после вычисления. |

**Использованные в программе функции**

|  |  |  |
| --- | --- | --- |
| Название функции | Тип возвращаемой переменной | Смысл функции |
| square | double | Возвращает квадрат числа |
| do\_nothing | double | Копирует значение в память, где double выделяется 64 бита, а не 80 бит |
| solve\_binary\_search | root | Решение уравнения методом половинного деления. Принимает f – функцию-параметр f(x), l – xL, r – xR , k и acc |

|  |  |  |
| --- | --- | --- |
| solve\_iteration | root | Решение уравнения методом итерации.  Принимает f – функцию- параметр φ(x), x0 – xM,  k и acc |
| solve\_newton | root | Решение уравнения методом Ньютона.  Принимает f – функцию- параметр f(x), derivative\_f – функцию- параметр f(1)(x), x0 – xM, k и acc |
| f13 | double | То самое f1(x) |
| f14 | double | То самое f2(x) |
| squeeze\_f13 | double | То самое φ1(x) |
| squeeze\_f14 | double | То самое φ2(x) |
| d\_dx\_f13 | double | То самое f1(1)(x) |
| d\_dx\_f14 | double | То самое f2(1)(x) |

**Программа**

#include <math.h>

#include <stdio.h>

typedef struct root\_x root;

struct root\_x{

double x;

int steps;

};

double do\_nothing(double x){

return x;

}

root solve\_binary\_search(double f(double), double l, double r, double

acc) {

int step = 0;

double m;

while(r - l > acc) {

step++;

m = (l + r) / 2.0;

if (f(m) \* f(l) < 0) {

r = m;

} else {

l = m;

}

}

root ans = {m, step};

return ans;

}

root solve\_iteration(double f(double), double x0, double acc) {

int step = 0;

double cur = x0;

double prev = cur + 1;

while (fabs(cur - prev) > acc) {

prev = cur;

cur = f(prev);

step++;

}

root ans = {cur, step};

return ans;

}

root solve\_newton(double f(double), double derivative\_f(double), double x0, double acc) {

int step = 0;

double cur = x0;

double prev = cur + 1;

while (fabs(cur - prev) > acc) {

prev = cur;

cur=prev - f(prev) / derivative\_f(prev);

step++;

}

root ans = {cur, step};

return ans;

}

double f13(double x) {

return sin(log(x))-cos(log(x))+2\*log(x);

}

double f14(double x) {

return x - 2 + sin(1/x);

}

double squeeze\_f13(double x) {

return x - f13(x) / 1.25602;

}

double squeeze\_f14(double x) {

return x - f14(x) / (1.000153);

}

double d\_dx\_f13(double x) {

return (sin(log(x))+cos(log(x))+2)/x;

}

double d\_dx\_f14(double x) {

return 1 - (cos(1/x)/(x\*x));

}

int main()

{

double e0 = 1.0;

while(do\_nothing(1.0 + e0 / 2.0) > 1.0) {

e0 = e0 / 2.0;

}

printf("Machine epsilon equals %.8e\n", e0);

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

double acc=10\*e0;

root x0;

printf("Answer for: sin(ln(x)) - cos(ln(x)) + 2\*ln(x) = 0 \n");

x0 = solve\_binary\_search(f13, 1.0, 3.0, acc);

printf("Методом дихотомии: %.16f | %d\n", x0.x, x0.steps);

x0 = solve\_iteration(squeeze\_f13, (1.0+3.0)/2.0, acc);

printf("Методом итераций: %.16f | %d\n", x0.x, x0.steps);

x0 = solve\_newton(f13, d\_dx\_f13, (1.0+3.0)/2.0, acc);

printf("Методом Ньютона: %.16f | %d\n\n", x0.x, x0.steps);

printf("Answer for: x - 2 + sin(1/x) = 0 \n");

x0 = solve\_binary\_search(f14, 1.2, 2.0, acc);

printf("Методом дихотомии: %.16f | %d\n", x0.x, x0.steps);

x0 = solve\_iteration(squeeze\_f14, (1.2+2.0)/2.0, acc);

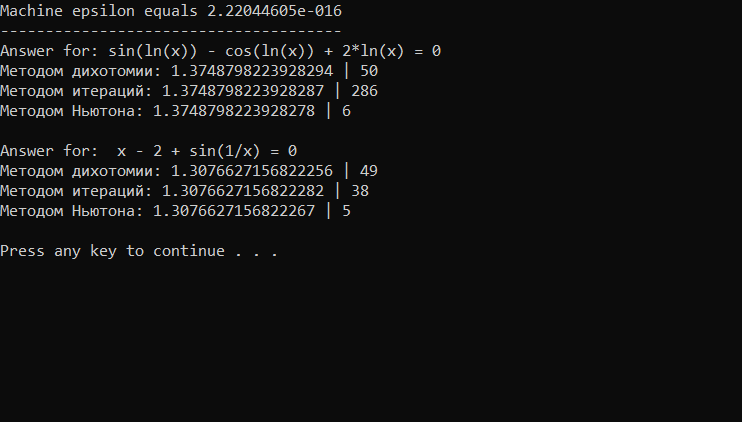
printf("Методом итераций: %.16f | %d\n", x0.x, x0.steps);

x0 = solve\_newton(f14, d\_dx\_f14, (1.2+2.0)/2.0, acc);

printf("Методом Ньютона: %.16f | %d\n", x0.x, x0.steps);

return 0;

}



**Вывод**

В работе описаны идеи и принципы трёх численных методов: дихотомии, итераций и Ньютона. Проверены условия сходимости данных уравнений методам и проведены нужные вычисления для использования методов.

Составлен алгоритм решения уравнений, на основе которого составлена программа на языке Си. Описан формат ввода и вывода, проведено тестирование программы, составлен протокол исполнения программы.

# Список литературы

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