Московский авиационный институт  
(национальный исследовательский университет)  
  
Институт №8 «Компьютерные науки и прикладная математика»  
[Кафедра 806 «Вычислительная математика и программирование»](https://mai.ru/content/org/index.php?SECTION_ID=&ID=5042)

КУРСОВОЙ ПРОЕКТ  
по курсу "Фундаментальная информатика"  
I семестр  
на тему «Процедуры и функции в качестве параметров»

Студент: Клименко В.М.  
Группа: М8О-103Б-22, № 11  
Руководитель: Никулин С.П., доцент 806 кафедры  
Москва, 2022

Оглавление

[Введение 3](#_Toc121269270)

[Теория 4](#_Toc121269271)

[Метод дихотомии 4](#_Toc121269272)

[Метод итераций 4](#_Toc121269273)

[Метод Ньютона 4](#_Toc121269274)

[Практика 5](#_Toc121269275)

[Задание 5](#_Toc121269276)

[Методы решения 5](#_Toc121269277)

[Сведения о программе 5](#_Toc121269278)

[Предназначение программы 5](#_Toc121269279)

[Описание программы 6](#_Toc121269280)

[Переменные 9](#_Toc121269281)

[Тесты 10](#_Toc121269282)

[Заключение 11](#_Toc121269283)

[Список литературы 11](#_Toc121269284)

## Введение

Нахождение корней трансцендентных уравнений очень часто является довольно сложной задачей, которую нельзя решить аналитически с помощью конечных формул или простого перебора. Также бывают случаи, когда на практике уравнение содержит коэффициенты, значения которых заданы приблизительно, так что речь о точном решении такого уравнения не идет, поэтому задачи приближенного определения корней уравнения и соответствующей оценки их точности имеют большое значение.

Цель этой работы – научиться реализовывать математические методы в языке программирования Си. В ходе проекта научиться передавать процедуры и функции в качестве параметров в другие процедуры или функции.

Проект реализован составлением программы 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.

## Теория

### Метод дихотомии

Пусть задана функция .

Разобьём мысленно заданный отрезок пополам и возьмём две симметричные относительно центра точки так, что:

где — некоторое число в интервале .

Вычислим два значения функции в новых точках. Сравнением определим в какой из двух новых точек значение функции максимально. Отбросим тот из концов изначального отрезка, к которому точка с максимальным значением функции оказалась ближе (напомним, мы ищем минимум), то есть:

* Если , то берётся отрезок , а отрезок отбрасывается.
* Иначе берётся зеркальный относительно середины отрезок , а – отбрасывается.
* Условие окончания: .

### Метод итераций

Идея метода заключается в замене исходного уравнения уравнением вида . Достаточное условие сходимости метода: .

* Начальное приближение корня: .
* Итерационный процесс: .
* Условие окончания: .

### Метод Ньютона

Метод Ньютона – частный случай метода итераций.

* Условие сходимости: на отрезке .
* Итерационный процесс: .

## Практика

### Задание

Составить программу на языке Си с процедурами решения трансцендентных алгебраических уравнений различными численными методами (итераций, 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, заданных двумя строками таблицы, начиная с варианта с заданным номером. Если метод неприменим, дать математическое обоснование и графическую иллюстрацию, например, с использованием gnuplot.

Заданные функции:

### Методы решения

Создать программу на Си с функциями, которые реализуют заданные математические методы приближения значения аргумента к корню математической функции. В ходе проекта надо использовать передачу программных функций как параметры в другие функции для универсализации. В методах итераций используется вспомогательная функция, которая вычислена заранее, так как ее вычисления – трудная отдельная задача. То же самое с производной в методе Ньютона. Также, математические функции, передаваемые в функции методов должны быть проверены пользователем на сходимость заранее, так как программирование такой проверки – слишком тяжелая задача для данного проекта.

### Сведения о программе

В программе присутствуют не только эквиваленции математических функций и функции реализации математических методов, но и вспомогательные функции, например функция вычисления машинного эпсилон.

### Предназначение программы

Цель проекта – реализовать математические методы, научиться более продвинуто использовать функции языка программирования Си.

Для программиста эта программа имеет ценность со стороны как раз реализации математических методов с использованием данных инструментов языка программирования Си, а для математика главная ценность представляет сравнение точности вычислений различными способами приближенных корней уравнения.

### Описание программы

Программы состоит из списка функций:

* *main* – основная функция программы, в которой осуществляется ввод и вывод:

int main() {

long double eps = calc\_eps(), k = 1.;

long double a1 = -1., b1 = 0.;

long double a2 = 2., b2 = 3.;

long double d1 = dichotomy(a1, b1, k \* eps, f1), d2 = dichotomy(a2, b2, k \* eps, f2);

long double i1 = iterations((a1 + b1)/2., k \* eps, f1, fi1), i2 = iterations((a2 + b2)/2., k \* eps, f2, fi2);

long double n1 = newton((a1 + b1)/2., k \* eps, f1, fd1), n2 = newton((a2 + b2)/2., k \* eps, f2, fd2);

printf("\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

printf("Function: | Method: | Root: | Function at found x: \n");

printf("e^x+sqrt(1+e^(2x))-2 | Dichotomy | %.20Lf | %.20Lf\n", d1, f1(d1));

printf("e^x+sqrt(1+e^(2x))-2 | Iterations | %.20Lf | %.20Lf\n", i1, f1(d1));

printf("e^x+sqrt(1+e^(2x))-2 | Newton | %.20Lf | %.20Lf\n", n1, f1(d1));

printf("lnx - x + 1.8 | Dichotomy | %.21Lf | %.20Lf\n", d2, f2(d2));

printf("lnx - x + 1.8 | Iterations | %.21Lf | %.20Lf\n", i2, f2(d2));

printf("lnx - x + 1.8 | Newton | %.21Lf | %.20Lf\n", n2, f2(d2));

printf("\nMachine-calculated epsilon: %.54Lf\n", eps);

return 0;

}

* *calc\_eps* – вычисление машинного эпсилон:

long double calc\_eps() {

long double e = 1;

long double eps;

while(1 < (1 + e)){

eps = e;

e /= 2;

}

return eps;

}

* *mabs* – вычисление абсолютного значения числа, представленного в типе данных *long double*:

long double mabs(long double a) {

if (a < 0) {

return -a;

}

return a;

}

* *f1, fd1, fi1, f2, fd2, fi2* – математические функции (сама функция, ее производная, вспомогательная функция для метода итераций; приписка с номером отвечает за номер функции в заданном варианте):

long double f1(long double x) {

return powl(M\_E, x) + sqrtl(1 + powl(M\_E, 2\*x)) - 2;

}

long double fd1(long double x) {

return powl(M\_E, x) + 1/(sqrtl(1 + powl(M\_E, 2\*x)));

}

long double fi1(long double x) {

return logl(2 - sqrtl(1 + powl(M\_E, 2\*x)));

}

long double f2(long double x) {

return logl(x) - x + 1.8;

}

long double fd2(long double x) {

return 1/x - 1;

}

long double fi2(long double x) {

return logl(x) + 1.8;

}

* *dichotomy* – реализация метода дихотомии:

long double dichotomy(long double a, long double b, long double e, long double (\*f) (long double)) {

long double c = (a + b) / 2., fb = f(b), fc = f(c);

long double answ;

if (c + e > b) { // same as x ≈ (ak + bk)/2

answ = c;

} else if ((fc \* fb) < 0) {

answ = dichotomy(c, b, e, f);

} else if ((fc \* fb) > 0) {

answ = dichotomy(a, c, e, f);

} else {

answ = c;

}

return answ;

}

* *iterations* – реализация метода итераций:

long double iterations(long double x, long double e, long double (\*f) (long double), long double (\*fi) (long double)) {

long double fx = fi(x);

if (mabs(x - fx) < e) {

return x;

}

return iterations(fx, e, f, fi);

}

* *newton* – реализация метода Ньютона:

long double newton(long double x, long double e, long double (\*f) (long double), long double (\*fd) (long double)) {

long double fx = x - f(x)/fd(x);

if (mabs(x - fx) < e) {

return fx;

}

return newton(fx, e, f, fd);

}

* для удобной компиляции и запуска программы был создан *makefile*:

CC = gcc

CFLAGS = -Wextra -Wall

main:

$(CC) $(CFLAGS) -o main.out main.c -lm

./main.out

### Переменные

|  |  |  |  |
| --- | --- | --- | --- |
| Имя | Тип | Начальное значение | Назначение |
| *eps* | *long double* | *calc\_eps()* | Машинный эпсилон |
| *k* | *long double* | 1.5 | Коэффициент *eps* |
| *a1* | *long double* | -1.0 | Начало заданного отрезка 1-ой функции |
| *b1* | *long double* | 0.0 | Конец заданного отрезка 1-ой функции |
| *a2* | *long double* | 2.0 | Начало заданного отрезка 2-ой функции |
| *b2* | *long double* | 3.0 | Конец заданного отрезка 2-ой функции |
| *d1* | *long double* | *dichotomy(a1*, *b1*, *k* \* *eps*, *f1)* | Корень 1-ой функции, найденный методом дихотомии |
| *d2* | *long double* | *dichotomy(a2*, *b2*, *k* \* *eps*, *f2)* | Корень 2-ой функции, найденный методом дихотомии |
| *i1* | *long double* | *iterations((a1* + *b1)*/2., *k* \* *eps*, *f1, fi1)* | Корень 1-ой функции, найденный методом итераций |
| *i2* | *long double* | *iterations((a2* + *b2)*/2., *k* \* *eps, f2, fi2)* | Корень 2-ой функции, найденный методом итераций |
| *n1* | *long double* | *newton((a1* + *b1)*/2., *k* \* *eps, f1, fd1)* | Корень 1-ой функции, найденный методом Ньютона |
| *n2* | *long double* | *newton((a2* + *b2)*/2., *k* \* *eps*, f2, fd2*)* | Корень 2-ой функции, найденный методом Ньютона |

## Тесты

Данные из варианта:

на отрезке ;

на отрезке ;

Результат:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Function: | Method: | Root: | Function at found x:

e^x+sqrt(1+e^(2x))-2 | Dichotomy | -0.28768207245178094278 | 0.00000000000000000000

e^x+sqrt(1+e^(2x))-2 | Iterations | -0.28768207245178093322 | 0.00000000000000000000

e^x+sqrt(1+e^(2x))-2 | Newton | -0.28768207245178094281 | 0.00000000000000000000

lnx - x + 1.8 | Dichotomy | 2.845868181474182400607 | 0.00000000000000000000

lnx - x + 1.8 | Iterations | 2.845868181474182400607 | 0.00000000000000000000

lnx - x + 1.8 | Newton | 2.845868181474182400607 | 0.00000000000000000000

Machine-calculated epsilon: 0.000000000000000000108420217248550443400745280086994171

## Заключение

В ходе данного проекта мне удалось запрограммировать несколько способов решения задачи о приближении аргумента к корню заданного уравнения. Во время написания программы я столкнулся с задачей передачи функции как параметра другой функции, которую я смог решить. Также было изучено больше информации о машинном эпсилон. Также я узнал о больших возможностях языка программирования Си (возможности математической библиотеки).

## Список литературы

<http://www.machinelearning.ru/wiki/index.php?title=Методы_дихотомии> - подробное описание метода дихотомии

<https://alexlarin.net/Int/zad.htm> - подробное описание метода итераций

<https://ru.algorithmica.org/cs/numerical/newton/> - подробное описание метода Ньютона