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»

Курсовая работа

по курсу «Вычислительные системы»

I семестр

Задание 4

«Процедуры и функции в качестве параметров»

|  |  |
| --- | --- |
| Группа: | М8О-107Б-19 |
| Студент: | Инютин Максим Андреевич |
| Преподаватель: | Сластушенский Юрий Викторович |
| Оценка: |  |
| Дата: |  |

**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 13

Уравнение

Отрезок [0.2, 1]

Вариант 14

Уравнение

Отрезок [1, 2]

**Теоретическая часть**

**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: Из непрерывности следует, что на отрезке существует хотя бы один корень уравнения. Далее 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** — 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 |φ(1)(x)|<0 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: Условием окончания итераций является достижение нужной точности между предыдущим и следующим значением.

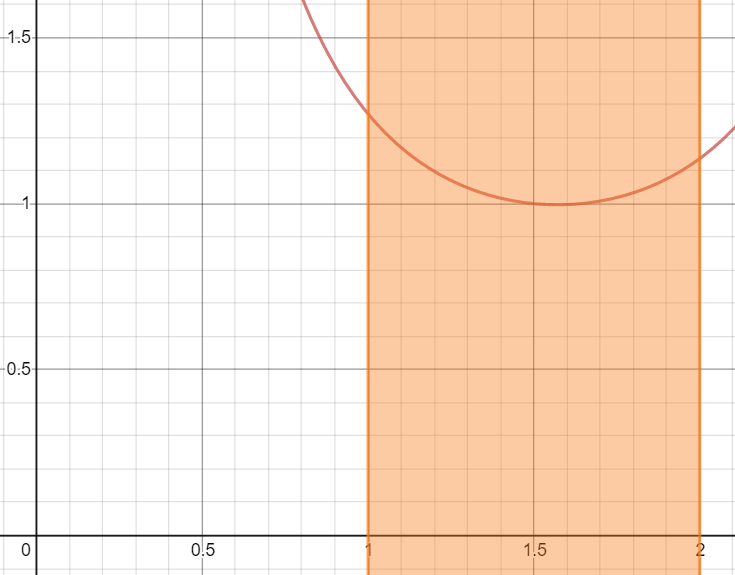
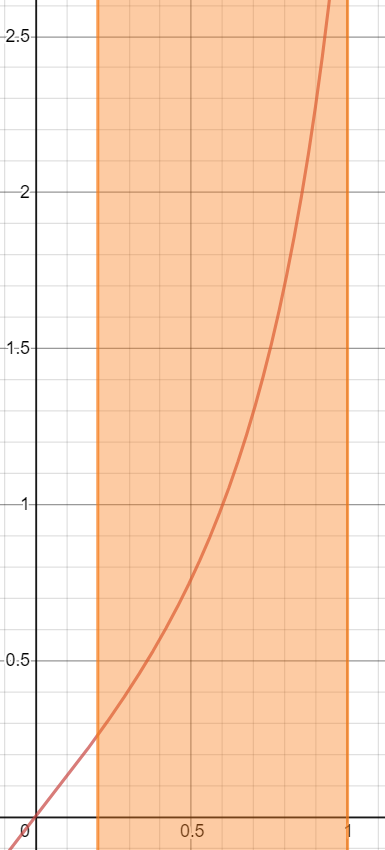
**Метод Ньютона** — итерационный численный метод нахождения корня заданной функции, который является частным случаем метода итераций. А именно за λ0 берётся значение производной в каждой новой точке. Тогда итерационный процесс имеет вид Условие окончания итераций и начальное значение абсолютно такие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

**Проверка условий сходимости**

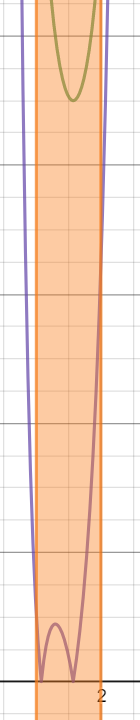
Пусть первое уравнение f1(x), а второе уравнение f2(x). Функции непрерывны на заданных промежутках, значит метод дихотомии применим к ним. Найдём производные f1(1)(x) и f2(1)(x) для заданных функций:

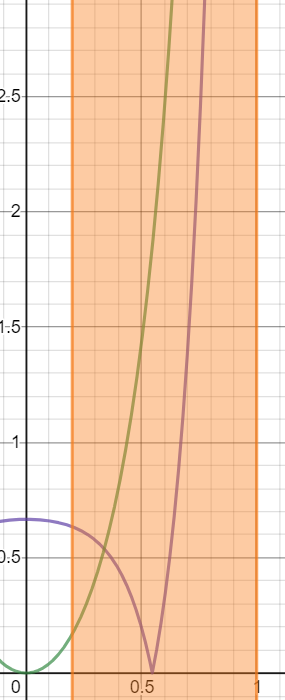
λ1=f1(1)(xM)=1.564962... и λ2=f2(1)(xM)=3.010057...

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



Оба уравнения удовлетворяют условию сходимости метода итераций

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



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.

**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**

Команды компилятора: -std=c11

#include <math.h>

#include <stdio.h>

typedef struct root\_x root;

struct root\_x{

double x;

int steps;

};

double square(double x){

return x\*x;

}

double do\_nothing(double x){

return x;

}

root solve\_binary\_search(double f(double), double l, double r, int k, 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, int k, 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, int k, 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 x\*tan(x)-1.0/3.0;

}

double f14(double x){

return tan(x/2.0)-1/tan(x/2.0)+x;

}

double squeeze\_f13(double x){

return x-f13(x)/1.564962;

}

double squeeze\_f14(double x){

return x-f14(x)/3.010057;

}

double d\_dx\_f13(double x){

return tan(x)+x/square(cos(x));

}

double d\_dx\_f14(double x){

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

}

int main(){

int k;

scanf("%d", &k);

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=e0\*pow(10, 16-k);

root x0;

printf("Answer for x\*tg(x)-1/3=0\n");

x0=solve\_binary\_search(f13, 0.2, 1.0, k, acc);

printf("%.\*f | %d\n", k, x0.x, x0.steps);

x0=solve\_iteration(squeeze\_f13, (0.2+1.0)/2.0, k, acc);

printf("%.\*f | %d\n", k, x0.x, x0.steps);

x0=solve\_newton(f13, d\_dx\_f13, (0.2+1.0)/2.0, k, acc);

printf("%.\*f | %d\n", k, x0.x, x0.steps);

printf("Answer for tg(x)-ctg(x)+x=0\n");

x0=solve\_binary\_search(f14, 1.0, 2.0, k, acc);

printf("%.\*f | %d\n", k, x0.x, x0.steps);

x0=solve\_iteration(squeeze\_f14, (1.0+2.0)/2.0, k, acc);

printf("%.\*f | %d\n", k, x0.x, x0.steps);

x0=solve\_newton(f14, d\_dx\_f14, (1.0+2.0)/2.0, k, acc);

printf("%.\*f | %d\n", k, x0.x, x0.steps);

return 0;

}

**Входные данные**

Единственная строка содержит одно целое число K (0≤K≤16) — коэффициент для точности вычисления искомого корня.

**Выходные данные**

Программа должна вывести значение машинного эпсилон, а затем 6 строк. В каждой строке необходимо вывести искомое значение корня x0 с точность K знаков после запятой и N — число итераций. В первых трёх строках для первого уравнения, а в следующих трёх для другого. Следует выводить сначала значения, полученные методом дихотомии, потом методом итерации, затем методом Ньютона.

**Протокол исполнения и тесты**

Тест #1

Ввод:

6

Вывод:

Machine epsilon equals 2.22044605e-016

--------------------------------------

Answer for x\*tg(x)-1/3=0

0.547160 | 19

0.547161 | 6

0.547161 | 4

Answer for tg(x)-ctg(x)+x=0

1.076876 | 19

1.076874 | 8

1.076874 | 4

Тест #2

Ввод:

10

Вывод:

Machine epsilon equals 2.22044605e-016

--------------------------------------

Answer for x\*tg(x)-1/3=0

0.5471607571 | 32

0.5471607573 | 11

0.5471607573 | 5

Answer for tg(x)-ctg(x)+x=0

1.0768739864 | 33

1.0768739863 | 14

1.0768739863 | 5

Тест #3

Ввод:

16

Вывод:

Machine epsilon equals 2.22044605e-016

--------------------------------------

Answer for x\*tg(x)-1/3=0

0.5471607572603301 | 52

0.5471607572603301 | 17

0.5471607572603300 | 5

Answer for tg(x)-ctg(x)+x=0

1.0768739863118035 | 52

1.0768739863118038 | 22

1.0768739863118035 | 6

**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 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