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-20 |
| Студент: | Алапанова Эльза Халилевна |
| Преподаватель: |  |
| Оценка: |  |
| Дата: |  |

Москва, 2021

**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 и тесты 15](#_bookmark15)

[Тест #1 15](#_bookmark16)

[Тест #2 15](#_bookmark17)

[Тест #3 15](#_bookmark18)

[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*.

Вариант 23

Уравнение 3𝑥 − 4𝑙𝑛𝑥 − 5 = 0

Отрезок [2, 4]

Вариант 24 Уравнение

𝑐𝑜𝑠 2

𝑥

− 2𝑠𝑖𝑛 1 + 1 = 0

𝑥 𝑥

Отрезок [1, 2]

# 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

|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: 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

2

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:

2 1

𝑓(1)(𝑥) = 3 − 4 ; 𝑓(1)(𝑥) = 2 sin (𝑥) + 2 cos (𝑥) − 1

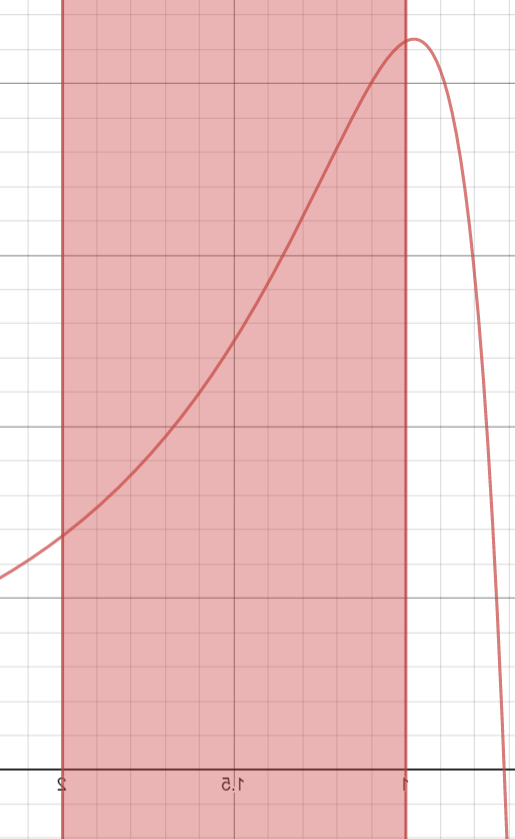
1 𝑥 2

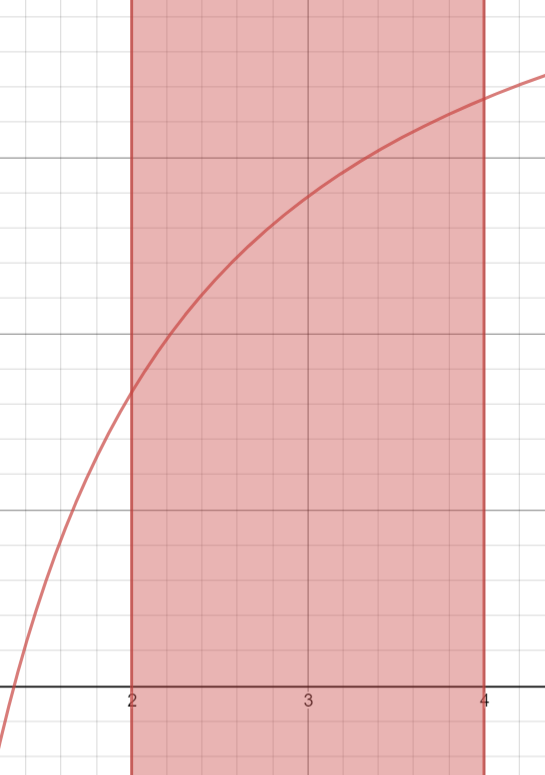
𝑥2

𝑥2

𝑥2

λ1=f1(1)(xM)=1.666666... и λ2=f2(1)(xM)=1.1180668...

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

𝑦 = 𝜆1

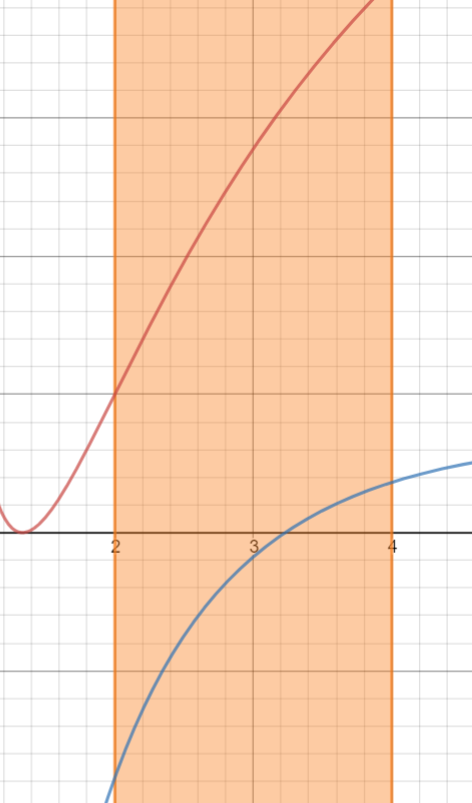
1

∗ 𝑓(1)(𝑥)

𝑦 = 𝜆2

2

∗ 𝑓(1)(𝑥)



Оба уравнения удовлетворяют условию сходимости метода итераций.

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

𝑦 = |𝑓 (𝑥) ∗ 𝑓(2)(𝑥)|

1 1 1

𝑦2 = (𝑓(1)( )

2

1 𝑥 )

𝑦 = |𝑓 (𝑥) ∗ 𝑓(2)(𝑥)|

1 2 2

𝑦2 = (𝑓(1)(𝑥))

2

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 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 3 \* x - log(x) - 5;

}

double f14(double x) {

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

}

double squeeze\_f13(double x) { return x - f13(x) / 1.666667;

}

double squeeze\_f14(double x) { return x - f14(x) / 1.118067;

}

double d\_dx\_f13(double x) {

return 3 - (4 / x);

}

double d\_dx\_f14(double x) {

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

}

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 3x-4ln(x)-5=0\n");

x0 = solve\_binary\_search(f13, 2.0, 4.0, k, acc); printf("%.\*f | %d\n", k, x0.x, x0.steps);

x0 = solve\_iteration(squeeze\_f13, (2.0+4.0)/2.0, k, acc); printf("%.\*f | %d\n", k, x0.x, x0.steps);

x0 = solve\_newton(f13, d\_dx\_f13, (2.0+4.0)/2.0, k, acc); printf("%.\*f | %d\n", k, x0.x, x0.steps);

printf("Answer for cos(2/x)-2sin(1/x)+1/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 и выходные данные

**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.

Ввод: 6

Вывод:

# Протокол исполнения и тесты

**Тест #1**

Machine epsilon equals 2.22044605e-016

Answer for 3x-4ln(x)-5=0 3.999998 | 20

1.876463 | 20

-nan | 3

Answer for cos(2/x)-2sin(1/x)+1/x=0 1.875616 | 19

1.875616 | 12

1.875617 | 5

Ввод: 10

Вывод:

**Тест #2**

Machine epsilon equals 2.22044605e-016

Answer for x\*tg(x)-1/3=0 Answer for 3x-4ln(x)-5=0 3.9999999999 | 34

1.8764628467 | 33

-nan | 3

Answer for cos(2/x)-2sin(1/x)+1/x=0 1.8756173925 | 33

1.8756173924 | 22

1.8756173925 | 5

Ввод: 16

Вывод:

**Тест #3**

Machine epsilon equals 2.22044605e-016

Answer for 3x-4ln(x)-5=0 4.0000000000000000 | 53

1.8764628467174300 | 49

-nan | 3

Answer for cos(2/x)-2sin(1/x)+1/x=0 1.8756173924904573 | 52

1.8756173924904571 | 36

1.8756173924904571 | 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 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>