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: 805 «Математическая кибернетика»

Курсовая работа
по курсу «Вычислительные системы»
1 семестр

Задание 4
«Процедуры и функции в качестве параметров»

Студент: Минеева Светлана Алексеевна
Группа: М8О-105Б-21
Руководитель: Титов Вячеслав Константинович
Оценка: _____
Дата: 21.12.21

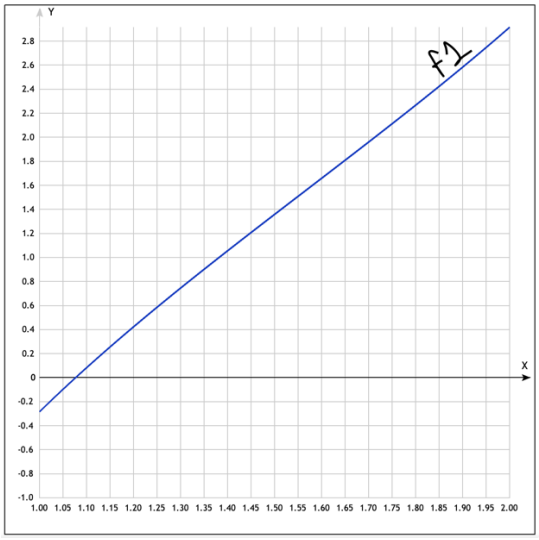
Москва, 2021

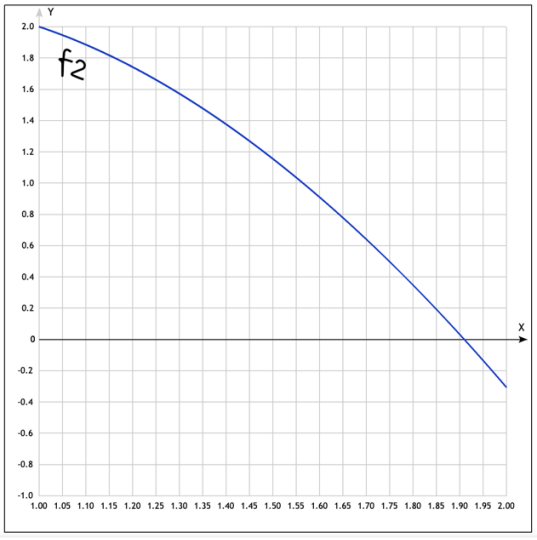
Содержание

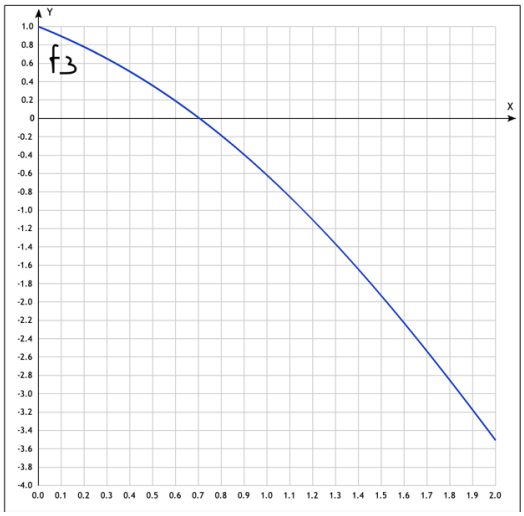
Введение

Задание (вариант №14): 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*.

№	Уравнение	Отрезок, содержащий корень	Приближенное значение корня
1 (из варианта №14)	tg(x/2) - ctg(x/2) + x	[1, 2]	1.0769
2	ln(x) - x*x + 3	[1, 2]	1.9097
3	cos(x) + sin(x) - 2x	[0, 2]	0.7048







1. Метод деления пополам (дихотомия)

Рассмотрим метод деления пополам на уравнении №2 (ln(x) – x*x +3).

Метод половинного деления — простейший численный метод для решения нелинейных уравнений вида f(x)=0. Предполагается только непрерывность функции f(x). Из непрерывности следует, что на отрезке существует хотя бы один корень уравнения.

Для начала итераций необходимо знать отрезок $[x_L, x_R]$ значений x, на концах которого функция принимает значения противоположных знаков. Это можно проверить так: $f(x_L) * f(x_R) < 0$. Проверим это для уравнения на данном отрезке [1, 2].
f(1) = ln(1) - 1*1 + 3 = 0 – 1 + 3 = 2 ;
f(2) = ln(2) – 2*2 + 3 = 0.693147 – 4 + 3 = -0.306853;
f(1)*f(2) = -0.306853 * 2 = -0.613706 < 0. Следовательно, отрезок удовлетворяет условию.

Далее нужно найти значение x_m середины отрезка $x_m = (x_L + x_R)/2$.
Вычислим значение функции $f(x_m)$ в середине отрезка и функцию в этой точке:
$x_m = (1 + 2)/2 = 1.5$;
f(1.5) = ln(1.5) – 1.5*1.5 + 3 = 0.405465 – 2.25 + 3 = 1.15547.

Если значения функции в середине отрезка и на левой границе разные $f(x_m) * f(x_L) < 0$, то нужно переместить правую границу в середину отрезка, иначе левую границу в середину отрезка. В нашем случае значения с разными знаками, следовательно правая граница меняет своё значение на значение середины отрезка.

Затем нужно повторить алгоритм начиная с вычисления значения x_M.
Алгоритм заканчивается тогда, когда $f(x_M)=0$ либо $x_L=x_R$.

Проверим отрезки для остальных уравнений:

Уравнение №1 (tg(x/2) – ctg(x/2) + x), отрезок – [1, 2]:
f(1) = tg(1/2) – ctg(1/2) + 1 = 0.546302 – 1.83049 + 1 = -0.284185;
f(2) = tg(1) – ctg(1) + 2 = 1.55741 - 0.642093 + 2 = 2.91532;
f(1)*f(2) = -0.284185 * 2.91532 = -0.82849021 < 0. Следовательно, отрезок удовлетворяет условию.

Уравнение №3 (cos(x) + sin(x) – 2x), отрезок – [0, 2]:
f(1) = cos(0) + sin(0) – 2*0 =1 + 0 - 0 = 1;
f(2) = cos(2) + sin(2) – 2*2 = -0.416147 + 0.909297 - 4 = -3.50685;

f(1)*f(2) = 1 * (-3.50685) = -3.50685 < 0. Следовательно, отрезок удовлетворяет условию.

2. 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 $|\varphi^{(1)}(x)|<0$ на всём отрезке [a, b]. Итерации начинаются со значения x_M середины отрезка. Однако φ(x) может выбрано неоднозначно. Сохраняет корни уравнения такое преобразование: $\varphi(x) = x - \lambda_0 * f(x)$. Здесь λ_0 – постоянная, которая не зависит от количества шагов. В данном случае мы возьмём $\lambda_0 = 1/f'(xm)$, что приводит к простому методу одной касательной и имеет условие сходимости $\lambda_0 * f^{(1)}(x) > 0$. Тогда итерационный процесс выглядит так: $x_{k+1} = x_k - \lambda_0 * f(x_k)$. Условием окончания итераций является достижение нужной точности между предыдущим и следующим значением.

Уравнение №1: φ(x) = cos(x/2)/sin(x/2) - tan(x/2);

Уравнение №2: φ(x) = sqrt(log(x)+3);

Уравнение №3: φ(x) = (cos(x)+sin(x))/3.

1) Рассмотрим можно ли использовать метод итерации на уравнении №2 (ln(x) – x*x +3) на отрезке [1, 2]. Найдем производную и середину отрезка:

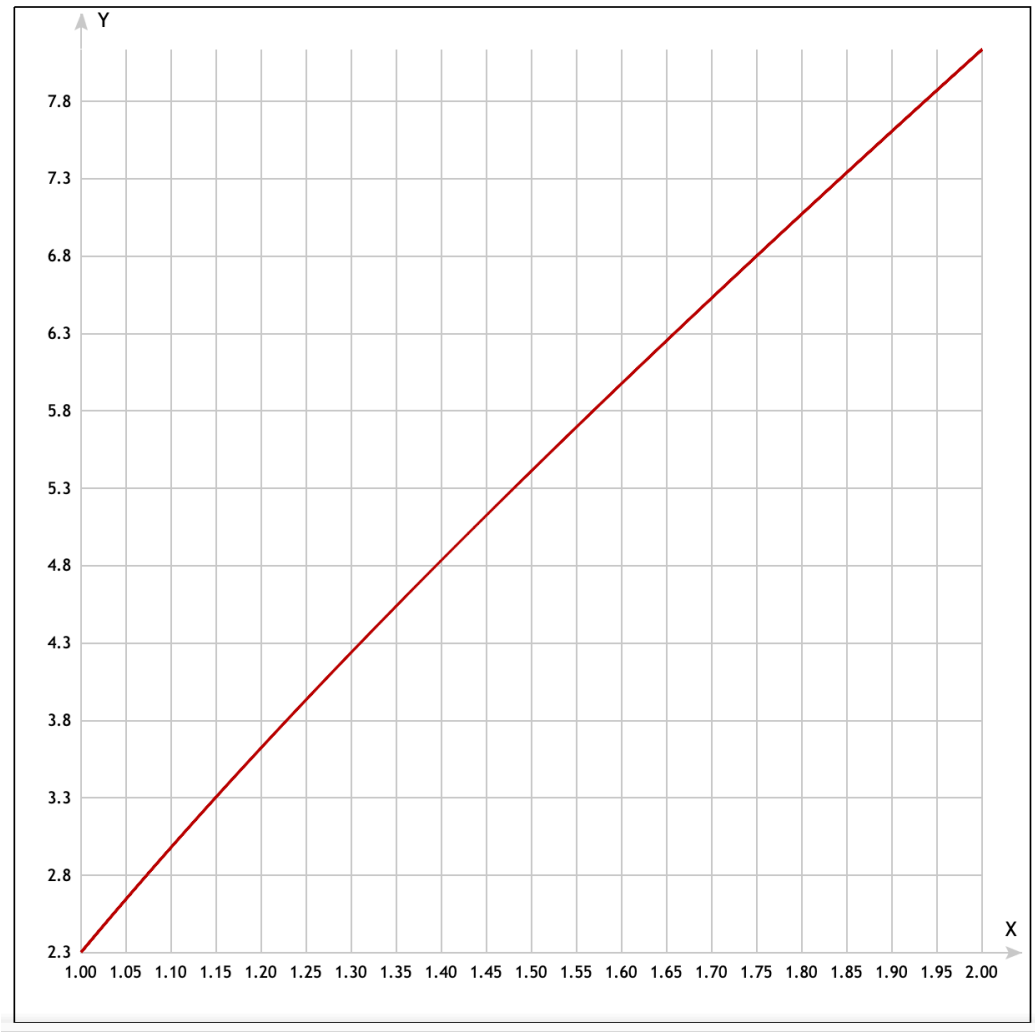
$f^{(1)}(x) = 1/x – 2x$;

$x_m = (1 + 2)/2 = 1.5$;

$\lambda = f^{(1)}(x_M) = 1/1.5 – 2*1.5 = -2.3333$;

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

$y=\lambda * f^{(1)}(x)$ = -2.3333 * (1/x – 2x). Построим график этой функции, проверяя на условие $\lambda * f^{(1)}(x) > 0$:



Условие выполнено, значит метод итераций может быть применен к этой функции.

2) Рассмотрим можно ли использовать метод итерации на уравнении №3 (cos(x) + sin(x) -2x) на отрезке [0, 2]. Найдем эквивалентное уравнение, производную от неё и середину отрезка:

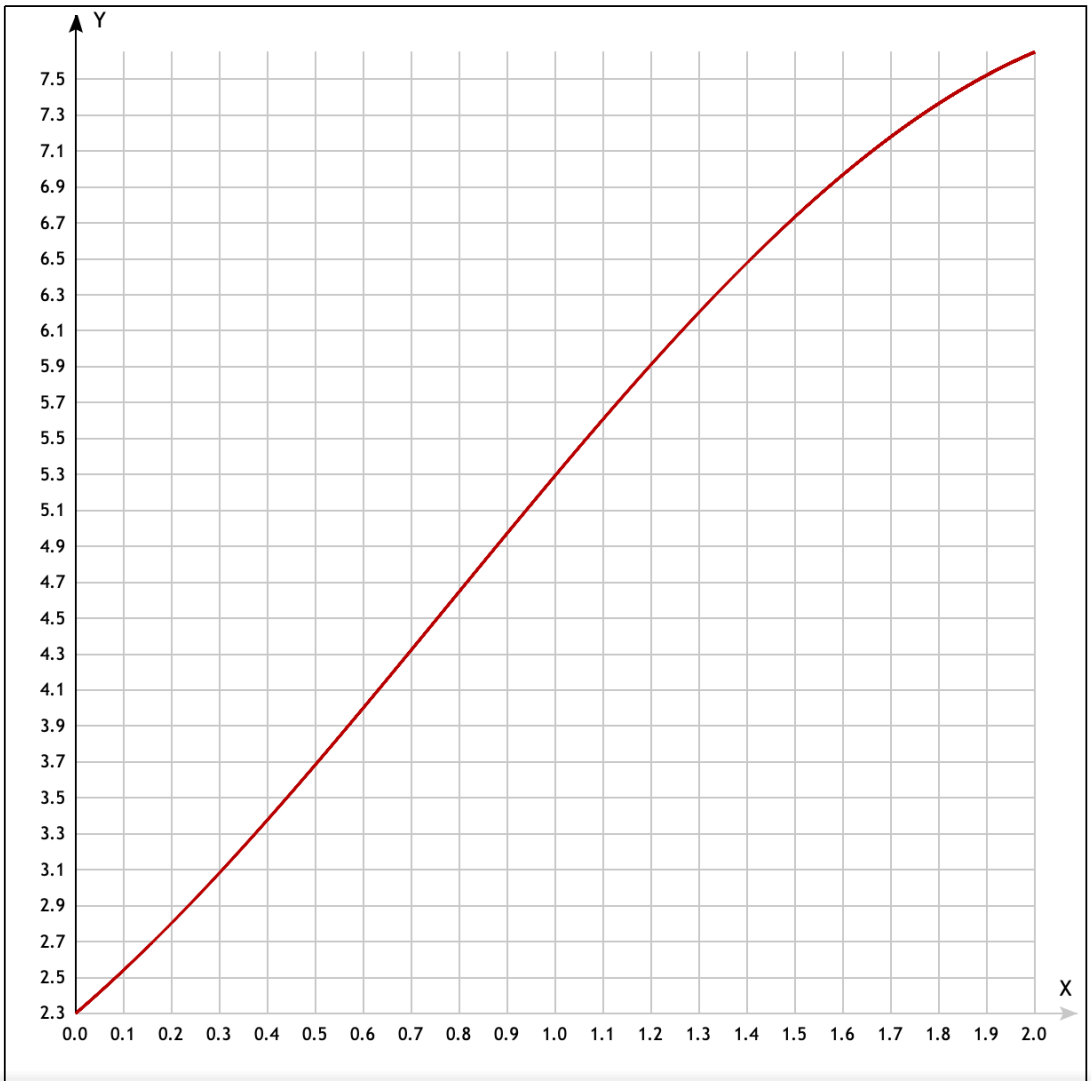
$f^{(1)}(x) = \cos(x) - \sin(x) - 2;$

$x_m = (0 + 2)/2 = 1;$

$\lambda = f^{(1)}(x_M) = \cos(1) - \sin(1) - 2 = -0.540302 - 0.841471 - 2 = -2.30117;$

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

$y=\lambda * f^{(1)}(x)$ = -2.30117 * (cos(x) – sin(x) - 2). Построим график этой функции, проверяя на условие $\lambda * f^{(1)}(x) > 0$:



Условие выполнено, значит метод итераций может быть применен к этой функции.

3) Рассмотрим можно ли использовать метод итерации на уравнении №1 (tg(x/2) – ctg(x/2) + x) на отрезке [1, 2]. Найдем эквивалентное уравнение, производную от неё и середину отрезка:

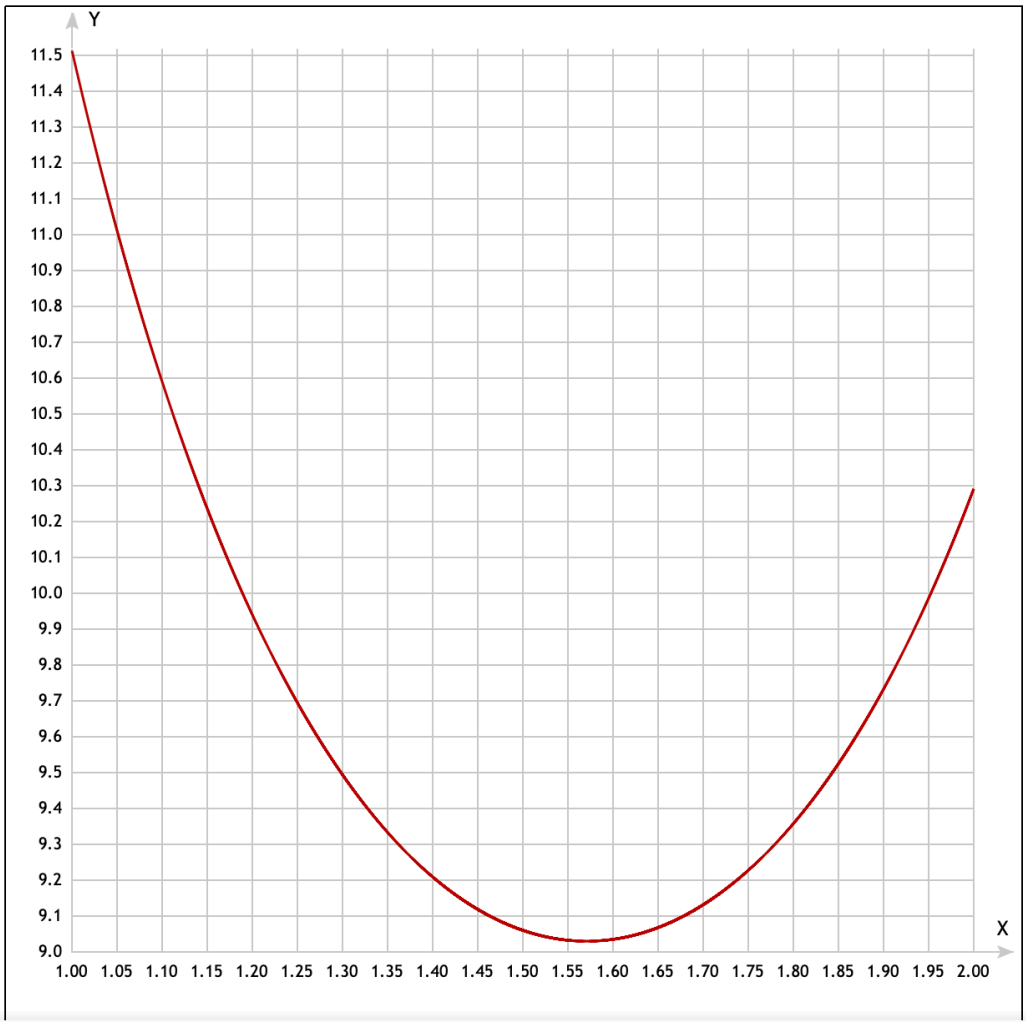
$f^{(1)}(x) = 1 + 1/(2*\cos^2(x/2)) + 1/(2*\sin^2(x/2))$;

$x_m = (1 + 2)/2 = 1.5$;

$\lambda = f^{(1)}(x_M) = 1/(2*\cos^2(3/4)) + 1/(2*\sin^2(3/4)) + 1 = 0.933936 + 1.07612 + 1 = 3.01006$;

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

$y=\lambda * f^{(1)}(x)$ = 3.01006 * (1 + 1/(2*cos^2(x/2)) + 1/(2*sin^2(x/2))). Построим график этой функции, проверяя на условие $\lambda * f^{(1)}(x) > 0$:



Условие выполнено, значит метод итераций может быть применен к этой функции.

3. Метод Ньютона (метод касательных)

Метод Ньютона — итерационный численный метод нахождения корня заданной функции, который является частным случаем метода итераций. А именно за λ_0 берётся значение производной в каждой новой точке. Тогда итерационный процесс имеет вид $x_{(k+1)} = x_{(k)} - f(x_k)/f(1)(x_k)$. Условия окончания итераций и начальное значение абсолютно такие же, как и в методе итерации. Условие сходимости метода можно записать как

$$|f(x) * f^{(2)}(x)| < (f^{(1)}(x))^2$$

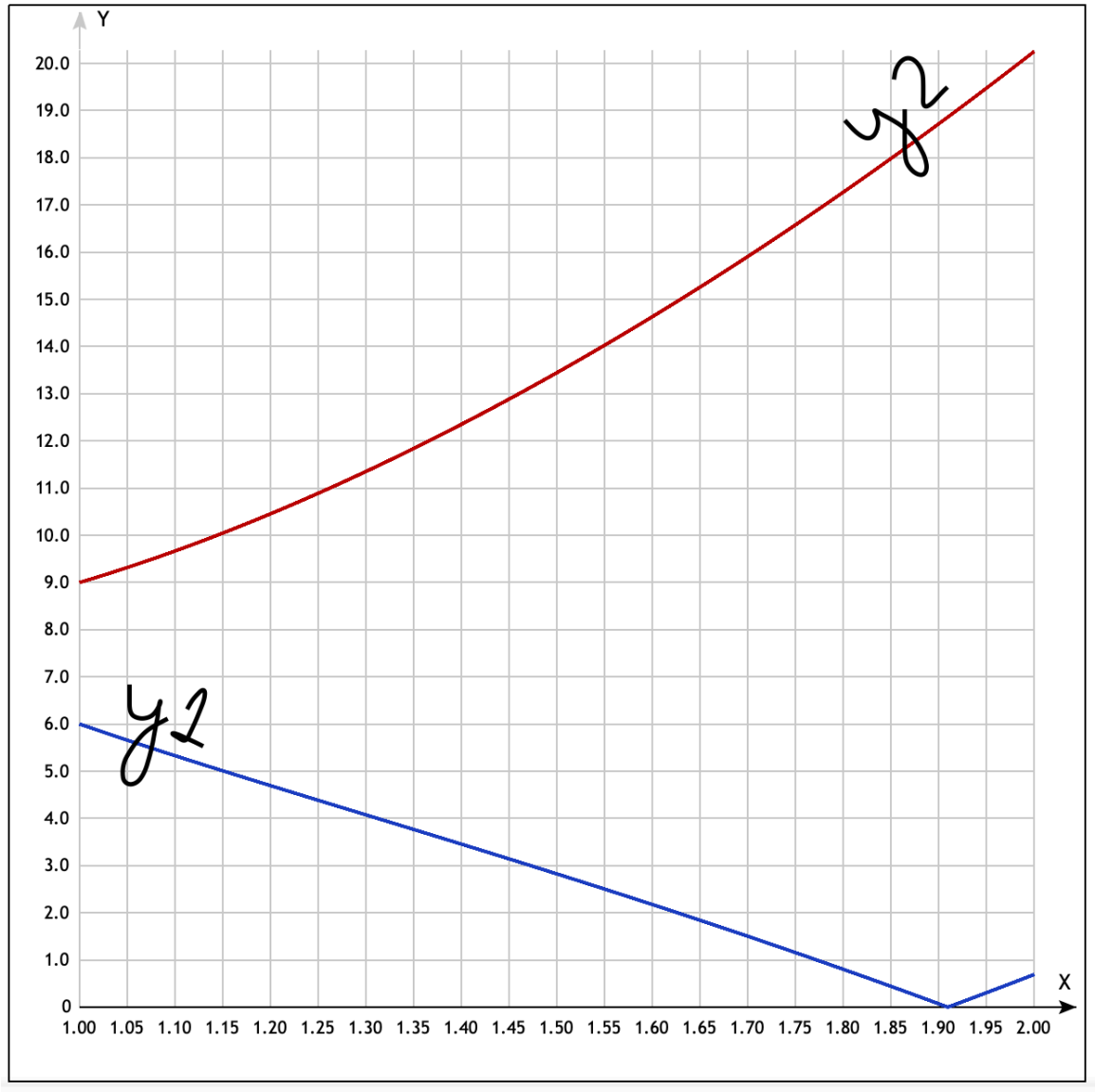
1) Рассмотрим можно ли использовать метод Ньютона на уравнении №2 (ln(x) – x*x +3) на отрезке [1, 2]. Найдем функции y1 = $|f(x) * f^{(2)}(x)|$ и y2 = $(f^{(1)}(x))^2$:

$f^{(1)}(x) = 1/x – 2x;$

$f^{(2)}(x) = -1/x^2 - 2;$

$y1 = |f(x) * f^{(2)}(x)| = |((\ln(x) – x^2 + 3)*(-1/x^2 – 2))|;$
$y2 = (f^{(1)}(x))^2 = 1/x^2 + 4x^2 + 4.$

Проверим условия сходимости для метода Ньютона, построив графики функций y1 и y2, проверяя на условие $|f(x) * f^{(2)}(x)| < (f^{(1)}(x))^2$:



Условие выполнено, значит метод Ньютона может быть применен к этой функции.

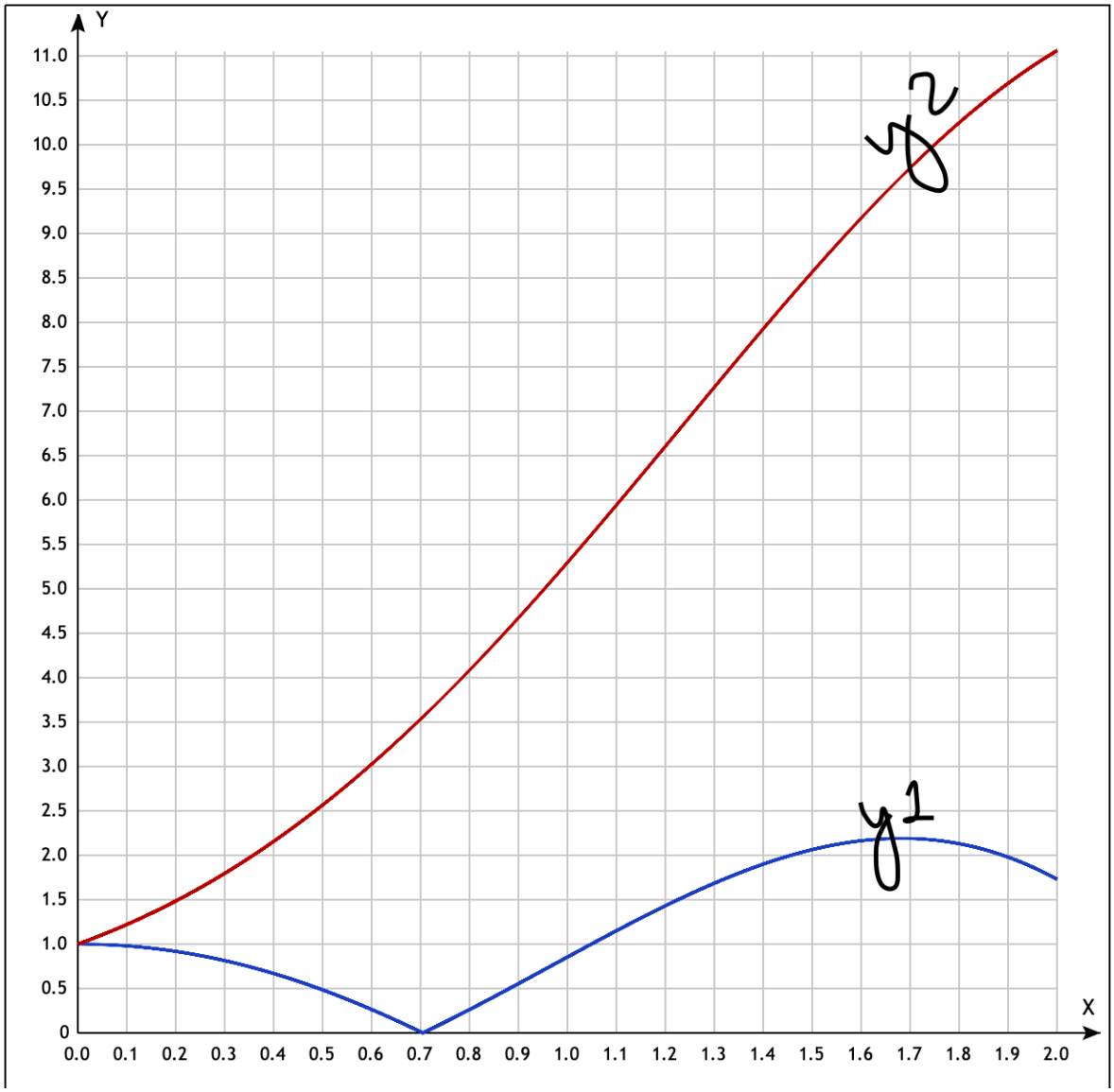
2) Рассмотрим можно ли использовать метод Ньютона на уравнении №3 (cos(x) + sin(x) -2x) на отрезке [0, 2]. Найдем функции y1 = $|f(x) * f^{(2)}(x)|$ и y2 = $(f^{(1)}(x))^2$:

$f^{(1)}(x) = \cos(x) - \sin(x) - 2;$

$f^{(2)}(x) = -\sin(x) - \cos(x);$

$y1 = |f(x) * f^{(2)}(x)| = |((\cos(x) + \sin(x) - 2x)*(-\sin(x) - \cos(x))|;$
$y2 = (f^{(1)}(x))^2 = \cos^2(x) + \sin^2(x) - 2*\cos(x)*\sin(x) - 4\cos(x) + 4\sin(x) + 4.$

Проверим условия сходимости для метода Ньютона, построив графики функций y1 и y2, проверяя на условие $|f(x) * f^{(2)}(x)| < (f^{(1)}(x))^2$:



Условие выполнено, значит метод Ньютона может быть применен к этой функции.

2) Рассмотрим можно ли использовать метод Ньютона на уравнении №1 (tg(x/2) – ctg(x/2) + x) на отрезке [1, 2]. Найдем функции y1 = $|f(x) * f^{(2)}(x)|$ и y2 = $(f^{(1)}(x))^2$:

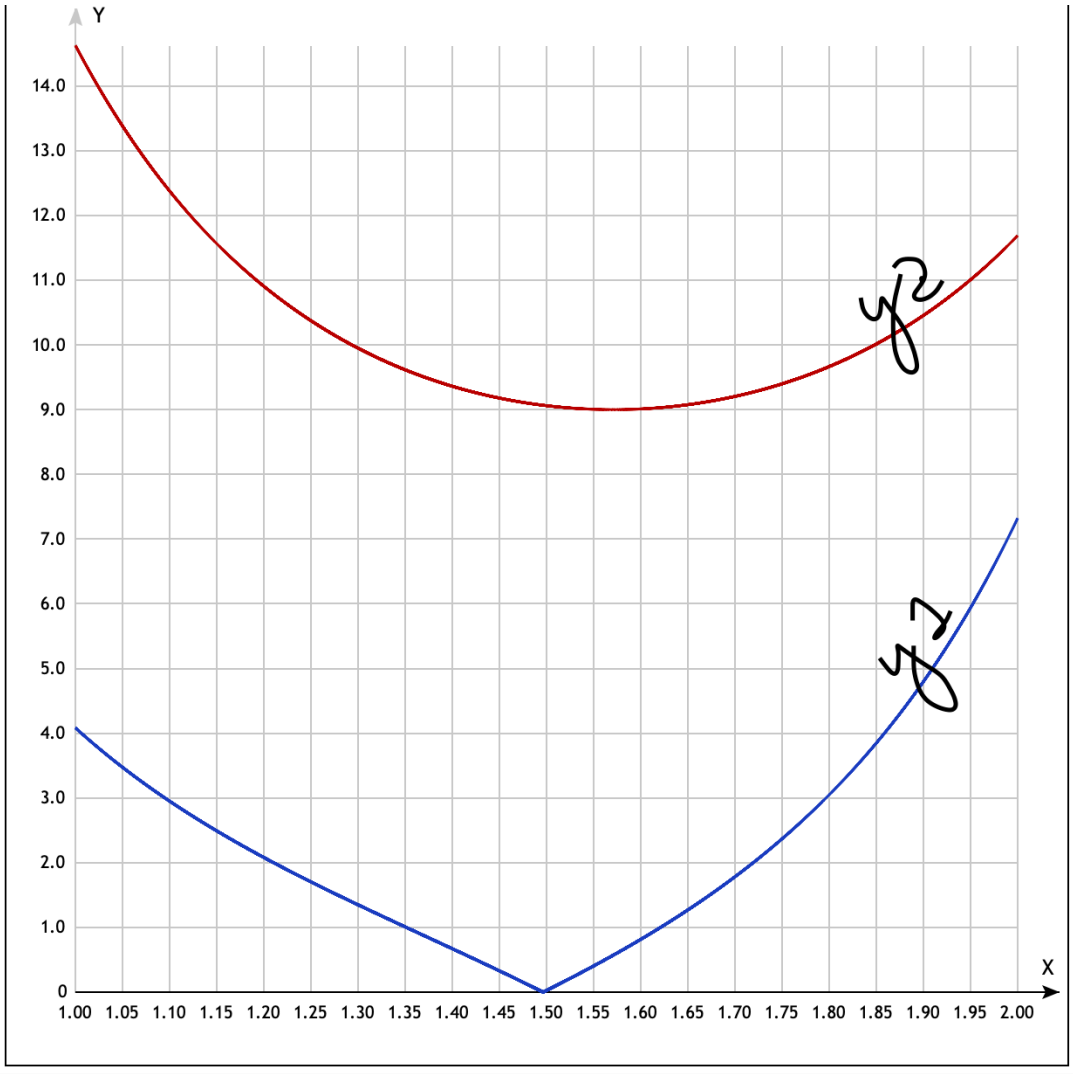
$f^{(1)}(x) = 1 + 1/(2*\cos(x/2)*\cos(x/2)) + 1/(2*\sin(x/2)*\sin(x/2))$;

$f^{(2)}(x) = \sin(x/2)/(2*\cos^3(x/2)) – \cos(x/2)/(2*\sin^3(x/2))$;

$y1 = |f(x) * f^{(2)}(x)| = |((tg(x/2) – ctg(x/2) + x)*(\sin(x/2)/(2*\cos^3(x/2)) – \cos(x/2)/(2*\sin^3(x/2)))|$;

$y2 = (f^{(1)}(x))^2 = 1 + 1/\cos^2(x/2) + 1/\sin^2(x/2) + 1/(4*\cos^4(x/2)) + 1/(2*\cos^2(x/2)*\sin^2(x/2)) + 1/(4*\sin^4(x/2))$.

Проверим условия сходимости для метода Ньютона, построив графики функций y1 и y2, проверяя на условие $|f(x) * f^{(2)}(x)| < (f^{(1)}(x))^2$:



Условие выполнено, значит метод Ньютона может быть применен к этой функции.

4. Метод хорд

Метод хорд (*метод секущих*) - один из методов решения нелинейных уравнений и основан на последовательном сужении интервала, содержащего единственный корень уравнения f(x)=0. Итерационный процесс выполняется до того момента, пока не будет достигнута заданная точность E.

В отличие от метода половинного деления, метод хорд предлагает, что деление рассматриваемого интервала будет выполняться не в его середине, а в точке пересечения хорды с осью абсцисс (ось - X). Следует отметить, что под хордой понимается отрезок, который проведен через точки рассматриваемой функции по концам рассматриваемого интервала. Рассматриваемый метод обеспечивает более быстрое нахождение корня, чем метод половинного деления, при условии задания одинакового рассматриваемого интервала.

Рассмотрим метод деления пополам на уравнении №2 (ln(x) – x*x +3) на отрезке [1, 2]. Также нужно задать погрешность расчета – E.

1) Найдём точку пересечения хорды с осью абцисс:
$f(a_k) = 2$; $f(b_k) = -0.306853$;
$c_k = a_k – f(a_k)/(f(b_k)-f(a_k))*(b_k-a_k) = 1 – 2/(-0.306853–2) * (2–1) = 1.86698$;
$f(c_k) = \ln(1.86698) – 1.86698^2 + 3 = 0.624322 – 3.485614 + 3 = 0.138708$;

2) Необходимо найти значение функции f(x) в точках a_k, b_k и c_k. Далее необходимо проверить два условия:

- если выполняется условие $f(a_k)*f(c_k)<0$, то искомый корень находится внутри левого отрезка положить $a_{k+1} = a_k$, $b_{k+1} = c_k$;

$f(a_k)*f(c_k) = 2*0.138708 = 0.277416 > 0$, следовательно, никаких преобразований не делаем;

- если выполняется условие $f(c_k)*f(b_k)<0$, то искомый корень находится внутри правого отрезка принять $a_{k+1} = c_k$, $b_{k+1} = b_k$;

$f(c_k)*f(b_k) = 0.138708*(-0.306853) = -0.042563 < 0$, следовательно, $a_{k+1} = c_k = 1.86698$, $b_{k+1} = b_k = 2$.

В результате находится новый интервал неопределенности, на котором находится искомых корень уравнения:

$L_{k+1} = [a_{k+1}, b_{k+1}] = [1.86698, 2]$.

3) Проверяем приближенное значение корня уравнения на предмет заданной точности, в случае:

- если разность двух последовательных приближений станет меньше заданной точности $|c_{k+1} – c_k| < E$, то итерационный процесс заканчивается. Приближенное значение корня определяется по формуле:

$$x_* = a_{k+1} - \frac{f(a_{k+1})}{f(b_{k+1}) - f(a_{k+1})} \cdot (b_{k+1} - a_{k+1})$$

- если разность двух последовательных приближений не достигает необходимой точности $|c_{k+1} - c_k| > E$, то необходимо продолжить итерационный процесс (k = k + 1) и перейти к п.2 рассматриваемого алгоритма.

Описание алгоритма

Сперва задаём эпсилон, на котором будет основываться точность вычисления. Опишем каждое из четырёх методов вычисления корня. Для метода дихотомии мы будем сужать отрезок с помощью половинного деления до того момента, как корень будет достигнут. Метод хорд очень похож на метод дихотомии, но сужать отрезок, мы будем с помощью точки пересечения хорд. Метод итераций заключается в сжимании отображения, вычисления тоже начинаются с середины отрезка. Метод Ньютона является частным случаем методом итераций, но здесь мы дополнительно будем высчитывать производную в каждой точке. Последовательно высчитываем корни уравнений всеми четырьмя способами и выводим его на экран.

Описание программы

1) Подключаем библиотеки;

2) Вводим эпсилон как константу;

3) Обозначаем нужные нам четыре функции (для четырех методов), функцию модуля, а также 9 функций – три входных уравнения, три производных от этих уравнений, три эквивалентных функций от этих уравнений;

4) Берем первое уравнение, вызываем от него четыре функции (функции метода дихотомии, итераций, Ньютона, хорд). Выводим на экран возвращенные от них значения. Проделываем это для второго и третьего уравнения;

5) Заполняем функции для входных уравнений, их производных и эквивалентных функций;

6) Функция модуля:
Если x больше нуля, то возвращаем x, иначе возвращаем -x;

7) Функция метода дихотомии:

1. Вычисляем середину отрезка и присваиваем это значение переменной x, присваиваем переменной prevX значение правой границы отрезка;
2. Пока модуль разницы prevX и x больше эпсилон, выполняем данные действия:
    1) Если произведения значений функций от левой границы отрезка и середины отрезка больше нуля, то левая граница становится равной середине отрезка, иначе правая граница становится равной середине отрезка;
    2) Переменная prevX становится равной середине отрезка, x меняет своё значение на значение середины нового отрезка;
3. Возвращаем значение x;

8) Функция метода итераций:

1. Присваиваем значение середины отрезка переменной prevX, x равно prevX+1;
2. Пока модуль разницы x и prevX больше эпсилон, выполняем данные действия:
    1) Переменная prevX становится равной x;
    2) Переменная x становится равной значении функции от x;
3. Возвращаем значение x;

9) Функция метода Ньютона:

1. Присваиваем значение середины отрезка переменной prevX, а переменной x – prevX-f(prevX)/F(prevX);
2. Пока модуль разницы prevX и x больше эпсилон, выполняем данные действия:
    1) Переменная prevX становится равной x;
    2) Переменная x становится равной значению prevX-f(prevX)/F(prevX);
3. Возвращаем значение x;

10) Функция метода хорд:

1. Значение правой границы отрезка присваивается переменной prevX, значение функции от левой границы – переменной ya, значение функции от правой границы – переменной yb;
2. Переменная x становится равной значению (ya*b-yb*a)/(ya-yb) (точка пересечения хорд);
3. Пока модуль разницы prevX и x больше эпсилон, выполняем данные действия:
    1) Если произведение значений переменной ya и функции от x больще нуля, то левая граница становится равной x, иначе правая граница становится равной x;
    2) Переменная ya становится равной значению функции от левой границы отрезка, переменная yb становится равной значению функции от правой границы отрезка, переменная prevX становится равной точке

пересечения хорд, x становится равной новой точке пересечения хорд для нового отрезка;

3. Возвращаем значение x.

Программа

```
#include <stdio.h>
#include <math.h>
#include <locale.h>
const double eps = 0.000001;
double dabs(double);
double dichotomy(double f(double), double, double);
double iteration(double f(double), double, double);
double tangent(double f(double), double F(double), double, double);
double chord(double f(double), double, double);
double f1(double);
double F1(double);
double Fp1(double);
double f2(double);
double F2(double);
double Fp2(double);
double f3(double);
double F3(double);
double Fp3(double);

int main() {
printf("Корень функции f1 методом деления пополам = %.5f\n",
dichotomy(f1,1.,2.));
printf("Корень функции f1 методом итераций = %.5f\n", iteration(F1, 1, 2));
printf("Корень функции f1 методом касательных = %.5f\n",
tangent(f1,Fp1,1.,2.));
printf("Корень функции f1 методом хорд = %.5f\n\n", chord(f1,1.,2.));

printf("Корень функции f2 методом деления пополам = %.5f\n",
dichotomy(f2,1.,2.));
printf("Корень функции f2 методом итераций = %.5f\n", iteration(F2,1, 2));
printf("Корень функции f2 методом касательных = %.5f\n",
tangent(f2,Fp2,1.,2.));
printf("Корень функции f2 методом хорд = %.5f\n\n", chord(f2,1.,2.));

printf("Корень функции f3 методом деления пополам = %.5f\n",
dichotomy(f3,0.,2.));
```

```
printf("Корень функции f3 методом итераций = %.5f\n", iteration(F3,0,2));
printf("Корень функции f3 методом касательных = %.5f\n",
tangent(f3,Fp3,0.,2.));
printf("Корень функции f3 методом хорд = %.5f\n\n", chord(f3,0.,2.));
return 0;}

double f1(double x) { return tan(x/2) - cos(x/2)/sin(x/2) + x; }
double F1(double x) { return cos(x/2)/sin(x/2) - tan(x/2); }
double Fp1(double x) { return 1 + 1/(2*cos(x/2)*cos(x/2)) +
1/(2*sin(x/2)*sin(x/2)); }
double f2(double x) { return log(x) - x*x + 3; }
double F2(double x) { return sqrt(log(x)+3); }
double Fp2(double x) { return 1/x - 2*x; }
double f3(double x) { return cos(x) + sin(x) - 2*x; }
double F3(double x) { return (cos(x)+sin(x))/2; }
double Fp3(double x) { return cos(x) - sin(x) - 2; }
double dabs(double x) { return (x > 0 ? x : -x); }

double dichotomy(double f(double), double a, double b) {
   double prevX = b, x = (a + b) / 2.;
   while(dabs(prevX - x) > eps) {
      if(f(x)*f(a) > 0) a = x; else b = x;
      prevX = x; x = (a + b) / 2.;}
   return x;}

double iteration(double f(double), double a, double b) {
   double prevX = (a+b)/2., x = prevX + 1;
   while(dabs(x-prevX) > eps) {
      prevX = x; x = f(x);}
   return x;}

double tangent(double f(double), double F(double), double a, double b) {
   double prevX = (a+b/2.), x = prevX - f(prevX)/F(prevX);
   while(dabs(prevX - x) > eps) {
      prevX = x; x = prevX - f(prevX)/F(prevX);}
   return x;}

double chord(double f(double), double a, double b) {
   double prevX = b, ya = f(a), yb = f(b);
   double x = (ya*b-yb*a)/(ya-yb);
   while(dabs(prevX - x) > eps) {
      if(ya*f(x) > 0) a = x; else b = x;
      ya = f(a), yb = f(b); prevX = x; x = (ya*b-yb*a)/(ya-yb);}
   return x;}
```

Выходные данные

Программа должна вывести 12 строк. В каждой строке необходимо вывести искомое значение корня. В первых четырёх строках для первого уравнения четырьмя способами, а в следующих четырёх – для второго, в последующих четырёх – для третьего.

Протокол исполнения программы

```
Last login: Tue Dec 21 21:11:50 on ttys000

The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s
/bin/zsh`.
For more details, please visit
https://support.apple.com/kb/HT208050.
MacBook-Pro-MacBook:~ macbookpro$ cat zag.txt
************************************
*   Минеева Светлана Алексеевна    *
*           М8О-105Б-21            *
*        Курсовая работа №4        *
************************************
MacBook-Pro-MacBook:~ macbookpro$ ls -l |tail -4
-rw-r--r--   1 macbookpro  staff     0 21 дек 22:13 f1.out
-rw-r--r--   1 macbookpro  staff  3320 21 дек 22:13 kp4.c
-rw-r--r--   1 macbookpro  staff  2895  7 июл  2020
pslog_20200707_123036.log
-rw-r--r--   1 macbookpro  staff     0  9 дек 19:10 zag.txt
MacBook-Pro-MacBook:~ macbookpro$ cat kp4.c
#include <stdio.h>
#include <math.h>
#include <locale.h>
const double eps = 0.000001;
double dabs(double);
double dichotomy(double f(double), double, double);
double iteration(double f(double), double, double);
double tangent(double f(double), double F(double), double,
double);
double chord(double f(double), double, double);
double f1(double);
double F1(double);
double Fp1(double);
double f2(double);
double F2(double);
double Fp2(double);
double f3(double);
double F3(double);
double Fp3(double);
```

```
int main() {
printf("Корень функции f1 методом деления пополам = %.5f\n",
dichotomy(f1,1.,2.));
printf("Корень функции f1 методом итераций = %.5f\n",
iteration(F1, 1, 2));
printf("Корень функции f1 методом касательных = %.5f\n",
tangent(f1,Fp1,1.,2.));
printf("Корень функции f1 методом хорд = %.5f\n\n",
chord(f1,1.,2.));

printf("Корень функции f2 методом деления пополам = %.5f\n",
dichotomy(f2,1.,2.));
printf("Корень функции f2 методом итераций = %.5f\n",
iteration(F2,1, 2));
printf("Корень функции f2 методом касательных = %.5f\n",
tangent(f2,Fp2,1.,2.));
printf("Корень функции f2 методом хорд = %.5f\n\n",
chord(f2,1.,2.));

printf("Корень функции f3 методом деления пополам = %.5f\n",
dichotomy(f3,0.,2.));
printf("Корень функции f3 методом итераций = %.5f\n",
iteration(F3,0,2));
printf("Корень функции f3 методом касательных = %.5f\n",
tangent(f3,Fp3,0.,2.));
printf("Корень функции f3 методом хорд = %.5f\n\n",
chord(f3,0.,2.));
return 0;}

double f1(double x) { return tan(x/2) - cos(x/2)/sin(x/2) + x; }
double F1(double x) { return cos(x/2)/sin(x/2) - tan(x/2); }
double Fp1(double x) { return 1 + 1/(2*cos(x/2)*cos(x/2)) +
1/(2*sin(x/2)*sin(x/2)); }
double f2(double x) { return log(x) - x*x + 3; }
double F2(double x) { return sqrt(log(x)+3); }
double Fp2(double x) { return 1/x - 2*x; }
double f3(double x) { return cos(x) + sin(x) - 2*x; }
double F3(double x) { return (cos(x)+sin(x))/2; }
double Fp3(double x) { return cos(x) - sin(x) - 2; }
double dabs(double x) { return (x > 0 ? x : -x); }

double dichotomy(double f(double), double a, double b) {
    double prevX = b, x = (a + b) / 2.;
    while(dabs(prevX - x) > eps) {
        if(f(x)*f(a) > 0) a = x; else b = x;
        prevX = x; x = (a + b) / 2.;}
    return x;}

double iteration(double f(double), double a, double b) {
    double prevX = (a+b)/2., x = prevX + 1;
    while(dabs(x-prevX) > eps) {
        prevX = x; x = f(x);}
```

```
    return x;}

double tangent(double f(double), double F(double), double a,
double b) {
    double prevX = (a+b/2.), x = prevX - f(prevX)/F(prevX);
    while(dabs(prevX - x) > eps) {
        prevX = x; x = prevX - f(prevX)/F(prevX);}
    return x;}

double chord(double f(double), double a, double b) {
    double prevX = b, ya = f(a), yb = f(b);
    double x = (ya*b-yb*a)/(ya-yb);
    while(dabs(prevX - x) > eps) {
        if(ya*f(x) > 0) a = x; else b = x;
        ya = f(a), yb = f(b); prevX = x; x = (ya*b-yb*a)/(ya-
yb);}
    return x;}
MacBook-Pro-MacBook:~ macbookpro$ gcc kp4.c -o f1.out
MacBook-Pro-MacBook:~ macbookpro$ ./f1.out
Корень функции f1 методом деления пополам = 1.07687
Корень функции f1 методом итераций = -1.07687
Корень функции f1 методом касательных = 1.07687
Корень функции f1 методом хорд = 1.07687

Корень функции f2 методом деления пополам = 1.90970
Корень функции f2 методом итераций = 1.90970
Корень функции f2 методом касательных = 1.90970
Корень функции f2 методом хорд = 1.90970

Корень функции f3 методом деления пополам = 0.70481
Корень функции f3 методом итераций = 0.70481
Корень функции f3 методом касательных = 0.70481
Корень функции f3 методом хорд = 0.70481

MacBook-Pro-MacBook:~ macbookpro$
```

Заключение

Дневник отладки:

№	Лаб. или дом.	Дата	Время	Событие	Действие по исправлению	Примечание
1	Дом.	21.12.21	20:50	Ошибка работы метода итераций	Ошибка исправлена, строка изменена на x = prevX + 1	Сначала строка выглядела так: x = f(prevX)

Вывод:

В работе описаны идеи и принципы четырёх численных методов: дихотомии, итераций, Ньютона и хорд. 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 вывода, проведено тестирование программы, составлен протокол исполнения программы.

Список литературы

1) Метод хорд: https://simenergy.ru/math-analysis/solution-methods/42-chord-method

2) Метод итераций: https://ru.wikipedia.org/wiki/Метод_простой_итерации

3) Метод Ньютона: https://ru.wikipedia.org/wiki/Метод_Ньютона

4) Метод дихотомии: https://ru.wikipedia.org/wiki/Метод_бисекции