Московский Авиационный институт

(государственный технический университет)

**Факультет прикладной математики и информационных технологий**

Кафедра вычислительной математики и программирования

# Курсовая работа

**По курсу**

**«Информатика»**

**I семестр**

**Задание IV. Процедуры и функции в качестве параметров**

Студент: Арешин Станислав Олегович

Группа: М8О-106Б, №1 по списку

Руководитель: Дубинин А.В.

Оценка:

Дата:

**Москва, 2017**

# Задача

Составить программу на языке Си с процедурами решения трансцендентных алгебраических уравнений различными численными методами (итерации, 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, начиная с варианта с заданным номером. Если метод неприменим, дать математическое обоснование и графическую иллюстрацию.

# Варианты

Варианты №22,23

| | | | | |
|---|---|---|---|---|
| 22 | $\arccos x - \sqrt{1 - 0{,}3x^3} = 0$ | [0, 1] | итераций | 0.5629 |
| 23 | $3x - 4\ln x - 5 = 0$ | [2, 4] | Ньютона | 3.23 |

# Теоритическая часть

Пусть имеется уравнение вида

$$\mathbf{f(x)= 0}$$

где **f(x)** - заданная алгебраическая или трансцендентная функция.

***Решить уравнение*** - значит найти все его корни, то есть те значения **x**, которые обращают уравнение в тождество.

Если уравнение достаточно сложно, то задача точного определения корней является в некоторых случаях нерешаемой. Поэтому ставится задача найти такое приближенное значение корня $\mathbf{x_{ПР}}$, которое отличается от точного значения корня **x***на величину, по модулю не превышающую указанной точности (малой положительной величины) **ε**, то есть

$$| \mathbf{x^* – x_{пр}} | < \boldsymbol{\varepsilon}$$

Величину **ε** также называют ***допустимой ошибкой***, которую можно задать по своему усмотрению.

### Этапы приближенного решения нелинейных уравнений

Приближенное решение уравнения состоит из двух этапов:

- Отделение корней, то есть нахождение интервалов из области определения функции **f(x)**, в каждом из которых содержится только один корень уравнения **f(x)=0**.
- Уточнение корней до заданной точности.

## Отделение корней

Отделение корней можно проводить графически и аналитически.

### Графическое отделение корней

Для того чтобы графически отделить корни уравнения, необходимо построить график функции **f(x)**. Абсциссы точек его пересечения с осью Ох являются действительными корнями уравнения.

### Аналитическое отделение корней

Аналитическое отделение корней основано на следующих теоремах.

***Теорема 1***. Если непрерывная функция **f(x)** принимает на концах отрезка [a; b]значения разных знаков, т.е.

$$f(a) \cdot f(b) < 0$$

то на этом отрезке содержится по крайней мере один корень уравнения.

***Теорема 2***. Если непрерывная на отрезке [a; b] функция **f(x)** принимает на концах отрезка значения разных знаков, а производная **f'(x)** сохраняет знак внутри указанного отрезка, то внутри отрезка существует единственный корень уравнения **f(x) = 0**.

## Уточнение корней

Для ***уточнения корней*** может использоваться один из следующих методов:

- Метод половинного деления (метод дихотомии)
- Метод последовательных приближений (метод итераций)
- Метод Ньютона (метод касательных)

# Метод дихотомии (метод половинного деления)

Метод заключается в делении отрезка пополам и его сужении в два раза на каждом шаге итерационного процесса в зависимости от знака функции в середине отрезка.Такой подход обеспечивает гарантированную сходимость метода независимо от сложности функции - и это весьма важное свойство. Недостатком метода является то же самое -

метод никогда не сойдется быстрее, т.е. сходимость метода всегда равна сходимости в наихудшем случае.

Метод половинного деления:

1. Один из простых способов поиска *корней функции одного аргумента*.
2. Применяется для нахождения *значений действительно-значной функции*, определяемому по какому-либо критерию (это может быть сравнение на *минимум*, *максимум* или конкретное число).

Очевидно, что если на отрезке [a, b] существует корень уравнения, то значения функции на концах отрезка имеют разные знаки: $F(a) \cdot F(b) < 0$ (***Теорема 1***).

## Описание метода

За начальное приближение принимаются границы исходного отрезка $a_0 = a$, $b_0 = b$.

Итерационный процесс:

1) $a_{k+1} = (a_k + b_k) / 2$, $b_{k+1} = b_k$ , если $F(a_k)*F((a_k + b_k) / 2) > 0$;

2) $a_{k+1} = a_k$, $b_{k+1} = (a_k + b_k) / 2$, если $F(b_k)* F((a_k + b_k) / 2) > 0$.

Условие окончания: $|a_k - b_k| < \varepsilon$.

Приближенное значение корня: $x^* \approx (a_{\text{конечное}} + b_{\text{конечное}}) / 2$.

# Метод итераций

**Метод итерации** — численный метод решения математических задач, приближённый метод решения системы линейных алгебраических уравнений. Суть такого метода заключается в нахождении по приближённому значению величины следующего приближения (являющегося более точным). Метод позволяет получить значения корней системы с заданной точностью в виде предела последовательности некоторых векторов (итерационный процесс). Характер сходимости и сам факт сходимости метода зависит от выбора начального приближения корня $x_0$.

Идея метода заключается в замене исходного уравнения $F(x) = 0$ уравнением вида $x = f(x)$.

### Достаточное условие

Достаточное условие сходимости метода $|f'(x)| < 1$, $x \in [a,b]$. Это условие необходимо проверить перед началом решения задачи, так как функция $f(x)$ может быть выбрана неоднозначно, причем в случае неверного выбора указанной функции метод расходится.

### Описание метода

Начальное приближение корня: $x_0 = (a + b) / 2$.

Итерационный процесс: $x_{k+1} = f(x_k)$.

Условие окончания: $|x_k - x_{k-1}| < \varepsilon$.

Приближенное значение корня: $x^* \approx x_{\text{конечное}}$.

# Метод Ньютона

**Метод Ньютона**, **алгоритм Ньютона** (также известный как **метод касательных**) — это итерационный численный метод нахождения корня (нуля) заданной функции. Метод был впервые предложен английским физиком, математиком и астрономом Исааком Ньютоном (1643—1727). Поиск решения осуществляется путём построения последовательных приближений и основан на принципах простой итерации. Метод обладает квадратичной сходимостью.

Метод Ньютона является частным случаем метода итераций.

### Условие сходимости метода:

$$|F(x) \cdot F''(x)| < (F'(x))^2 \text{ на отрезке } [a, b].$$

### Итерационный процесс:

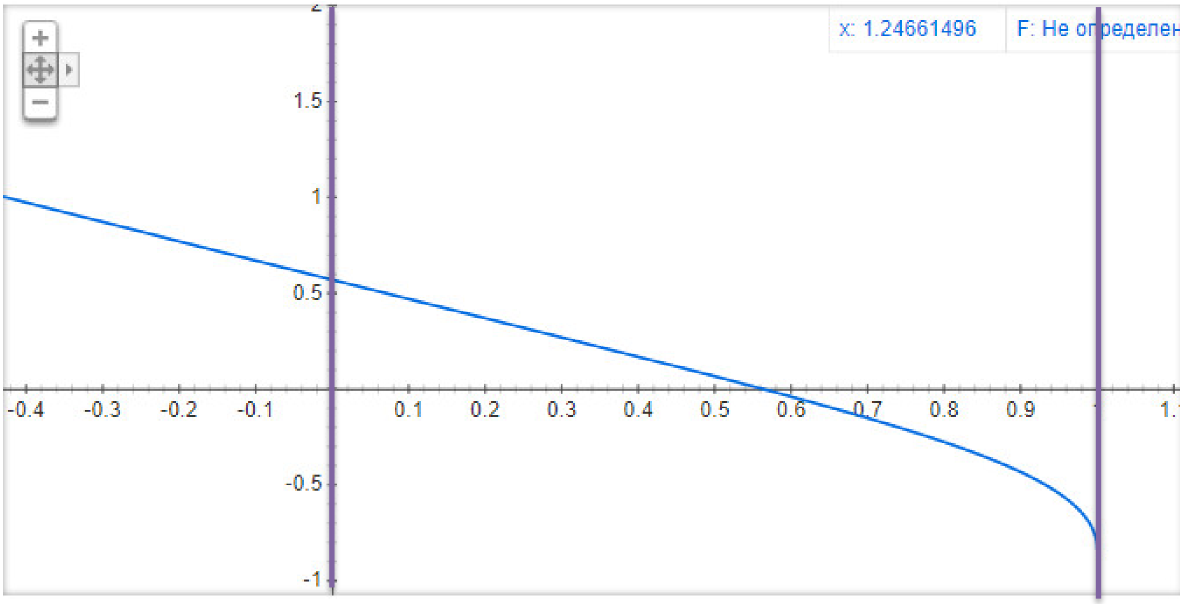
$$x_{k+1} = x_k - F(x_k)/F'(x_k).$$

# Практическая часть

## Исследование первого уравнения  F1  (вариант 22)

### 1)Построим график функции F1(x):

График функции arccos(x)-sqrt(1-0.3*x^3)




По графику функции F1 видно, что условие $f(a)\cdot f(b)<0$ на отрезке [a,b], где a=0,b=1 выполняется, следовательно, метод дихотомии применим к этой функции.

### 2) Выразим x из F1 для метода итераций:

arccos(x) –sqrt(1-0.3*x^3)=0

cos(arccos(x))=cos(sqrt(1-0.3*x^3))

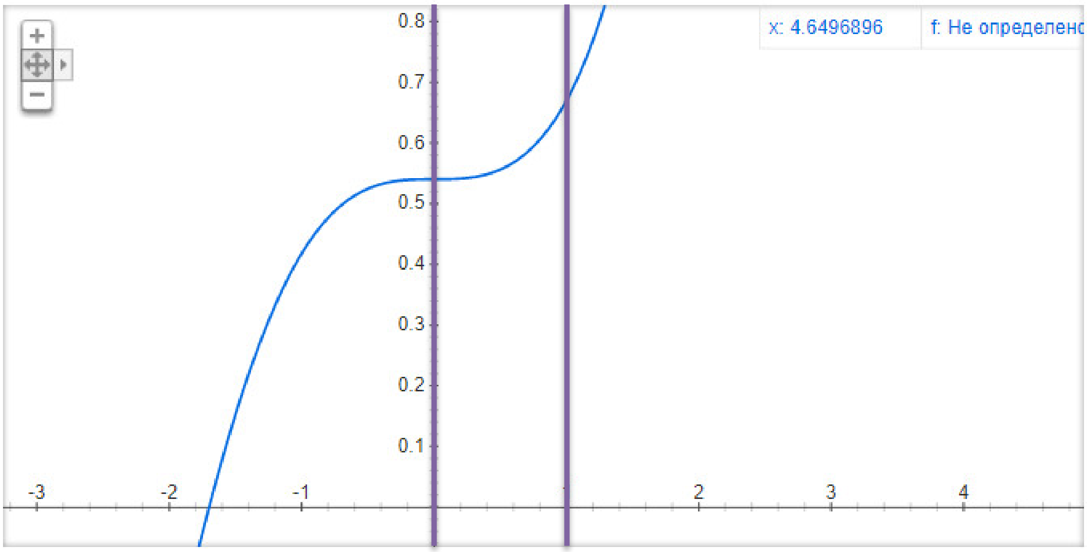
x=cos(sqrt(1-0.3*x^3))

Таким образом, f1(x)=cos(sqrt(1-0.3*x^3))

**Построим график функции f1(x):**

По графику мы видим, что функция f1(x) непрерывная, возрастает на отрезке [a,b], где a=0,b=1.

## График функции cos(sqrt(1-0.3*x^3))




**Проверим условие сходимости метода итераций:**

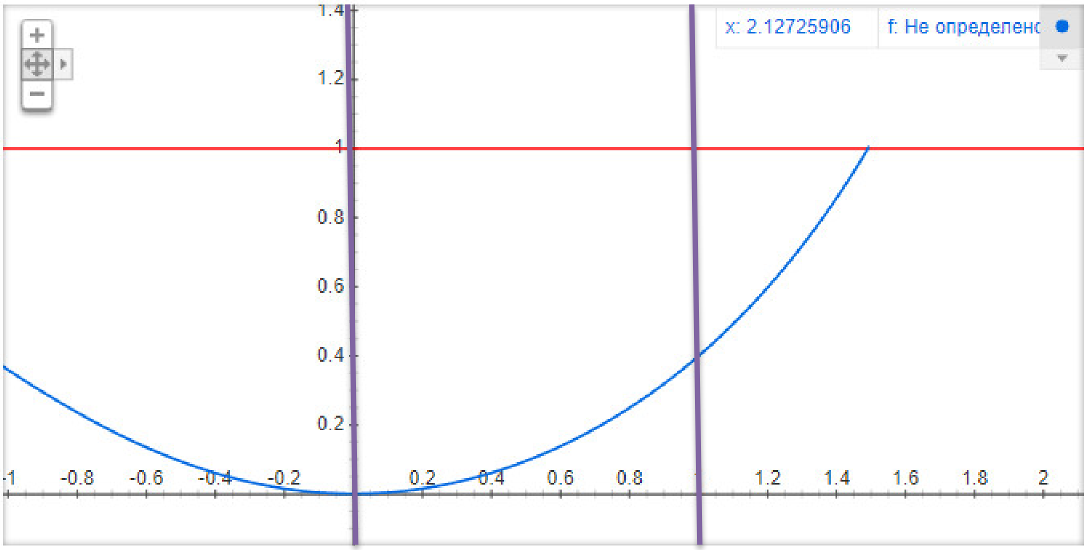
Для этого необходимо посчитать первую производную f1' (x) функции f1(x):

**derivative of cos(sqrt(1-0.3*x^3))**

$$\frac{d}{dx}\left(\cos\left(\sqrt{1-0.3\,x^3}\right)\right)=\frac{0.45\,x^2\sin\left(\sqrt{1-0.3\,x^3}\right)}{\sqrt{1-0.3\,x^3}}$$

**Построим в одной плоскости графики |f1' (x)| и y=1:**

## Графики функций 0.45*x^2*sin(sqrt(1-0.3*x^3))/sqrt(1-0.3*x^3), y=1



По графикам видно, что условие $|f1'(x)| < 1$, $x \in [a,b]$, $a=0, b=1$ выполняется, следовательно метод итераций для этого уравнения сходится.

**3) Найдем первую F1' (x) и вторую F1'' (x) производные функции F1(x):**

**F1' (x)**

**derivative of arccos(x)-sqrt(1-0.3*x^3)**

Derivative:

$$\frac{d}{dx}\left(\cos^{-1}(x) - \sqrt{1-0.3\,x^3}\right) = \frac{0.45\,x^2}{\sqrt{1-0.3\,x^3}} - \frac{1}{\sqrt{1-x^2}}$$

**F1'' (x)**

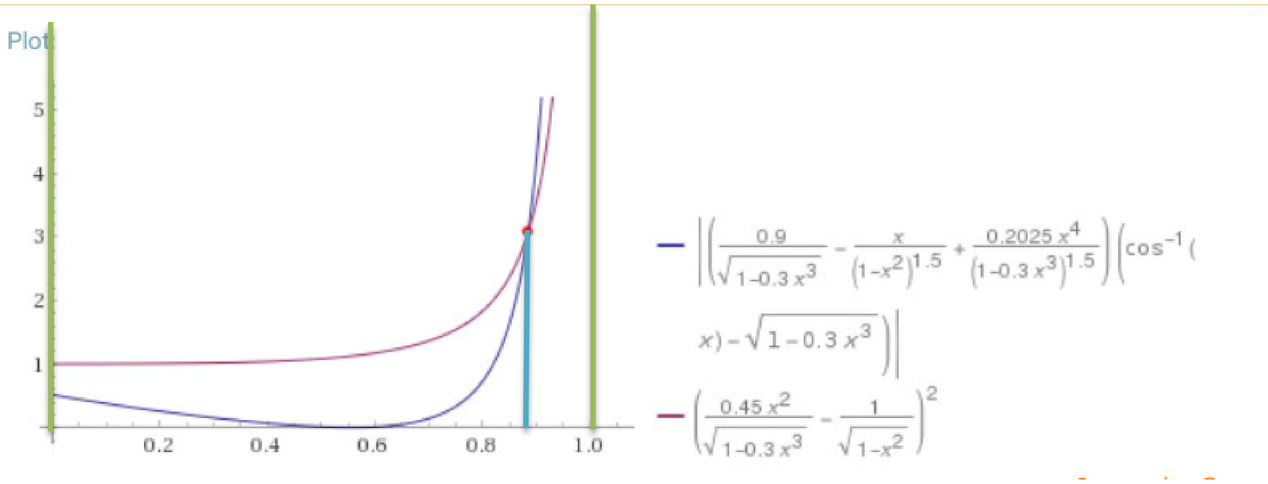
**derivative of (0.45*x^2)/(sqrt(1-0.3*x^3)) - 1/(sqrt(1-x^2))**

Derivative:

$$\frac{d}{dx}\left(\frac{0.45\,x^2}{\sqrt{1-0.3\,x^3}} - \frac{1}{\sqrt{1-x^2}}\right) = \frac{0.9\,x}{\sqrt{1-0.3\,x^3}} - \frac{x}{(1-x^2)^{3/2}} + \frac{0.2025\,x^4}{(1-0.3\,x^3)^{3/2}}$$

**Проверим условие сходимости метода Ньютона:**

**Для этого построим в одной плоскости два графика $y1=|F1(x) \cdot F1''(x)|$ и $y2=(F1'(x))^2$ :**



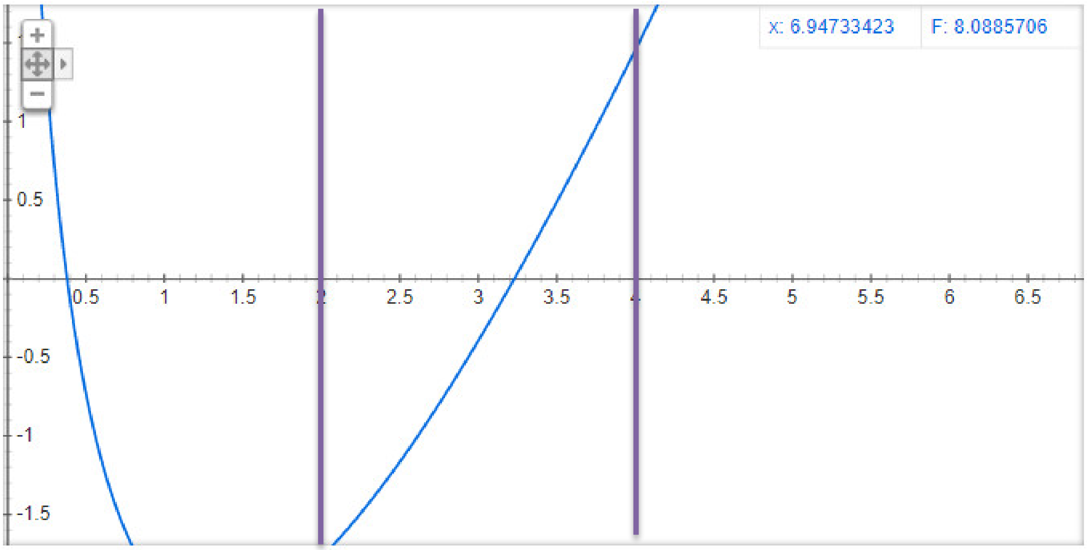

X0

По графику видно, что условие $|F1(x) \cdot F1''(x)| < (F1'(x))^2$ выполняется не на всем отрезке $[a,b]$, $a=0, b=1$. В точке X0 графики y1 и y2 пересекаются, точка $X0 \approx 0.9$, следовательно, метод сходится на отрезке $[0,X0)$, а корень уравнения $x \approx 0.5629 \in [0,X0)$, тогда мы соответственно можем найти этот корень.

# Исследование первого уравнения F2 (вариант 23)

## 1)Построим график функции F2(x):

График функции 3*x-4*ln(x)-5




По графику функции F2 видно, что условие $f(a) \cdot f(b) < 0$ на отрезке [a,b], где a=2,b=4 выполняется, следовательно, метод дихотомии применим к этой функции.

## 2) Выразим x из F2 для метода итераций:

3*x-4*ln(x)-5=0

3*x-4*ln(x)=5

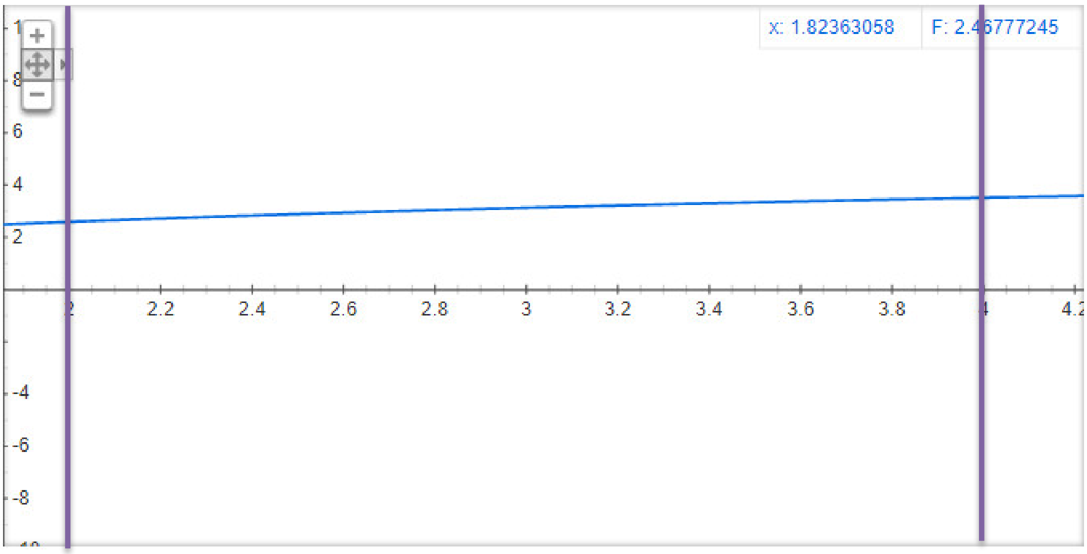
x=(4*ln(x)+5)/3

Таким образом, f2(x)=(4*ln(x)+5)/3

**Построим график функции f(x):**

По графику мы видим, что функция f(x) непрерывная, возрастает на отрезке [a,b], где a=2,b=4.

## График функции (4*ln(x)+5)/3




**Проверим условие сходимости метода итераций:**

Для этого необходимо посчитать первую производную f2' (x) функции f2(x):

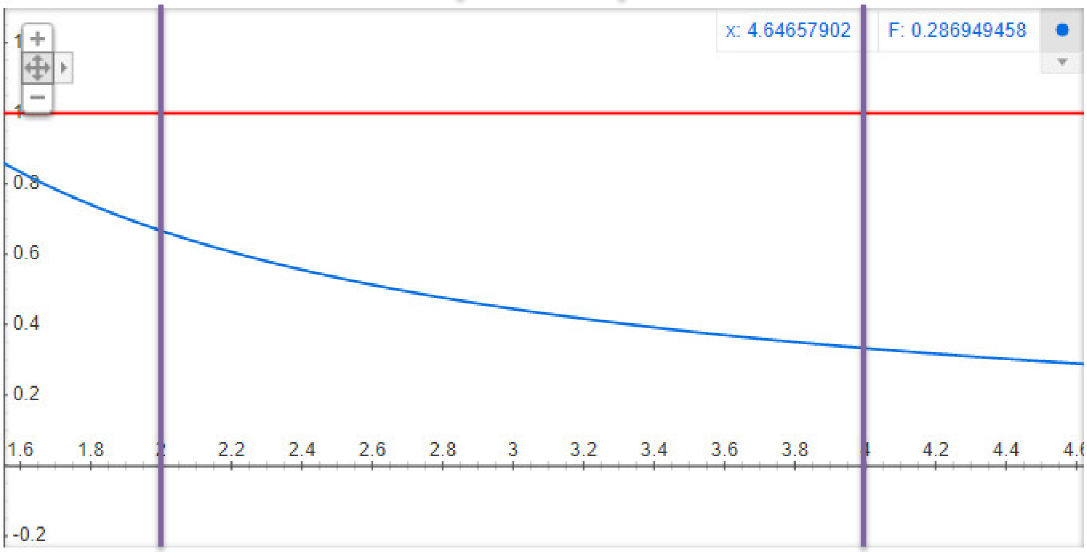
**derivative of (4*ln(x)+5)/3**

Derivative:

$$\frac{d}{dx}\left(\frac{1}{3}(4\log(x)+5)\right)=\frac{4}{3x}$$

**Построим в одной плоскости графики |f2' (x)| и y=1:**

## Графики функций |4/(3*x)|, y=1




По графикам видно, что условие |f2' (x)| < 1, x ∈ [a,b], a=2,b=4 выполняется, следовательно, метод итераций для этого уравнения сходится.

**3) Найдем первую F2' (x) и вторую F2" (x) производные функции F2(x):**

**F2' (x):**

**derivative of 3*x-ln(x)-5**

Derivative:

$$\frac{d}{dx}(3\,x - 4\log(x) - 5) = 3 - \frac{4}{x}$$

**F2" (x):**

**derivative of 3-4/x**

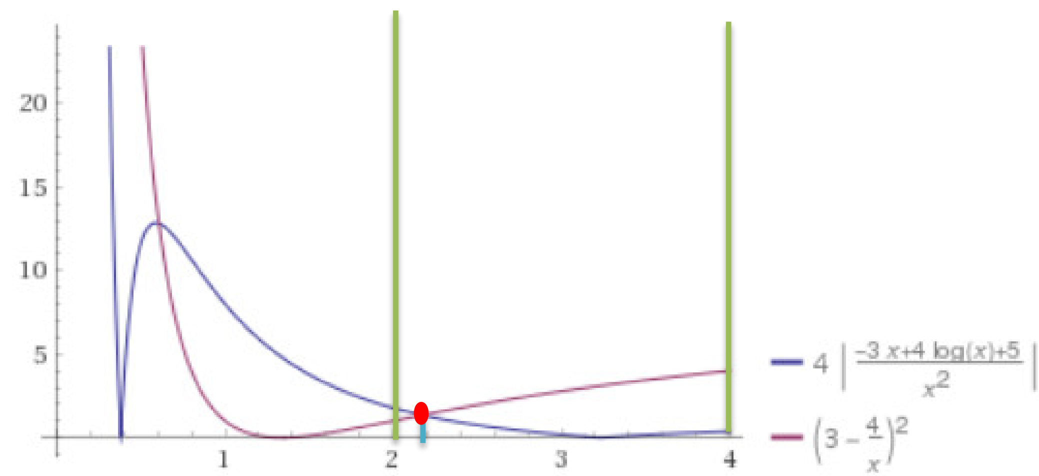
Derivative:

$$\frac{d}{dx}\left(3 - \frac{4}{x}\right) = \frac{4}{x^2}$$

**Проверим условие сходимости метода Ньютона:**

**Для этого построим в одной плоскости два графика y1=|F2(x) · F2''(x)|  и y2=(F2'(x))$^2$ :**

Plot:




**X1**

По графику видно, что условие |F2(x) · F2''(x)| < (F2'(x))$^2$ выполняется не на всем отрезке [a,b], a=0,b=1. В точке X0 графики y1 и y2 пересекаются,  точка X1 ≈ 2.2, следовательно, метод сходится на отрезке (X1,4], а корень уравнения x ≈ 3.23 ∈ (X1,4], тогда мы соответственно можем найти этот корень.

# Программа на СИ

## Описание

Функция double eps вычисляет машинный эпсилон.

Функции double F1, double f1, double F2, double f2 возвращают значение функций F1(x), f1(x), F2(x), f2(x) соответственно в произвольной точке $x \in [a,b]$.

Программа сама вычисляет первую и вторую производные заданных функций в произвольной точке $x \in [a,b]$:

Функция double der1 вычисляет первую производную функции;

Функция double der2 вычисляет вторую производную функции.

Функции double dih, double itter, double newton вычисляют корень уравнения методами дихотомии, итераций, Ньютона, если метод сходится.

Программа сама проверяет условия схождения метода в теле функции, отвечающей за него.

Также программа высчитывает количество итераций, за которое был найден корень тем или иным методом – переменная count.

## Код на СИ

```
#include <stdio.h>
#include <math.h>

int count;

double eps(){
	double a=1;
	while (1+a/2>1){
		a=a/2;
	}
	return(a);
}

double F1 (double x){
	return(acos(x)-sqrt(1-0.3*pow(x,3)));
}
```

```
double f1 (double x){
        return(cos(sqrt(1-0.3*pow(x,3))));
}

double F2 (double x){
        return(3*x-4*log(x)-5);
}

double f2 (double x){
        return((4*log(x)+5)/3);
}

double der1 (double(*F)(double),double x){
        double dx=pow(2,-26);
        return(((*F)(x+dx)-(*F)(x-dx))/(2*dx));
}

double der2 (double(*F)(double),double x){
       double dx=pow(2,-13);
           return(((*F)(x+dx)-2*(*F)(x)+(*F)(x-dx))/(pow(dx,2)));
}

double dih (double(*F)(double),double a, double b){
        double x,e=eps();
        int f,k=0;
           if ((*F)(a)*(*F)(b)<0){
                   f=0;
           while (fabs(a-b)>e){
                   x=(a+b)/2.0;
                   if ((*F)(a)*(*F)(x)<0){
                           b=x;
                   }
                   else{
                           a=x;
```

```
                    }
                k++;
            }
            }
            else{
                f=1;
            }
            if (f==0){
                count=k;
            return(x);
            }
            else{
                return(a-1);
            }
}

double itter (double(*F)(double),double a, double b){
    double x,x2,e=eps();
    int f,k=0;
    x=(a+b)/2.0;
    x2=a;
    while (fabs(x-x2)>e){
            if (fabs(der1((*F),x))<1){
                f=0;
            x2=x;
            x=(*F)(x);
                k++;
            }
            else{
                f=1;
                break;
            }
    }
        if (f==0){
            count=k;
```

```
            return(x);
        }
        else{
            return(a-1);
        }
}

double newton (double(*F)(double),double a,double b){
    double x,x2,e=eps(),dr1,dr2;
    int f,k=0;
    x=(a+b)/2.0;
    x2=a;
    while (fabs(x-x2)>e){
            dr1=der1((*F),x);
            dr2=der2((*F),x);
            if (fabs((*F)(x)*dr2)<pow(dr1,2)){
                f=0;
            x2=x;
                x=x-(*F)(x)/dr1;
            k++;
            }
            else{
                f=1;
                break;
            }
    }
        if (f==0){
            count=k;
            return(x);
        }
        else{
            return(a-1);
        }
}
```

```
int main(){
        double a1,b1,a2,b2;
        printf("Mashinnoe eps = ");
        printf("%e\n",eps());
        printf("vvedite otrezok var22\n");
        scanf("%lf%lf",&a1,&b1);
        printf("vvedite otrezok var23\n");
        scanf("%lf%lf",&a2,&b2);
        printf("                    Tablica znacheniy                    \n");
        printf("|var|  metod dihotomii   |  metod itteraciy   |   metod newtona    |\n");
        printf("|___|    koren'    |itter|shojd| koren' |itter|shojd| koren' |itter|\n");
        printf("|22 |");
        if (dih(F1,a1,b1)==a1-1){
           printf("     no       |  -  |");
      }
      else{
           printf("   %lf   |",dih(F1,a1,b1));
           printf(" %d  |",count);
      }
        if (itter(f1,a1,b1)==a1-1){
                printf(" no  |   -    |  -  |");
        }
        else{
                printf(" yes |");
                printf("%lf|",itter(f1,a1,b1));
                printf(" %d  |",count);
        }
        if (newton(F1,a1,b1)==a1-1){
           printf(" no  |   -    |  -  |\n");
      }
      else{
           printf(" yes |");
           printf("%lf|",newton(F1,a1,b1));
           printf("  %d  |\n",count);
      }
```

```
    printf("|23 |");
  if (dih(F2,a1,b1)==a2-1){
       printf("    no       |  -  |");
  }
  else{
      printf("   %lf   |",dih(F2,a2,b2));
      printf(" %d  |",count);
  }
  if (itter(f2,a2,b2)==a2-1){
      printf(" no  |   -    |  -  |");
  }
  else{
      printf(" yes |");
      printf("%lf|",itter(f2,a2,b2));
      printf(" %d  |",count);
  }
  if (newton(F2,a2,b2)==a2-1){
      printf(" no  |   -    |  -  |\n");
  }
  else{
      printf(" yes |");
      printf("%lf|",newton(F2,a2,b2));
      printf("  %d  |\n",count);
  }
}
```

# Результаты

Mashinnoe eps = 2.220446e-16

vvedite otrezok var22

0 1

vvedite otrezok var23

2 4

Tablica znacheniy

| var | metod dihotomii | | | metod itteraciy | | | metod newtona | |
|---|---|---|---|---|---|---|---|---|
| __ | koren' | itter | shojd | koren' | itter | shojd | koren' | itter |
| 22 | 0.562926 | 52 | yes | 0.562926 | 17 | yes | 0.562926 | 5 |
| 23 | 3.229959 | 53 | yes | 3.229959 | 39 | yes | 3.229959 | 5 |

# Заключение

Нахождение корней трансцендентных уравнений является зачастую сложной задачей, не решаемой аналитически с помощью конечных формул. На практике уравнение содержит коэффициенты, значения которых заданы приблизительно, так что говорить о точном решении уравнений в таких случаях некорректно. Поэтому задачи приближенного определения корней уравнения и соответствующей оценки их точности имеют большое значение.

По полученным данным можно сделать вывод, что наиболее быстрым по вычислениям является метод Ньютона, так как в обоих вариантах этим методом корень был найден всего за 5 итераций.

Корни, полученные разными методами, совпали для каждого варианта и совпали с корнями, данными в условии, что говорит о правильности вычислений.

# Список информационных ресурсов


- https://prog-cpp.ru/digital-find/ (Численные методы решения нелинейных уравнений)
- http://www.machinelearning.ru/wiki/index.php?title=%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%D0%B4%D0%B8%D1%85%D0%BE%D1%82%D0%BE%D0%BC%D0%B8%D0%B8 (Метод дихотомии)
- https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%D0%B8%D1%82%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%B8 (Метод итераций)
- https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%D0%9D%D1%8C%D1%8E%D1%82%D0%BE%D0%BD%D0%B0 (Метод Ньютона)
- https://www.wolframalpha.com (Построение графиков. вычисление производных)
- https://www.google.ru (Построение графиков)
- Методические материалы к КП4 Кичинский К.А. под рук. проф. Ревизникова Д.Л. (информация о методах решения трансцендентных алгебраических уравнений)