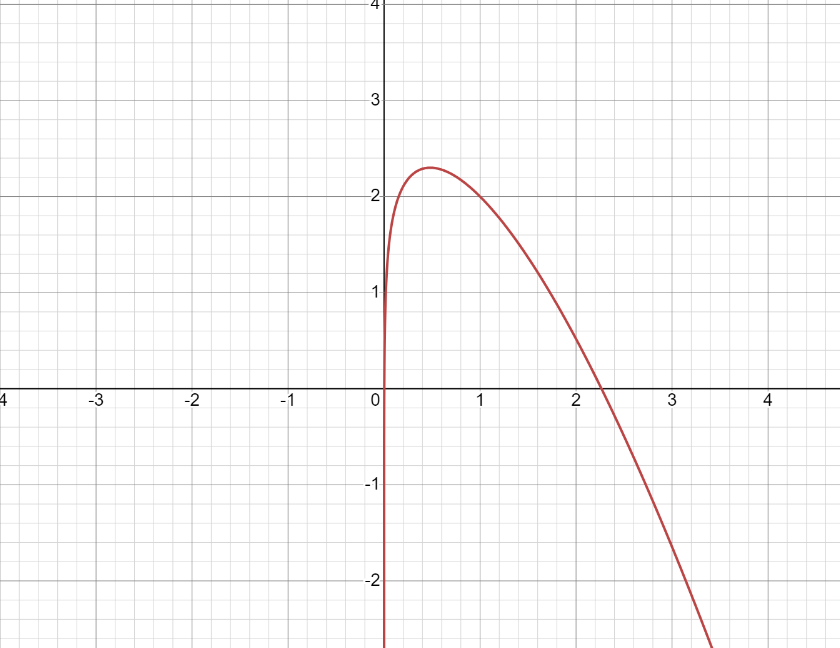
Исходное уравнение:

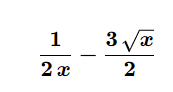
****

и его график:

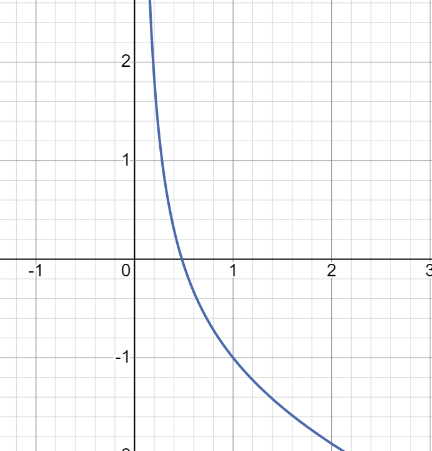


Уравнение имеет один действительный корень на отрезке единичной длины x∈[2;3 ].

Первая производная:

****

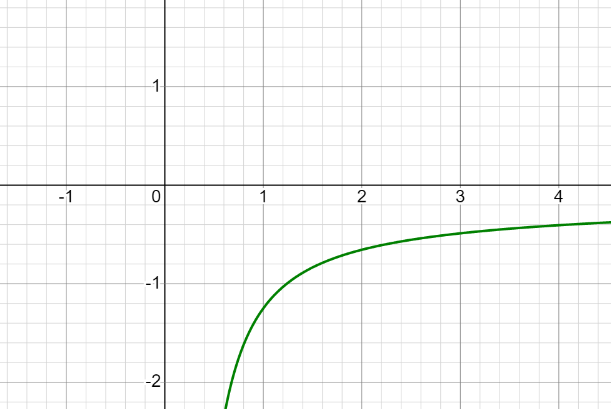
её график:



Вторая производная:



её график:



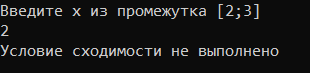
**Необходимые условия сходимости метода Ньютона:**

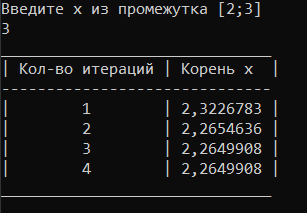
1. Функция дважды дифференцируема и непрерывна, имеет непрерывную первую производную.
2. f’(x) ≠ 0 на всем промежутке, содержащем корень.
3. f’’(x) сохраняет знак на [a,b], f’’(x) < 0 – функция выпукла вверх.
4. 4. Начальное приближение f(x) f’’(x)>0

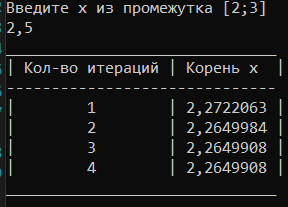
Выполняется при верхнем значении , то есть 3: f(3)=3-√33+0,5ln3=-1,646

f’’(3)= -=-0,488, а минус на минус равно плюс

**Скриншоты работы:**

****

****

****

**Код:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace Nuton

{

class Program

{

static void Main(string[] args)

{

//Console.WriteLine("3-Sqrt(x^3)+0.5ln(x) metod Nuton");

Console.WriteLine("Введите x из промежутка [2;3]");

float x = float.Parse(Console.ReadLine());

double e = 1e-6;

float x1, xt;

int k = 0;//кол-во итераций

if (F(x) \* F2(x) > 0) //проверка условия сходимости

{

Console.WriteLine("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

Console.WriteLine("| Кол-во итераций | Корень x |");

Console.WriteLine("------------------------------");

do

{

x1 = x - F(x) / F1(x);//следующий х получаемый вычитанием из нынешненего х отношения F(x) к F'(x)

xt = x;//временная переменная для хранения значения х

x = x1;

k = k + 1;//счетчик итераций

Console.WriteLine("| {0} | {1, -9} |", k, x);//выводим таблицу

} while (Math.Abs(x1 - xt) > e);//пока не станет |xn - xn-1|<=e

Console.WriteLine("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

}

else

{

Console.WriteLine("Условие сходимости не выполнено");

}

}

static float F(float x)

{

return (float)(3 - Math.Sqrt(x \* x \* x) + 0.5 \* Math.Log(x));

}

static float F1(float x)

{

return (float)(0.5/x-1.5\*Math.Sqrt(x));

}

static float F2(float x)

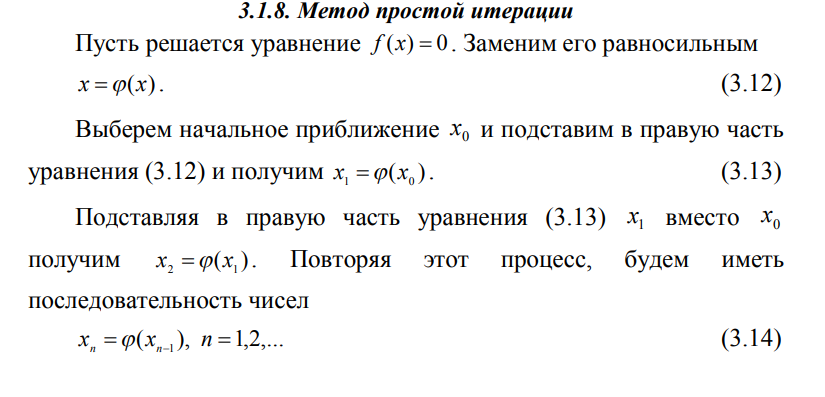
{

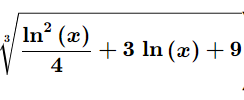
return (float)(-3/(4\*Math.Sqrt(x))-0.5/(x\*x));

}

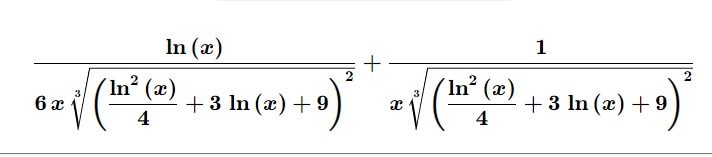
}

}

****

Пусть решается уравнение ****Заменим его равносильным x = , то есть просто выражаем х из исходного уравнения.

Первая производная будет выглядеть так:



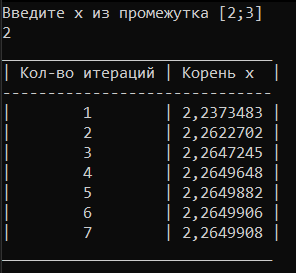
Согласно теореме о простых итерациях. Пусть f(x) определена и дифференцируема на [a;b], причем все ее значения принадлежат [a;b]. Тогда, если ∃q – правильная дробь: f’(x) ≤ 1, то при a < x < b процесс итерации сходится независимо от начального значения x0 ∈ [a;b]

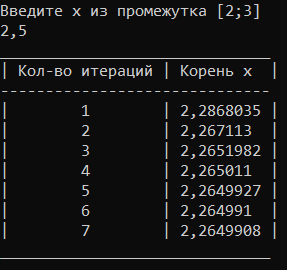
Тут условие сходимости так же выполняется на обоих концах

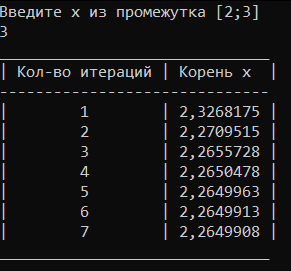
f’(2)=0,111

f(3)=0,72

**Скриншоты работы:**

****

****

****

**Код:**

namespace Iteration

{

class Program

{

static void Main(string[] args)

{

//Console.WriteLine("3-Sqrt(x^3)+0.5ln(x) metod Iteration");

Console.WriteLine("Введите x из промежутка [2;3]");

float x = float.Parse(Console.ReadLine());

double e = 1e-6;

float x1, xt;

int k = 0;//кол-во итераций

if (F1(x) <= 1)

{

Console.WriteLine("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

Console.WriteLine("| Кол-во итераций | Корень x |");

Console.WriteLine("------------------------------");

do

{

x1 = F(x);

xt = x;//временная переменная для хранения значения х

x = x1;

k = k + 1;

Console.WriteLine("| {0} | {1, -9} |", k, x);//выводим таблицу

} while (Math.Abs(x1 - xt) > e);//пока не станет |xn - xn-1|<=e

Console.WriteLine("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

}

}

static float F(float x)

{

return (float)(Math.Pow(9.0 + 3.0 \* Math.Log(x) + 0.25 \* Math.Log(x) \* Math.Log(x), 1.0 / 3));

}

static float F1(float x)

{

return (float)((Math.Log(x) / (6.0 \* x) + 1.0 / x) / Math.Pow((3.0 \* Math.Log(x) + 9.0 + Math.Log(x) \* Math.Log(x) / 4.0), 2.0/3));

}

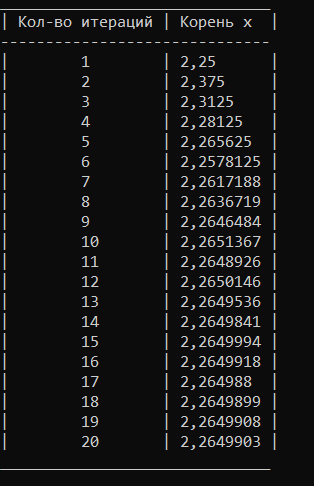
}

}

**Метод дихотомии**

Считаем, что отделение корней произведено и на интервале [a,b] расположен один корень, который необходимо уточнить с погрешностью ε.  
Итак, имеем f(a)f(b)<0. Метод дихотомии заключается в следующем. Определяем половину отрезка c=(a+b)/2 и вычисляем f(x). Проверяем следующие условия  
1. Если |f(c)| < ε, то x – корень. Здесь ε - заданная точность.  
2. Если f(c)f(a)<0, то корень лежит в интервале [a,c].  
3. Если f(c)f(b)<0, то корень лежит на отрезке[c,b].

**Скриншоты работы:**

****

**Код:**

namespace Dichotomy

{

class Program

{

static void Main(string[] args)

{

double e = 1e-6;//погрешность вычисления

float a = 2, b = 3;//интервал [2, 3]

int k = 0;//кол-во итераций

Console.WriteLine("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

Console.WriteLine("| Кол-во итераций | Корень x |");

Console.WriteLine("------------------------------");

while (b - a > e)

{

float c = (a + b) / 2;

if (f(b) \* f(c) < 0)

a = c;

else

b = c;

k++;

Console.WriteLine("| {0, -9}| {1, -9} |", k, (a + b) / 2);//выводим таблицу

}

Console.WriteLine("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

}

static float f(float x)

{

return (float)(3 - Math.Sqrt(x \* x \* x) + 0.5 \* Math.Log(x));

}

}