

# Лабораторная работа №1

## Тема «РЕШЕНИЕ НЕЛИНЕЙНЫХ УРАВНЕНИЙ С ОДНОЙ НЕИЗВЕСТНОЙ».

Рассмотрим уравнение  $f(x) = 0$ , где функция  $f(x)$  определена и непрерывна в некотором конечном и бесконечном интервале  $a < x < b$ .

**Корнем** уравнения  $f(x) = 0$  называется значение  $\xi$ , обращающее функцию  $f(x)$  в нуль, т.е. такое, что  $f(\xi) = 0$ .

Уравнение  $f(x) = 0$  называется **алгебраическим**, если функция  $f(x)$  является многочленом  $f(x) = P_n(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0$ , в противном случае уравнение  $f(x) = 0$  называется **трансцендентным**.

Встречающиеся на практике уравнения часто не удается решить аналитическими методами. Для решения таких уравнений используются численные методы.

Алгоритм нахождения корня уравнений с помощью численного метода состоит из двух этапов:

1. **отделение** и **локализация** корня, т.е. установление промежутка  $[a, b]$ , в котором содержится один корень;
2. **уточнение** значения корня, т.е. находят корни с заданной точностью.

Для отделения корней удобно пользоваться двумя следующими теоремами (шаговым методом).

**Теорема 1:** Если функция  $f(x)$  непрерывна на отрезке  $[a, b]$ , причем  $f(a) * f(b) < 0$ , то на этом отрезке существует хотя бы один корень уравнения  $f(x) = 0$ .

**Теорема 2:** Если непрерывная функция  $f(x)$  **монотонна** (если функция  $f(x)$  дифференцируема и ее производная сохраняет знак на отрезке  $c$ , то  $f(x)$  монотонна на этом отрезке) на отрезке  $[a, b]$ , причем  $f(a) * f(b) < 0$ , то на этом отрезке существует единственный корень уравнения  $f(x) = 0$ .

### Шаговый метод

Дано уравнение  $f(x) = 0$ . Задан интервал поиска  $[x_0, x_1]$ . Требуется найти интервал  $[a, b]$  длиной  $h$ , содержащий первый корень уравнения, начиная с левой границы интервала поиска.

*Алгоритм метода:*

1. Установить интервал  $[a, b]$  на начало интервала поиска ( $a = x_0$ ).
2. Определить координату точки  $b$  ( $b = a + h$ ), а также значения функции в точках  $a$  и  $b$ :  $F(a)$  и  $F(b)$ .

3. Проверить условие  $F(a) \cdot F(b) < 0$ . Если условие не выполнено - передвинуть интервал  $[a, b]$  на один шаг ( $a=b$ ) и перейти к пункту 2. Если условие выполнено - закончить алгоритм.

**Пример 1:** Рассмотрим нелинейное уравнение  $y = 2^x + 2x - 5$  отделим и локализуем корни, используя **шаговый метод** на отрезке  $[1.25; 1.3]$  с шагом 0.01.

*Решение*

**Ручной счет:**

$a$	$b$	$f(a)$	$f(b)$	$f(a) \cdot f(b) < 0$
1.25	1.26	-0.12	-0.085	нет
1.26	1.27	-0.085	-0.048	нет
1.27	1.28	-0.048	-0.012	нет
1.28	1.29	-0.012	0.025	да
1.29	1.3	0.025	0.062	нет

Таким образом, на отрезке  $[1.28; 1.29]$  существует единственный корень уравнения  $y = 2^x + 2x - 5$  рассмотренного на интервале  $[1.25; 1.3]$ .

**Реализация в Microsoft Excel:**

Шаговый метод	
Начальное значение	1,25
Шаг табуляции	0,01
x	f(x)
1,25	-0,122
1,26	-0,085
1,27	-0,048
<b>1,28</b>	-0,012
<b>1,29</b>	0,025
1,3	0,062

### ***Реализация в Mathcad:***

#### 1. Шаговый метод

$$x := 1.25, 1.26 \dots 1.3$$

$$f(x) := 2^x + 2 \cdot x - 5$$

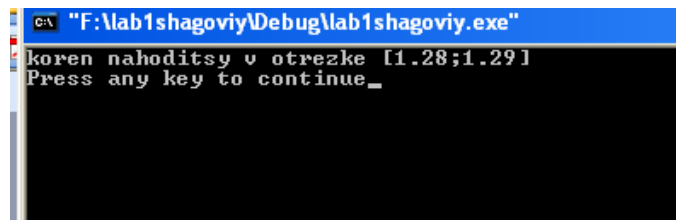
x =	f(x) =
1.25	-0.122
1.26	-0.085
1.27	-0.048
1.28	-0.012
1.29	0.025
1.3	0.062

### ***Реализация в Microsoft Visual C++:***

```
#include "stdafx.h"
#include "stdio.h"
#include "iostream.h"
#include "math.h"

double f(double x)
{
    return pow(2,x)+2*x-5;
}

int main(int argc, char* argv[])
{
    double a,b,h;
    a=1.25;
    b=1.3;
    h=0.01;
    do
    {
        b=a+h;
        if(f(a)*f(b)<0)
        {
            cout<<"koren nahoditsy v otrezke ["<<a<<" "<<b<<"]"<<endl;
        }
        a=b;
    }
    while(b<=1.3);
    return 0;
}
```



После того как найден интервал, содержащий корень, применяют итерационные методы уточнения корня с заданной точностью. Мы разберем следующие методы:

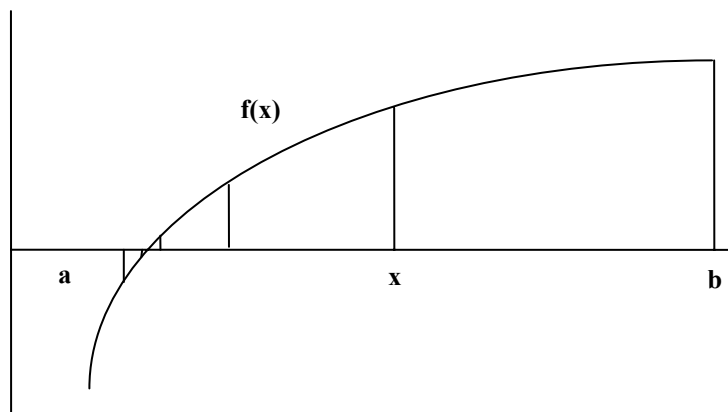
1. метод половинного деления
2. метод Ньютона (метод касательных)
3. метод итерации

### ***Метод половинного деления***

Метод основан на последовательном сужении интервала, содержащего единственный корень уравнения  $f(x) = 0$  до тех пор, пока не будет достигнута заданная точность  $\varepsilon$ . Пусть задан отрезок  $[a, b]$ , содержащий один корень уравнения. Этот отрезок может быть предварительно найден с помощью шагового метода.

*Алгоритм метода* (рис. 1):

1. Определить новое приближение корня  $x$  в середине отрезка  $[a, b]$ :  $x = (a+b)/2$ .
2. Найти значения функции в точках  $a$  и  $x$ :  $f(a)$  и  $f(x)$ .
3. Проверить условие  $f(a) \cdot f(x) < 0$ . Если условие выполнено, то корень расположен на отрезке  $[a, x]$ . В этом случае необходимо точку  $b$  переместить в точку  $x$  ( $b = x$ ). Если условие не выполнено, то корень расположен на отрезке  $[x, b]$ . В этом случае необходимо точку  $a$  переместить в точку  $x$  ( $a = x$ ).
4. Перейти к пункту 1 и вновь поделить отрезок пополам. Алгоритм продолжить до тех пор, пока не будет выполнено условие  $|f(x)| < \varepsilon$ .



**Рис. 1. Иллюстрация метода половинного деления**

**Пример 2:** Рассмотрим нелинейное уравнение  $y = 2^x + 2x - 5$  известно, что единственный корень находится на отрезке  $[1.28; 1.29]$ . Требуется уточнить значение корня методом половинного деления с точностью  $\varepsilon = 0,001$ .

*Решение*

**Ручной счет**

Построим таблицу в соответствии с алгоритмом метода.

a	x	b	f(a)	f(x)	f(a)*f(x)<0
1,28	1,285	1,29	-0,01161	0,00682	да
1,28	1,2825	1,285	-0,01161	-0,0024	нет
1,285	1,28375	1,285	0,002399	0,00221	да
1,2825	1,283125	1,28375	0,002399	-0,0001	нет
1,283125	1,283438	1,28375	-0,0001	0,00106	стоп

Алгоритм остановлен, поскольку  $|-0,00106| < 0,001$ .

*Ответ:* уточненное значение корня  $x \approx 1,283438$ .

**Реализация в Microsoft Excel:**

Метод половинного деления					
Начальное значение	1,28				
Шаг табуляции	нет				
Точность	0,001				
a	x	b	f(a)	f(x)	f(a)*f(x)<0

1,2800	1,2850	1,2900	-0,0116	0,0068	да
1,2800	1,2825	1,2850	-0,0116	0,0024	нет
1,2825	1,2838	1,2850	-0,0024	0,0022	да
1,2825	1,2831	1,2838	-0,0024	0,0001	нет
1,2831	<b>1,2834</b>	1,2838	-0,0001	0,0011	стоп

### Реализация в Mathcad:

Метод половинного деления

$$a := 1.28 \quad b := 1.29$$

$$f(x) := 2^x + 2 \cdot x - 5$$

$$f(a) f(b) = -2.935 \times 10^{-4}$$

$$xc(a, b) := \frac{a + b}{2}$$

$$int(a, b) := \text{if} \left[ f(a) f(xc(a, b)) < 0, \begin{pmatrix} a \\ xc(a, b) \end{pmatrix}, \begin{pmatrix} xc(a, b) \\ b \end{pmatrix} \right]$$

$$int(a, b) = \begin{pmatrix} 1.28 \\ 1.285 \end{pmatrix}$$

$a_0 := 1.28$   $b_0 := 1.29$   
 $i := 0..5$

$$\begin{pmatrix} a_{i+1} \\ b_{i+1} \end{pmatrix} := int(a_i, b_i)$$

i =	a <sub>i</sub> =	b <sub>i</sub> =	f(a <sub>i</sub> ) =	f(b <sub>i</sub> ) =
0	1.28	1.29	-0.012	0.025
1	1.28	1.285	-0.012	6.821·10 <sup>-3</sup>
2	1.283	1.285	-2.399·10 <sup>-3</sup>	6.821·10 <sup>-3</sup>
3	1.283	1.284	-2.399·10 <sup>-3</sup>	2.21·10 <sup>-3</sup>
4	1.283	1.284	-9.443·10 <sup>-5</sup>	2.21·10 <sup>-3</sup>
5	1.283	1.283	-9.443·10 <sup>-5</sup>	1.058·10 <sup>-3</sup>

### Реализация в Microsoft Visual C++:

```
#include "stdafx.h"
#include "stdio.h"
#include "iostream.h"
#include "math.h"
#define eps 0.001
```

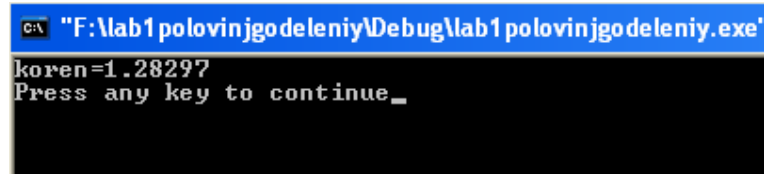
```
double f(double x)
{
    return pow(2,x)+2*x-5;
}
```

```
int main(int argc, char* argv[])
{
    double a,b,c;
    a=1.28;
    b=1.29;
    while(fabs(f(c))>eps)
    {
        if(f(a)*f(c)<0)
```

```

    {
        b=c;
    }
    else
    {
        a=c;
    }
    c=(a+b)/2;
}
cout<<"koren="<<c<<endl;
return 0;
}

```



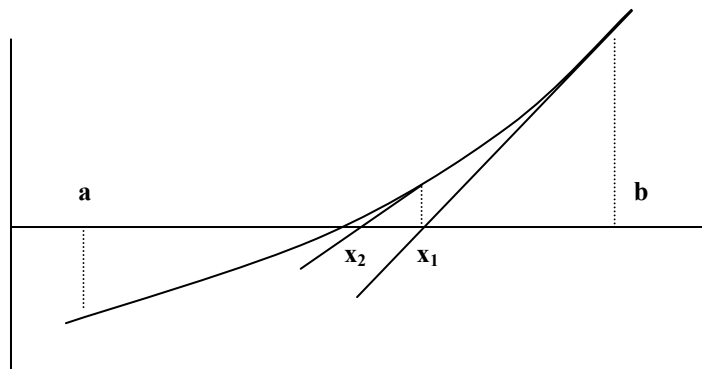
Достоинство метода: более быстрая сходимость к заданной точности, чем у шагового. Недостаток: если на отрезке  $[a,b]$  содержится более одного корня, то метод не работает.

### ***Метод Ньютона (метод касательных)***

Задан отрезок  $[a,b]$ , содержащий корень  $f(x) = 0$ . Уточнение значения корня производится путем использования уравнения касательной. В качестве начального приближения задается тот из концов отрезка  $[a,b]$ , где значение функции и ее второй производной имеют одинаковые знаки (т.е. выполняется условие  $f(x_0) \cdot f''(x_0) > 0$ ). В точке  $f(x_0)$  строится касательная к кривой  $y = F(x)$  и ищется ее пересечение с осью  $x$ . Точка пересечения принимается за новую итерацию. Итерационная формула имеет вид:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

Итерационный процесс продолжается до тех пор, пока не будет выполнено условие  $|f(x)| < \varepsilon$ , где  $\varepsilon$  - заданная точность.



**Рис. 2. Иллюстрация метода Ньютона**

Рис. 2. иллюстрирует работу метода Ньютона. В данном случае вторая производная функции положительна, поэтому в качестве начального приближения выбрана точка  $x_0 = b$ . Как видно из рисунка, метод имеет очень быструю сходимость: обычно заданная точность достигается за 2-3 итерации.

**Пример 3:** Дано нелинейное уравнение  $y = 2^x + 2x - 5$ . Известно, что корень находится на отрезке  $[1.28; 1.29]$ . Требуется уточнить значение корня методом Ньютона с точностью  $\varepsilon = 0,001$ .

*Решение*

**Ручной счет**

Найдем первую и вторую производную функции  $f(x)$ .  $f'(x) = 2^x \ln(2) + 2$ ;  $f''(x) = 2^x \ln^2(2)$ .  $f(1,28) = -0,116$ ;  $f(1,29) = 0,025$ . Следовательно, в качестве начального приближения выбираем точку  $x_0 = b = 1,29$ . Построим таблицу в соответствии с алгоритмом метода.

i	$x_i$	$f(x_i)$	$f'(x_i)$	$f(x_i) < 0,001$
0	1,29	0,025	3,695	нет
1	1,2832	0,0002	3,6869	да

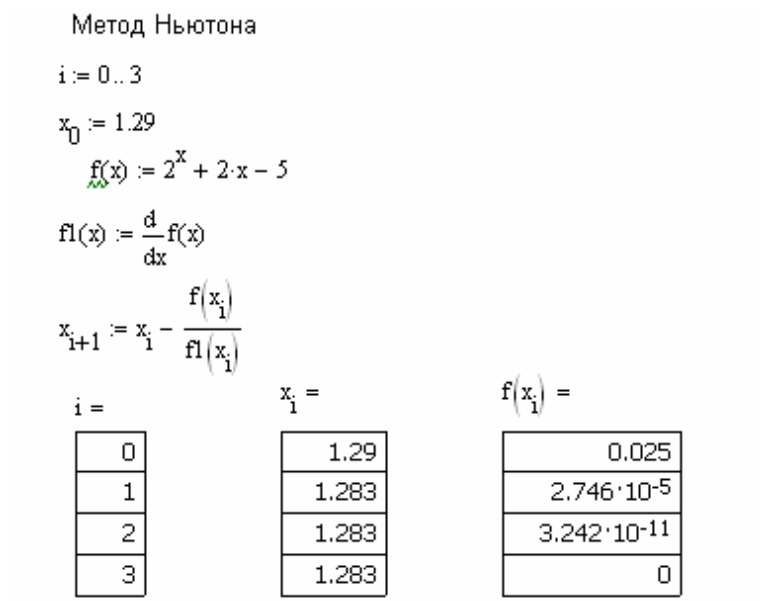
*Ответ:* уточненное значение корня  $x \approx 1,2832$ .

**Реализация в Microsoft Excel:**

Метод Ньютона	
Начальное значение	1,29
Шаг табуляции	нет
Точность	0,001
$x_{i+1} = x_i - \frac{2^{x_i} + 2 * x_i - 5}{2^{x_i} \ln(2) + 2}$	
$x_{i+1}$	$f(x_{i+1})$
<b>1,2832</b>	<b>0,000003</b>
1,2832	0,000009
1,2832	0,000003



### Реализация в Mathcad:



### Реализация в Microsoft Visual C++:

```
#include "stdafx.h"
#include "stdio.h"
#include "iostream.h"
#include "math.h"
#define eps 0.001

double f(double x)
{
    return pow(2,x)+2*x-5;
}

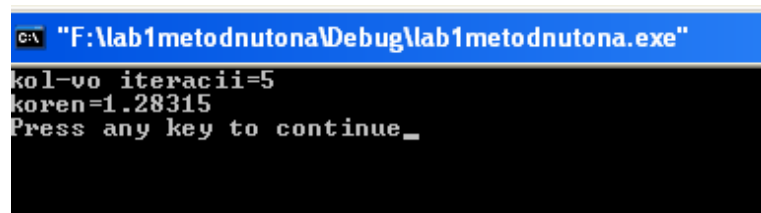
double f1(double x)
{
    return pow(2,x)*log(2)+2;
}

double f2(double x)
{
    return pow(2,x)*pow(log(2),2);
}
```

```

int main(int argc, char* argv[])
{
    double a=1.28, b=1.29,x,x1;
    int k=0;
    x=a;
    if(f(a)*f2(a)>0)
    {
        x=a;
    }
    else
    {
        x=b;
    }
    while(fabs(x1-x)>eps)
    {
        x=x1;
        x1=x-f(x)/f1(x);
        k++;
    }
    cout<<"kol-vo iteracii="<<k<<endl;
    cout<<"koren="<<x1<<endl;
    return 0;
}

```



Достоинство метода: очень быстрая сходимость к заданной точности. Недостаток: громоздкий алгоритм: на каждой итерации необходимо вычислять значение функции и ее первой производной.

### ***Метод простой итерации***

Метод основан на замене исходного уравнения  $f(x)=0$  на эквивалентное  $x=\varphi(x)$ . Функция  $\varphi(x)$  выбирается таким образом, чтобы на обоих концах отрезка  $[a,b]$  выполнялось условие сходимости  $|\varphi'(x)| < 1$ . В этом случае в качестве начального приближения можно выбрать любой из концов отрезка. Итерационная формула имеет вид

$$x_{i+1} = \varphi(x_i)$$

Итерационный процесс продолжается до тех пор, пока не будет выполнено условие  $|f(x)| < \varepsilon$ , где  $\varepsilon$  - заданная точность.

**Пример 4:** Дано нелинейное уравнение  $y = 2^x + 2x - 5$ . Известно, что корень находится на отрезке  $[1.28; 1.29]$ . Требуется уточнить значение корня методом простой итерации с точностью  $\varepsilon = 0,001$ .

*Решение*

**Ручной счет**

На первом этапе нам необходимо выбрать функцию  $\varphi(x)$ , удовлетворяющую условию сходимости.

Запишем исходное уравнение в виде  $x = \frac{5 - 2^x}{2}$ . Тогда  $\varphi(x) = \frac{5 - 2^x}{2}$ ;  
 $\varphi'(x) = -\frac{2^x \ln(2)}{2}$ ;  $\varphi(1,28) = -0,842$ ;  $\varphi(1,29) = -0,847$ . Условие сходимости выполнено, поскольку  $|-0,842| < 1$  и  $|-0,847| < 1$ .

Следовательно, итерационная формула имеет вид:

$$x_{i+1} = \frac{5 - 2^{x_i}}{2}.$$

В качестве начального приближения можно выбрать любой из концов отрезка, например  $x_0 = a = 1,28$ . Построим таблицу в соответствии с алгоритмом метода.

i	$x_i$	$f(x_i)$	$ f(x_i)  < 0,001$
0	1,28	-0,0116	нет
1	1,2858	0,009791	нет
2	1,2809	-0,00826	нет
3	1,285039	0,006966	нет
4	1,28156	-0,0059	нет
5	1,28449	0,00496	нет
6	1,282	-0,0042	нет
7	1,284	0,0035	нет
8	1,282	-0,00297	нет
9	1,28383	0,0025	нет
10	1,2826	-0,0021	нет
11	1,2836	0,00178	нет
12	1,2827	-0,0015	нет
13	1,2835	0,0013	нет
14	1,28286	-0,0012	нет
15	1,2834	0,0009	да

**Ответ:** уточненное значение корня  $x \approx 1,2834$ .

**Реализация в Microsoft Excel:**

Метод простой итерации	
Начальное значение	1,28
Шаг табуляции	нет
Точность	0,001
Эквивалентная формула $x_{i+1} = \frac{5 - 2^{x_i}}{2}$	
Функция $\varphi(x_i)$ $f(x)$	
1,2858	0,0098
1,2809	-0,0083
1,2850	0,0070
1,2816	-0,0059
1,2845	0,0050
1,2820	-0,0042
1,2841	0,0035
1,2823	-0,0030
1,2838	0,0025
1,2826	-0,0021
1,2836	0,0018
1,2827	-0,0015
1,2835	0,0013
1,2829	-0,0011
<b>1,2834</b>	<b>0,0009</b>

**Реализация в Mathcad:**

---

Метод простой итерации

$$f(x) := 2^x + 2 \cdot x - 5$$

$$\varphi(x) := \frac{5 - 2^x}{2}$$

$$a := 1.28$$

$$b := 1.29$$

$$i := 0..15$$

$$\varphi_1(x) := \frac{d}{dx} \varphi(x)$$

Условия сходимости:

$$\varphi_1(a) = -0.842$$

$$\varphi_1(b) = -0.847$$

выполнены, т.е. выбранная  $\varphi(x)$  нам подходит.

$$x_{i+1} := \varphi(x_i)$$

$i =$	$x_1 =$	$f(x_1) =$
0	1.29	0.025
1	1.277	-0.021
2	1.288	0.018
3	1.279	-0.015
4	1.287	0.013
5	1.28	-0.011
6	1.286	$9.104 \cdot 10^{-3}$
7	1.281	$-7.68 \cdot 10^{-3}$
8	1.285	$6.477 \cdot 10^{-3}$
9	1.282	$-5.463 \cdot 10^{-3}$
10	1.284	$4.608 \cdot 10^{-3}$
11	1.282	$-3.887 \cdot 10^{-3}$
12	1.284	$3.278 \cdot 10^{-3}$
13		

**Реализация в Microsoft Visual C++:**

```
#include "stdafx.h"
#include "iostream.h"
#include "math.h"
#define eps 0.001

double f(double x)
{
    return pow(2,x)+2*x-5;
}

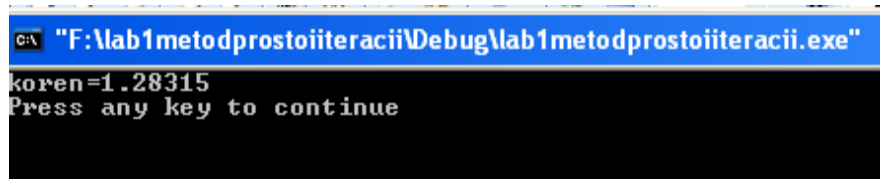
double fl(double x)
{
    return pow(2,x)*log(2)+2;
}

int main(int argc, char* argv[])
{
    double a=1.28,b=1.29,x,x1,max;
    x1=b;
    if(fl(a)>fl(b))
    {
        max=f1(a);
    }
}
```

```

else
{
    max=f1(b);
}
while(fabs(x1-x)>eps)
{
    x=x1;
    x1=x-f(x)/max;
}
cout<<"koren="<<x1<<endl;
return 0;
}

```



```

C:\> "F:\lab1metodprostoiiteracii\Debug\lab1metodprostoiiteracii.exe"
koren=1.28315
Press any key to continue

```

Достоинство метода: простота алгоритма. Недостатки: возможные сложности с выбором функции  $\varphi(x)$ ; более медленное достижение заданной точности, чем у других методов уточнения.