|  |  |  |
| --- | --- | --- |
| **Лабораторная работа № 1** **По вычисленной математике**  «Решение нелинейных интрансцендентных алгебраических уравнений» | Студент | Муха А. В. |
| Группа | ИВТ-262 |
| Дата выполнения |  |
| Оценка |  |
| Преподаватель | Скворцов М.И. |

1. **Постановка задачи**

Уточнение корней нелинейных алгебраических уравнений

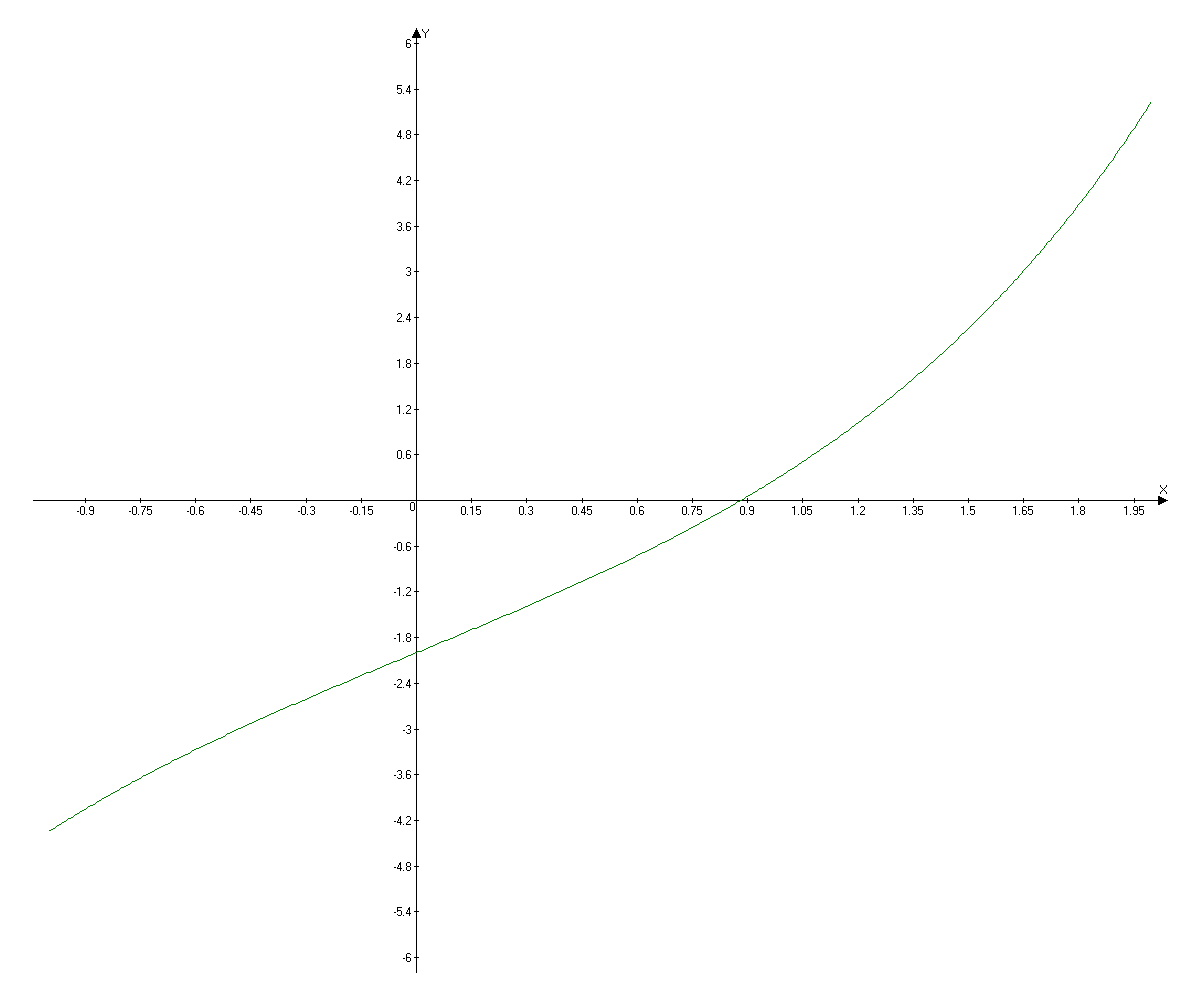
На интервале [1,3]

Используемые методы

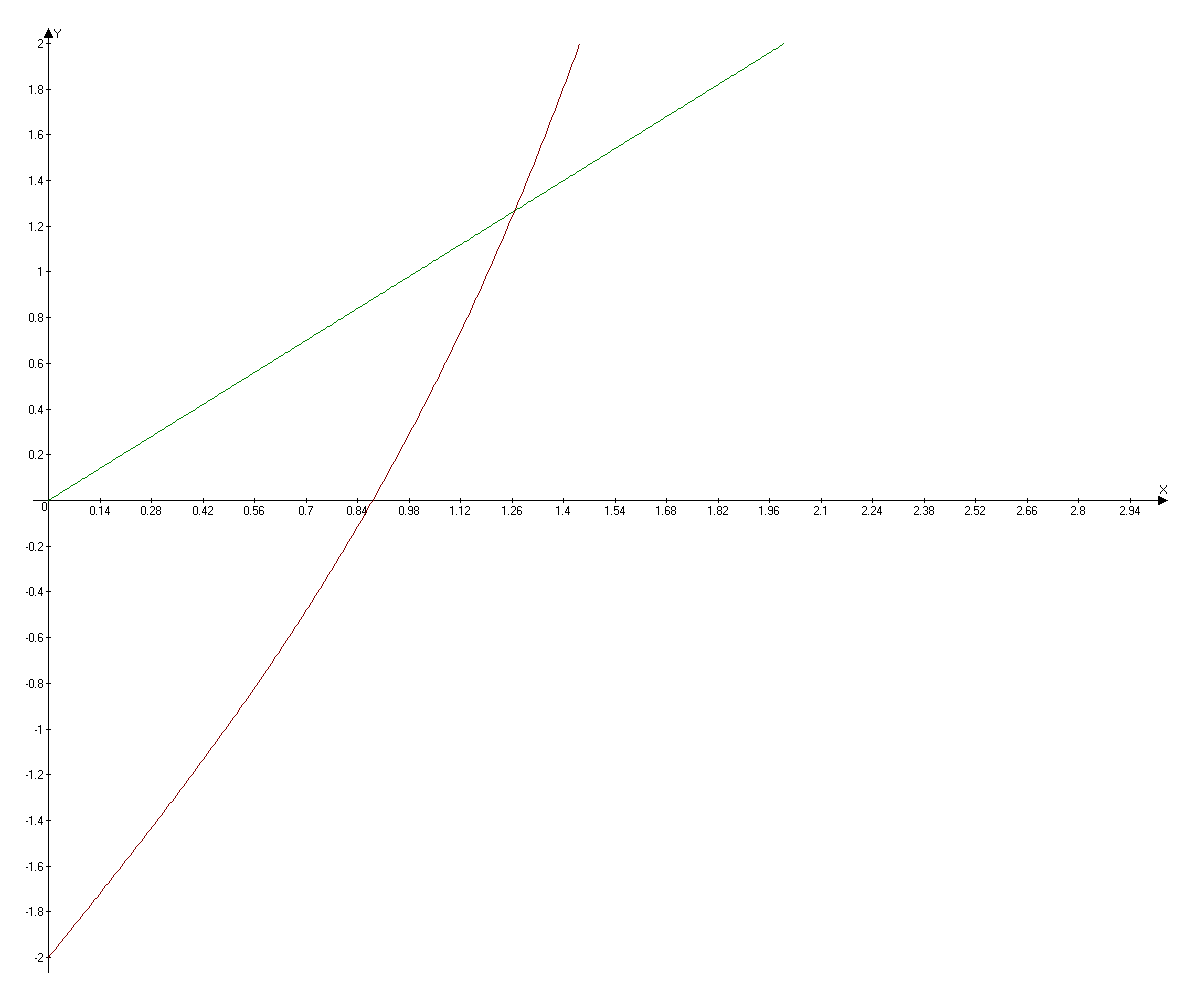
1. Метод дихотомии
2. Метод простых итераций
3. Метод Ньютона

**2. Графики**

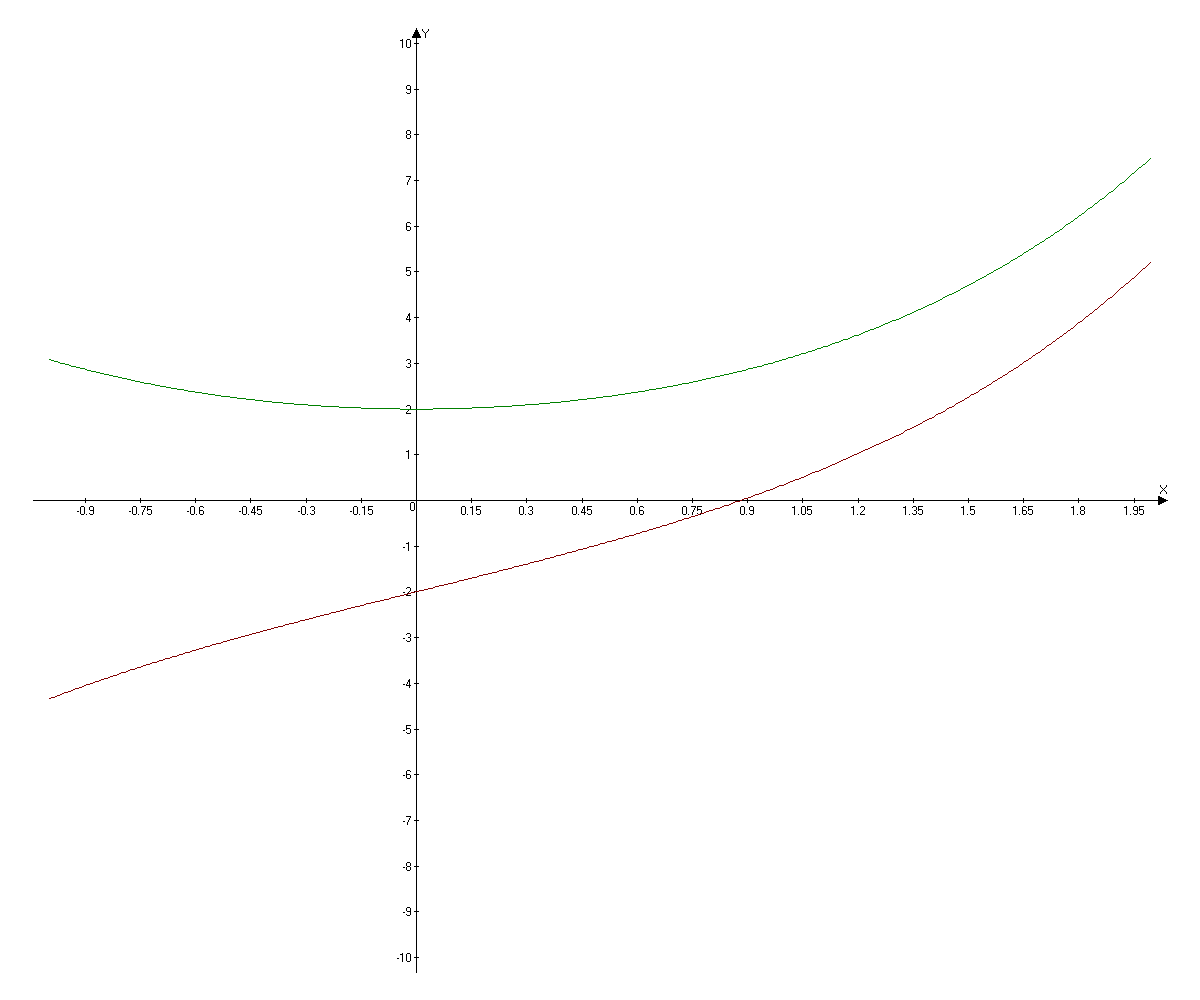
1. Метод дихотомии



**2.Метод итерации**



**3.Метод Ньютона**

****

**3.Блок-схема**

1: Дихотомии

2: Итерации

3: Ньютон

choice

choice ≠ 4

1

2

3

да

нет

choice

Дихотомии

1

2

3

Итерация

Ньютон

4

Ввод: a,b,s

N = 0

X = (a+b)/2

F(a).F(x) < 0

N=N+1

b = x

a=x

F(a) = F(x)

|b-a|>s

Вывод: x,N

да

нет

да

нет

**Метод интерации**

Ввод: a,b,s

N = 0

Func\_s =

Arg\_p = func\_s

Func\_s =

N=N+1

|arg\_p-func\_s|>e

Вывод: X,N

да

нет

**Метод Ньютона**

Ввод: a,b,epsilon

N = 0

Xs =

Arg\_p = func\_s

Func\_s = agr\_p-F(arg\_p)/F’(arg\_p)

N=N+1

|arg\_p-func\_s|>s

Вывод: func\_s,N

да

нет

**4.Таблица**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Методы | | | | | | | | |
| Дихотомии | | | Интерации | | | Ньютона | | |
| Результат  Корни  **x** | Число  Циклов  **N** | Проверка  **F(x)** | Результат  Корни  **x** | Число  циклов  **N** | Проверка  **F(x)** | Результат  Корни  **x** | Число  циклов  **N** | Проверка  **F(x)** |
| 10-4 |  |  |  |  |  |  |  |  |  |
| 10-5 |  |  |  |  |  |  |  |  |  |
| 10-6 |  |  |  |  |  |  |  |  |  |

**5.Вывод**

Метод дитохомии − простой и надежный метод поиска простого корня любой функции, устойчивый к погрешности округления. Даже если на отрезке есть несколько корней (нечетное количество),то будет найден один из них.

Недостатки метода: скорость сходимости низкая, не обобщается на систему уравнений.

Метод дихотомии нельзя использовать для уточнения не простого корня − корень совпадает с точкой экстремума функции, т.к. в этом случае функция не изменяет свой знак в окрестности корня.

Метод простых итераций достаточно прост, обобщается на системы уравнений, устойчив к погрешности округления (она не накапливается).

Достоинства метода: высокая скорость сходимости; обобщается на системы уравнений.

Недостатки: сложный, т.к. требуется вычисление производных; сильная зависимость сходимости от вида функции и выбора начального приближения.

**КОД ПРОГРАММЫ**

#include "stdafx.h"

#include <math.h>

#include <windows.h>

#include <iostream>

using namespace std;

void Dihotomy(float lim1, float lim2, float epsil);

void Iteration(float lim1, float lim2, float epsil);

void NewtonBinom(float lim1, float lim2, float epsil);

void Output(float argum, float func, int N);

int \_tmain(int argc, \_TCHAR\* argv[])

{

int choice; // переменная выбора

char buf[51]; // строка

float lim1, // нижний предел

lim2, // верхний предел

epsil; // погрешность

do

{

system("cls");

CharToOem("Программа нахождения корней нелинейных уравнений", buf);

cout << buf << endl << endl << endl;

CharToOem("1. Метод дихотомии", buf);

cout << buf << endl;

CharToOem("2. Метод простых итераций", buf);

cout << buf << endl;

CharToOem("3. Бином Ньютона", buf);

cout << buf << endl << endl;

CharToOem("Введите цифру, соответствующую методу: ", buf);

cout << buf;

cin >> choice;

cout << endl << endl;

if ((choice != 1) && (choice != 2) && (choice != 3))

{

CharToOem("Программа завершена", buf);

cout << buf << endl << endl << endl;

// Задержка

system("pause");

return 0;

}

////////////////////////////////////////////////

// Ввод данных

CharToOem("Введите интервал [0, 1]\na = ", buf);

cout << buf;

cin >> lim1;

CharToOem("b = ", buf);

cout << buf;

cin >> lim2;

if ((lim1 < 0) || (lim1 >= 1) || (lim2 > 1) || (lim2 <= 0))

{

CharToOem("\nОшибка в указании интервала!!\a", buf);

cout << buf << endl << endl << endl;

}

else

{

CharToOem("\nВведите погрешность вычислений\nepsil = ", buf);

cout << buf;

cin >> epsil;

if (choice == 1)

{

Dihotomy(lim1, lim2, epsil);

}

else if (choice == 2)

{

Iteration(lim1, lim2, epsil);

}

else if (choice == 3)

{

NewtonBinom(lim1, lim2, epsil);

}

else

{

CharToOem("Программа завершена", buf);

cout << buf << endl << endl << endl;

}

}

// Задержка

system("pause");

}

while ((choice == 1) || (choice == 2) || (choice == 3));

return 0;

}

void Dihotomy(float lim1, float lim2, float epsil)

{

float argum; // аргумент функции

float func;

int N; // число ???

float func1; // функция от a

float f\_arg; // функция от x

// Функция от a

func1 = exp(lim1) - exp(-lim1) - 2;

N = 0;

do

{

argum = (lim1 + lim2) / 2;

f\_arg = exp(argum) - exp(-argum) - 2;

if (func1 \* f\_arg < 0)

{

lim2 = argum;

}

else

{

lim1 = argum;

func1 = f\_arg;

}

N++;

}

while (fabs(lim2 - lim1) > epsil);

func = exp(argum) - exp(argum) - 2;

Output(argum, func, N);

}

void Iteration(float lim1, float lim2, float epsil)

{

float func\_s, // аргумент функции

func;

int N; // число повторений цикла

float arg\_p; // p

N = 0;

func\_s = exp(lim1) - exp(-lim2) - 2;

do

{

arg\_p = func\_s;

func\_s = exp(arg\_p) - exp(-arg\_p) - 2;

N++;

}

while (fabs(arg\_p - func\_s) > epsil);

func = exp(func\_s) - exp(func\_s) - 2;

// Вывод результатов

Output(func\_s, func, N);

}

void NewtonBinom(float lim1, float lim2, float epsil)

{

float func\_s, // аргумент функции

func;

int N; // число повторений цикла

float arg\_p; // p

N = 0;

func\_s = exp(lim1) - exp(-lim1) - 2;

do

{

arg\_p = func\_s;

func\_s = arg\_p - (exp(arg\_p) - exp(-arg\_p) - 2) / (exp(arg\_p) + exp(-arg\_p));

N++;

}

while (fabs(func\_s - arg\_p) > epsil);

func = exp(func\_s) - exp(func\_s) - 2;

Output(func\_s, func, N);

}

void Output(float argum, float func, int N)

{

char buf[51];

// Вывод результатов

CharToOem("\nАргумент функции = ", buf);

cout << buf << argum;

CharToOem("\tФункция = ", buf);

cout << buf << func;

CharToOem("\tN = ", buf);

cout << buf << N << endl << endl << endl;

}