**Министерство науки и высшего образования Российской Федерации**

федеральное государственное автономное образовательное учреждение

высшего образования

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ**

**ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Инженерная школа природных ресурсов

Направление подготовки Химическая технология

Отделение химической инженерии

**Итерационные методы решения нелинейных уравнений**

**Лабораторная работа по дисциплине «Углубленный курс информатики»**

Выполнил студент гр. 2Д91 А.А. Бахтин

(Подпись)

\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г.

Отчет принят:

Преподаватель

доцент ОХИ ИШПР, к.т.н. В.А. Чузлов

(Подпись)

\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г.

Томск 2020 г.

**Цель работы:** изучить численные методы решения и научиться составлять программы для решения нелинейных уравнений методом половинного деления, простых итераций и методом Ньютона.

**Теоретическая часть**

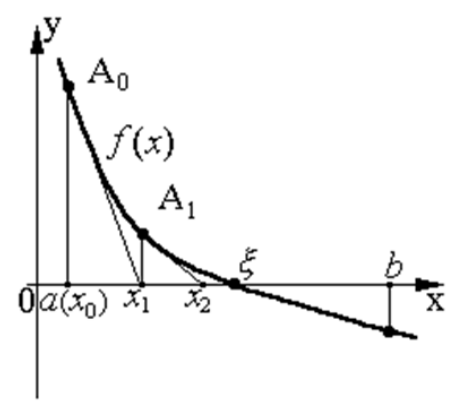
Любое уравнение можно представить в виде ƒ(x)=0, перенеся всё в одну сторону, тогда поиск корней уравнения сводится к поиску точек пересечения функции ƒ(x) с осью абсцисс.

Метод половинного деления основан на поиске отрезка, содержащего корень и последующим уменьшением его размеров до достижения заданной точности вычислений. Уменьшение размеров отрезка осуществляется циклическим делением его пополам и отбрасыванием половинки, не содержащей корня.

1. ***f(a) \* f(x1) > 0***→ корень находится на отрезке ***[x1, b]***,исключаем отрезок ***[a, x1];***
2. ***f(a) \* f(x1) < 0*** → корень находится на отрезке ***[a, x1]***,исключаем отрезок ***[x1, b].***

Метод простых итераций (метод последовательных приближений) - итерационное приближение этой точки к истинному значению корня до достижения заданной точности. Каждая последующая точка вычисляется, зная предыдущую точку и значение производной функции в этой точке.

Метод Ньютона (метод касательных) является эффективным методом уточнения корней, если известно начальное приближение к корню уравнения f(x)=0. В данном методе процесс итераций состоит в том, что в качестве приближений к корню принимаются значения x0, x1, x2... точек пересечения касательной к кривой y=f(x) с осью абсцисс.



1. В качестве начального приближения выберем х0=a, для которого выполняется условие 
2. Проведем касательную в точке A0[x0, f(x0)].
3. Первым приближением корня будет точка пересечения этой касательной с осью абсцисс х1.
4. Через точку A1[x1, f(x1)] снова проводим касательную, точка пересечения которой с осью ОХ даст нам второе приближение корня х2 и т.д.

**Практическая часть**

**Задание:** составить программу для решения нелинейных уравнений методом половинного деления, простых итераций и методом Ньютона:

1. Интервал [1; 2], допустимая точность 10-2;

**Программная реализация:**

(метод половинного деления)

**program** lb1;

**const**

eps = 1e-2;

**function** f(x: real): real;

**begin**

result := sqr(x) \* sqr(x) + 3 \* x - 20

**end**;

**function** dihotomy(a, b: real; eps: real): real;

**var**

x: real;

**begin**

**repeat**

x := (a + b) / 2;

**if** f(a) \* f(x) > 0 **then**

a := x

**else**

b := x

**until** (abs(a - b) <= eps) **or** (f(x) = 0);

result := x

**end**;

**begin**

writeln('x= ', dihotomy(1, 2, eps):2:3)

**end**.

**Ответ:**

x= 1.945

**Программная реализация:**

(метод простых итераций)

**program** lb2;

**const**

eps = 1e-2;

**function** f(x: real): real;

**begin**

result := exp(ln(20 - 3 \* x) / 4)

**end**;

**function** iterations(a, b: real; eps: real): real;

**var**

x: real;

**begin**

result := a;

**repeat**

x := f(result);

result := f(x)

**until** abs(result - x) <= eps;

**end**;

**begin**

writeln('x= ', iterations(1, 2, eps):2:2)

**end**.

**Ответ:**

x= 1.94

**Программная реализация:**

(метод Ньютона)

**program** lb3;

**const**

eps = 1e-2;

**function** f(x: real): real;

**begin**

result := sqr(x) \* sqr(x) + 3 \* x - 20

**end**;

**function** f1(x: real): real;

**begin**

result := 4 \* sqr(x) \* x + 3

**end**;

**function** f2(x: real): real;

**begin**

result := 12 \* sqr(x)

**end**;

**function** newton(a, b: real; eps: real): real;

**var**

x: real;

**begin**

**if** f(a) \* f2(a) > 0 **then**

result := a

**else**

**if** f(b) \* f2(b) > 0 **then**

result := b

**else**

**begin**

writeln('Метод Ньютона, решений нет!');

**exit**

**end**;

**repeat**

x := result;

result := x - f(x) / f1(x)

**until** abs(result - x) <= eps;

**end**;

**begin**

writeln('x= ', newton(1, 2, eps):2:2)

**end**.

**Ответ:**

x= 1.94



Интервал [0; 1], допустимая точность 10-3;

**Программная реализация:**

**(**метод половинного деления)

**program** lb1;

**const**

eps = 1e-3;

**function** f(x: real): real;

**begin**

result := exp(x) + x - 2

**end**;

**function** dihotomy(a, b: real; eps: real): real;

**var**

x: real;

**begin**

**repeat**

x := (a + b) / 2;

**if** f(a) \* f(x) > 0 **then**

a := x

**else**

b := x

**until** (abs(a - b) <= eps) **or** (f(x) = 0);

result := x

**end**;

**begin**

writeln('x= ', dihotomy(0, 1, eps):2:2)

**end**.

**Ответ:**

x= 0.44

**Программная реализация:**

(метод простых итераций)

**program** lb2;

**const**

eps = 1e-3;

**function** g(x: real): real;

**begin**

result := ln(2 - x)

**end**;

**function** iterations(a, b: real; eps: real): real;

**var**

x: real;

**begin**

result := a;

**repeat**

x := g(result);

result := g(x)

**until** abs(result - x) <= eps;

**end**;

**begin**

writeln('x= ', iterations(0, 1, eps):2:2)

**end**.

**Ответ:**

x= 0.44

**Программная реализация:**

(метод Ньютона)

**program** lb3;

**const**

eps = 1e-3;

**function** f(x: real): real;

**begin**

result := exp(x) + x - 2

**end**;

**function** f1(x: real): real;

**begin**

result := exp(x) + 1

**end**;

**function** f2(x: real): real;

**begin**

result := exp(x)

**end**;

**function** newton(a, b: real; eps: real): real;

**var**

x: real;

**begin**

**if** f(a) \* f2(a) > 0 **then**

result := a

**else**

**if** f(b) \* f2(b) > 0 **then**

result := b

**else**

**begin**

writeln('Метод Ньютона, решений нет!');

**exit**

**end**;

**repeat**

x := result;

result := x - f(x) / f1(x)

**until** abs(result - x) <= eps;

**end**;

**begin**

writeln('x= ', newton(0, 1, eps):2:2)

**end**.

**Ответ:**

x= 0.44

**3)**

Интервал [0.5; 1.5], допустимая точность 0.2\*10-4;

**Программная реализация:**

(метод половинного деления)

**program** lb1;

**const**

eps = 0.2e-4;

**function** f(x: real): real;

**begin**

result := ln(x) + x

**end**;

**function** dihotomy(a, b: real; eps: real): real;

**var**

x: real;

**begin**

**repeat**

x := (a + b) / 2;

**if** f(a) \* f(x) > 0 **then**

a := x

**else**

b := x

**until** (abs(a - b) <= eps) **or** (f(x) = 0);

result := x

**end**;

**begin**

writeln('x= ', dihotomy(0.5, 1.5, eps):2:2)

**end**.

**Ответ:**

x= 0.57

**Программная реализация:**

(метод простых итераций)

**program** lb2;

**const**

eps = 0.2e-4;

**function** g(x: real): real;

**begin**

result := exp(-x);

**end**;

**function** iterations(a, b: real; eps: real): real;

**var**

x: real;

**begin**

result := a;

**repeat**

x := g(result);

result := g(x)

**until** abs(result - x) <= eps;

**end**;

**begin**

writeln('x= ', iterations(0.5, 1.5, eps):2:2)

**end**.

**Ответ:**

x= 0.57

**Программная реализация:**

(метод Ньютона)

**program** lb3;

**const**

eps = 0.2e-4;

**function** f(x: real): real;

**begin**

result := ln(x) + x

**end**;

**function** f1(x: real): real;

**begin**

result := 1 / x + 1

**end**;

**function** f2(x: real): real;

**begin**

result := -1 / sqr(x)

**end**;

**function** newton(a, b: real; eps: real): real;

**var**

x: real;

**begin**

**if** f(a) \* f2(a) > 0 **then**

result := a

**else**

**if** f(b) \* f2(b) > 0 **then**

result := b

**else**

**begin**

writeln('Метод Ньютона, решений нет!');

**exit**

**end**;

**repeat**

x := result;

result := x - f(x) / f1(x)

**until** abs(result - x) <= eps;

**end**;

**begin**

writeln('x= ', newton(0.5, 1.5, eps):2:2)

**end**.

**Ответ:**

x= 0.57

**4)**

Интервал [0.2; 1.5], допустимая точность 0.5\*10-4;

**Программная реализация:**

(метод половинного деления)

**program** lb1;

**const**

eps = 0.5e-4;

**function** f(x: real): real;

**begin**

result := 2 \* x - exp(-0.1 \* x)

**end**;

**function** dihotomy(a, b: real; eps: real): real;

**var**

x: real;

**begin**

**repeat**

x := (a + b) / 2;

**if** f(a) \* f(x) > 0 **then**

a := x

**else**

b := x

**until** (abs(a - b) <= eps) **or** (f(x) = 0);

result := x

**end**;

**begin**

writeln('x= ', dihotomy(0.2, 1.5, eps):2:2)

**end**.

**Ответ:**

x= 0.48

**Программная реализация:**

(метод простых итераций)

**program** lb2;

**const**

eps = 0.5e-4;

**function** g(x: real): real;

**begin**

result := exp(-0.1 \* x) / 2

**end**;

**function** iterations(a, b: real; eps: real): real;

**var**

x: real;

**begin**

result := a;

**repeat**

x := g(result);

result := g(x)

**until** abs(result - x) <= eps;

**end**;

**begin**

writeln('x= ', iterations(0.2, 1.5, eps):2:2)

**end**.

**Ответ:**

x= 0.48

**Программная реализация:**

(метод Ньютона)

**program** lb3;

**const**

eps = 0.5e-4;

**function** f(x: real): real;

**begin**

result := 2 \* x - exp(-0.1 \* x)

**end**;

**function** f1(x: real): real;

**begin**

result := (20 \* exp(x / 10) + 1) / 10 \* exp(x / 10)

**end**;

**function** f2(x: real): real;

**begin**

result := -1 / 100 \* exp(x / 10)

**end**;

**function** newton(a, b: real; eps: real): real;

**var**

x: real;

**begin**

**if** f(a) \* f2(a) > 0 **then**

result := a

**else**

**if** f(b) \* f2(b) > 0 **then**

result := b

**else**

**begin**

writeln('Метод Ньютона, решений нет!');

**exit**

**end**;

**repeat**

x := result;

result := x - f(x) / f1(x)

**until** abs(result - x) <= eps;

**end**;

**begin**

writeln('x= ', newton(0.2, 1.5, eps):2:2)

**end**.

**Ответ:**

x= 0.48

**Вывод:** Входе лабораторной работы я изучил численные методы решений нелинейных уравнений (метод половинного деления, метод простых итераций, метод Ньютона). Составил программы для вычисления уравнений по данным методам.