**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. Циттель

(Подпись)

\_\_16\_\_\_ \_ \_ апреля\_\_\_\_\_\_\_\_\_\_\_ 2020 г.

Отчет принят:

Преподаватель

доцент ОХИ ИШПР, к.т.н. В.А. Чузлов

(Подпись)

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

Томск 2020 г.

**Цель работы:** научиться составлять программы итерационных методов решения нелинейных уравнений.

**Теоретическая часть**

Нелинейные уравнения можно разделить на 2 класса - алгебраические и трансцендентные. Алгебраическими уравнениями называют уравнения, содержащие только алгебраические функции (целые, рациональные, иррациональные). В частности, многочлен является целой алгебраической функцией. Уравнения, содержащие другие функции (тригонометрические, показательные, логарифмические и др.) называются трансцендентными. Методы решения нелинейных уравнений делятся на две группы: − точные методы; − итерационные методы (за счет последовательных приближений получить решение уравнения с необходимой точностью). Точные методы решения уравнений основываются на поиске равносильных преобразований алгебраических выражений, например, перенос слагаемых из одной части уравнения в другую с противоположным знаком, деление обеих частей уравнения на одинаковое число не равное 0, а также точные способы решений позволяют записать корни уравнения в виде некоторого конечного соотношения (формулы). Точные решения существуют только для некоторых уравнений определенного вида (линейные, квадратные, тригонометрические и др.), поэтому для большинства уравнений приходится использовать методы приближенного решения с заданной точностью (графические или численные). В первую очередь это относится к большинству трансцендентных уравнений. Доказано также, что нельзя построить формулу, по которой можно было бы решить произвольное алгебраическое уравнение выше четвертой степени. Отделение корней уравнения можно осуществить путем построения компьютерных моделей: построение графика функции с помощью одного из языков программирования (в данном случае Free Pascal); построение графика функции в электронных таблицах Microsoft Excel путем построения диаграммы типа График. При построении графика функции корни уравнения можно получить лишь с небольшой степенью точности. Поэтому, чтобы эти значения получить с любой заданной степенью точности, необходимо применять методы, которые позволяют «уточнять» найденные значения. Рассмотрим методы уточнения корней и их основные идеи. Отметим следующий момент: при прочих равных условиях, тот метод уточнения корней будет более эффективен, в котором результат с той же погрешностью найден за меньшее число раз вычисления функции f(x). Метод половинного деления.

Самый простой из них – метод половинного деления, или иначе метод дихотомии. Метод дихотомии получил свое название от древнегреческого слова διχοτομία, что в переводе означает деление надвое. Его мы используем довольно часто. Допустим, играя в игру "Угадай число", где один игрок загадывает число от 1 до 100, а другой пытается его отгадать, руководствуясь подсказками "больше" или "меньше". Логично предположить, что первым числом будет названо 50, а вторым, в случае если оно меньше - 25, если больше - 75. Таким образом, на каждом этапе неопределенность неизвестного x3 x1 x2 X Y y=f(x) Приближенные методы Отделение корней Решение уравнений с заданной точностью Графический метод f(x)=0, где f(x) - непрерывная функция Численные методы Метод половинного деления Метод хорд Метод касательных Комбинированный метод уменьшается в 2 раза. Т.е. даже самый невезучий в мире человек отгадает загаданное число в данном диапазоне за 7 предположений вместо 100 случайных утверждений. Алгоритм метода половинного деления основан на теореме Больцано - Коши о промежуточных значениях непрерывной функции и следствии из неё. Теорема Больцано - Коши: если непрерывная функция принимает два значения, то она принимает любое значение между ними. Следствие (теорема о нуле непрерывной функции): если непрерывная функция принимает на концах отрезка положительное и отрицательное значения, то существует точка, в которой она равна 0.

**Практическая часть**

**Задание 1**

**Исходные данные**:

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

**Задание**

Составьте программу для решения нелинейных уравнений методом половинного деления, простых итераций и методом Ньютона.

**Программная реализация**

**1 способ:**

**Program** lab9;

**const**

eps = 1e-2;

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

**begin**

result:= exp(ln(x) \* 4) + 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('Корень уравнения равен: ', dihotomy(1, 2, eps))

**end**.

**Ответ:**

Корень уравнения равен: 1.9453125

**2 способ:**

**Program** lab9;

**const**

eps = 1e-2;

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

**begin**

result := exp(ln(20 - 3 \* x) \* 1 / 4)

**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('Корень уравнения равен: ', iterations(1, 2, eps))

**end**.

**Ответ:**

Корень уравнения равен: 1.94037733840934

**3 способ:**

**Program** lab9;

**const**

eps = 1e-2;

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

**begin**

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

**end**;

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

**begin**

result := 4 \* exp(ln(x) \* 3) + 3

**end**;

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

**begin**

result := 12 \* exp(ln(x) \* 2)

**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('Корень уравнения равен: ', newton(1, 2, eps))

**end**.

**Ответ:**

Корень уравнения равен: 1.94047935224908

**Задание 2**

**Исходные данные**:

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

**Задание**

Составьте программу для решения нелинейных уравнений методом половинного деления, простых итераций и методом Ньютона.

**Программная реализация**

**1 способ:**

**Program** lab9;

**const**

eps = 1e-3;

**function** f(x: 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('Корень уравнения равен: ', dihotomy(0, 1, eps))

**end**.

**Ответ:**

Корень уравнения равен: 0.4423828125

**2 способ:**

**Program** lab9;

**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('Корень уравнения равен: ', iterations(0, 1, eps))

**end**.

**Ответ:**

Корень уравнения равен: 0.442509950010955

**3 способ:**

**Program** lab9;

**const**

eps = 0.2 \* 1e-4;

**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('Корень уравнения равен: ', newton(0, 1, eps))

**end**.

**Ответ:**

Корень уравнения равен: 0.4423828125

**Задание 3**

**Исходные данные**:

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

**Задание**

Составьте программу для решения нелинейных уравнений методом половинного деления, простых итераций и методом Ньютона

**Программная реализация**

**1 способ:**

**Program** lab9;

**const**

eps = 0.2 \* 1e-4;

**function** f(x: 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('Корень уравнения равен: ', dihotomy(0.5, 1.5, eps))

**end**.

**Ответ:**

Корень уравнения равен: 0.567143290399369

**2 способ:**

**Program** lab9;

**const**

eps = 0.2 \* 1e-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('Корень уравнения равен: ', iterations(0.5, 1.5, eps))

**end**.

**Ответ:**

Корень уравнения равен: 0.567140763269807

**3 способ:**

**Program** lab9;

**const**

eps = 0.2 \* 1e-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 / (x \* 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('Корень уравнения равен: ', newton(0.5, 1.5, eps))

**end**.

**Ответ:**

Корень уравнения равен: 0.567143290399369

**Задание 4**

**Исходные данные**:

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

**Задание**

Составьте программу для решения нелинейных уравнений методом половинного деления, простых итераций и методом Ньютона.

**Программная реализация**

**1 способ:**

**Program** lab9;

**const**

eps = 0.5 \* 1e-4;

**function** f(x: 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('Корень уравнения равен: ', dihotomy(0.2, 1.5, eps))

**end**.

**Ответ:**

Корень уравнения равен: 0.476721637710739

**2 способ:**

**Program** lab9;

**const**

eps = 0.5 \* 1e-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('Корень уравнения равен: ', iterations(0.2, 1.5, eps))

**end**.

**Ответ:**

Корень уравнения равен: 0.476721637710739

**3 способ:**

**Program** lab9;

**const**

eps = 0.5 \* 1e-4;

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

**begin**

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

**end**;

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

**begin**

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

**end**;

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

**begin**

result := 0.1 \* exp(-0.1 \* 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('Корень уравнения равен: ', newton(0.2, 1.5, eps))

**end**.

**Ответ:**

Корень уравнения равен: 0.476721637710739

**Выводы**

В ходе работы мы научились составлять программы итерационных методов решения нелинейных уравнений.