МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра математического обеспечения и применения ЭВМ

ОТЧЕТ

по практической работе №5

по дисциплине «Вычислительная математика»

Тема: Метод Ньютона

Студент гр. 7383 _____ Кирсанов А.Я.

Преподаватель _____ Сучков А.И.

Санкт-Петербург

2018

Цель работы.

Изучить метод Ньютона для нахождения простого корня уравнения, обусловленность и скорость сходимости этого метода.

Основные теоретические положения.

В случае, когда известно хорошее начальное приближение решения уравнения $f(x)=0$, эффективным методом повышения точности является метод Ньютона. Он состоит в построении итерационной последовательности

$$x_{n+1}=x_n-\frac{f(x_n)}{f'(x_n)},$$

сходящейся к корню уравнения $f(x)=0$.

Метод Ньютона допускает простую геометрическую интерпретацию, как показано на рис. 1.

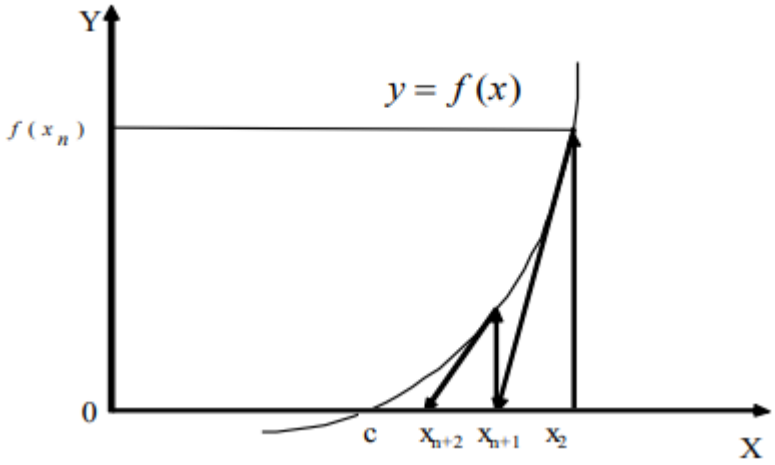

Рисунок 1 – Метод Ньютона

Если через точку с координатами $(x_n; f(x_n))$ провести касательную, то абсцисса точки пересечения этой касательной с осью Ox будет очередным приближением x_{n+1} корня уравнения $f(x)=0$.

Для оценки погрешности n-го приближения корня предлагается пользоваться неравенством

$$|c - x_n| \le \frac{M_2}{2m_1}|x_n - x_{n-1}|^2,$$

где M_2 - наибольшее значение модуля второй производной $|f''(x)|$ на отрезке $[a,b]$, m_1 – наименьшее значение модуля первой производной $|f'(x)|$ на отрезке $[a,b]$. Таким образом, если $|x_n - x_{n-1}| < \varepsilon$, то $|c - x_n| < \frac{M_2\varepsilon}{2m_1}$. Это означает, что при хорошем начальном приближении корня после каждой итерации число верных десятичных знаков в очередном приближении удваивается, т.е. процесс сходится очень быстро (имеет место квадратичная сходимость). Из указанного следует, что при необходимости нахождения корня с точностью ε, итерационный процесс можно прекращать, когда

$$|x_n - x_{n-1}| < \varepsilon_0 = \sqrt{\frac{2m_1\varepsilon}{M_2}}.$$

Пусть между абсолютными погрешностями входных данных X и решения Y установлено неравенство

$$\Delta(y^*) \le \nu_\Delta \Delta(x^*), \qquad (1)$$

где x^* и y^* – приближенные входные данные и приближенное решение. Тогда величина ν_Δ называется абсолютным числом обусловленности.

Если рассматривать задачу вычисления корня уравнения $Y = f(X)$, то роль числа обусловленности будет играть величина

$$\nu_\Delta = \frac{1}{|f'(\bar{x})|}, \qquad (2)$$

где $\bar{x}$ – корень уравнения.

Теоретически метод Ньютона имеет квадратичную сходимость, следовательно, для скорости сходимости метода верна следующая оценка:

$$|x_n - \bar{x}| \le \frac{1}{\sigma}|x_{n-1} - \bar{x}|, n \ge 0,$$

где $\bar{x}$ – корень уравнения, x_n – очередное приближение корня, σ – погрешность.

Следствием этой оценки является априорная оценка:

$$\left|x_n - \bar{x}\right| \le \sigma q^{2^n}, n \ge 0, \tag{3}$$

в которой $q = \sigma^{-1}\left|x_0 - \bar{x}\right|$.

Для проверки этого утверждения используется апостериорная оценка

$$\left|x_n - \bar{x}\right| \le \left|x_n - x_{n-1}\right|. \tag{4}$$

Постановка задачи.

Порядок выполнения:

1) Графически или аналитически отделить корень уравнения $f(x) = 0$ (т.е. найти отрезки $[Left, Right]$, на котором функция $f(x)$ удовлетворяет условиям сходимости метода Ньютона).

2) Составить подпрограммы – функции вычисления $f(x)$, $f'(x)$, предусмотрев округление их значений с заданной точностью `Delta`.

3) Составить головную программу, вычисляющую корень уравнения $f(x) = 0$ и содержащую обращение к подпрограммам , $f(x)$, $f'(x)$, `Round`, `NEWTON` и индикацию результатов.

4) Выбрать начальное приближение корня x_0 из $[Left, Right]$ так, чтобы $f(x) f''(x) > 0$.

5) Провести вычисления по программе. Исследовать скорость сходимости метода и чувствительность метода к ошибкам в исходных данных. Для приближенного вычисления корней уравнения $f(x) = 0$ методом Ньютона предназначена программа – функция `NEWTON`.

Выполнение работы.

Файлы программ methods.cpp и methods.h взяты из директории LIBR1. В текст программы methods.cpp добавлена подпрограмма вычисления функции $f(x)$ и $f'(x)$. Программа main.cpp написана согласно требованиям, описанным

в постановке задачи. Тексты программы main.cpp, methods.cpp methods.h и представлены в приложениях А, Б и В соответственно.

В задании, согласно варианту, использовалась функция

$$f(x) = 2x^2 - x^4 - 1 - \ln(x). \quad (5)$$

На рис. 2 представлен график функции (4). По нему можно найти границы отрезка, на котором расположен корень функции. Левая граница a равна 0.3, так как в 0 функция стремится к ∞. Правая граница b равна 1.2. В качестве начального приближения корня x_0 взята правая граница b.

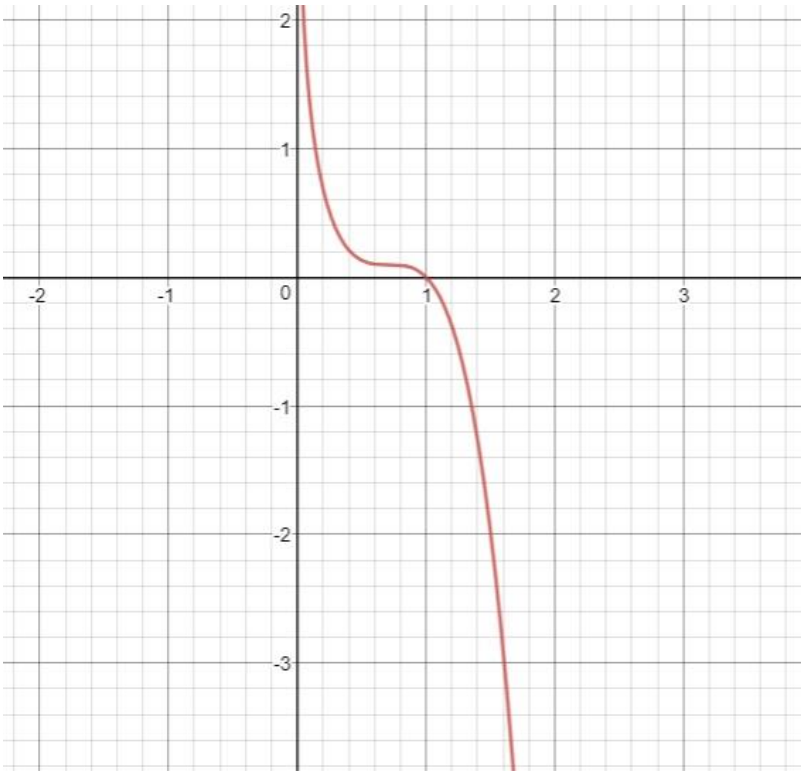

Рисунок 2 – График функции (5)

В качестве теоретических данных выступает число обусловленности ν_Δ, рассчитываемое по формуле (2). Обусловленность задачи определяется выражением (1), в котором $\Delta(y^*)$ = `Eps`, а $\Delta(x^*)$ = `Delta`.

Таблица 1 – Результаты вычислений при меняющемся `Delta`

Значение `delta`	Значение x	Значение k	Значение ν_Δ	Обусловленность
0.00001	1.000019	5	1	Хорошая

Окончание таблицы 1

0.0001	0.999975	5	1	Хорошая
0.001	1.000261	5	1	Хорошая
0. 01	1.002520	4	1	Хорошая
0.1	1.038380	3	1	Плохая

Таблица 2 – Результаты вычислений при меняющемся `Eps`

Значение `Eps`	Значение x	Значение k	Значение ν_{Δ}	Обусловленность
0.00001	1.000261	5	1	Плохая
0.0001	1.000261	5	1	Плохая
0.001	1.000261	5	1	Хорошая
0.01	1.000261	5	1	Хорошая
0.1	1.026498	3	1	Плохая

Для проверки априорной оценки скорости сходимости (3) используем апостериорную оценку (4). Результаты подстановки значений $\bar{x}$, x_n, x_{n-1} в неравенство (4) при `Delta` = 0.0000001 и `EPS` = 0.0000001 представлены в табл. 3.

Таблица 3 – Результаты вычислений неравенства

Номер итерации	Значение корня на данной итерации	Значение левой части неравенства (5)	Значение правой части неравенства (5)
1	1.25902	0.259021	0.260979
2	1.10594	0.105938	0.153083
3	1.02635	0.0263548	0.0795833
4	1.00214	0.0021423	0.0242125
5	0.999975	2.49779e-005	0.00216727

Выводы.

В работе изучен метод Ньютона для нахождения простого корня уравнения, обусловленность этого метода, зависимость числа итераций от точности задания исходных данных. Задача хорошо обусловлена, если выполняется неравенство (1). Это подтверждается экспериментальными результатами (см. табл. 1 и табл. 2).

При повышении точности вычисления корня Eps число итераций увеличивается.

Скорость сходимости метода Ньютона постоянна при неизменяющемся отрезке поиска корня, при этом из оценки (3) следует, что имеет место квадратичная скорость сходимости, то есть при хорошем начальном приближении корня после каждой итерации число верных десятичных знаков в очередном приближении удваивается. Это подтверждается выполнимостью неравенства (4) на каждом шаге итерации (см. табл. 3).

Скорость сходимости метода Ньютона выше скорости сходимости метода хорд и метода бисекции, но метод Ньютона требует, чтобы на каждом шаге итерации существовала ненулевая производная функции (5) в точке приближения корня.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

```
#include<cmath>
#include<iostream>
#include<cstdlib>
#include"METHODS.H"
double delta,c,d;
int main(){
    int k;
    float eps1,delta1;
    double a,b,eps,x;
    printf(" eps:");
    scanf("%f",&eps1);
    eps = eps1;
    a = 0.1;
    b = 1.5;
    printf(" delta:");
    scanf("%f",&delta1);
    delta = delta1;
    x = NEWTON(b,eps,k);
    printf("x=%f    k=%d\n",x,k);
    return 0;
}
```

ПРИЛОЖЕНИЕ Б

ИСХОДНЫЙ КОД ФУНКЦИЙ

```
#include <stdio.h>
#include <cmath>
#include <stdlib.h>
#include <iostream>
#include <conio.h>
#include "methods.h"
using namespace std;
double n, n1;
void speed(double X, int &N);
double Round (double X,double Delta)
{
 if (Delta<=1E-9) {puts("nevok\n");exit(1);}
 if (X>0.0) return (Delta*(long((X/Delta)+0.5)));
    else    return (Delta*(long((X/Delta)-0.5)));
}

double F(double x)
{
extern double delta;
double s;
long int S;
s = 2*pow(x, 2)-pow(x, 4) - 1 - log(x);
if( s/delta < 0 )
S = s/delta - .5;
else
S = s/delta + .5;
s = S*delta;
s = Round( s,delta );
return(s);
}

double F1(double x)
{
extern double delta;
double s;
long int S;
s = 4*x-4*pow(x, 3) - 1/x;
if( s/delta < 0 )
S = s/delta - .5;
else
S = s/delta + .5;
```

```
s = S*delta;
s = Round( s,delta );
return(s);
}

double NEWTON (double X,double Eps,int &N)
{
  extern double F1 (double);
  double Y,Y1,DX;
  N=0;
  do
  {
      n1 = X;
    Y  = F(X);
    if (Y==0.0) return (X);

    Y1 = F1(X);
    if (Y1==0.0) {puts("prjo0\n");exit(1);}

    DX=Y/Y1; X=X-DX; N++;
    speed(X, N);
  }
  while (fabs(DX)>Eps);
  return (X);
}

void speed(double X, int &N){
    cout << N << ": " << X << " " << abs(X-1) << " <= " << abs(X - n1) <<
endl;
}
```

ПРИЛОЖЕНИЕ В

ИСХОДНЫЙ КОД ЗАГОЛОВОЧНОГО ФАЙЛА

```
extern double F(double);
double F1(double x);
double Round (double X,double Delta);
double NEWTON (double X,double Eps,int &N);
```