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

ОТЧЕТ

по практической работе №6

по дисциплине «Вычислительная математика»

Тема: Метод простых итераций

Студент гр. 7383 _____ Кирсанов А.Я.

Преподаватель _____ Сучков А.И.

Санкт-Петербург

2018

Цель работы.

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

Основные теоретические положения.

Метод простых итераций решения уравнения $f(x)=0$ состоит в замене исходного уравнения эквивалентным ему уравнением $x=\varphi(x)$ и построении последовательности $x_{n+1}=\varphi(x_n)$, сходящейся при $n\to\infty$ к точному решению. Достаточные условия сходимости метода простых итераций формулируются теоремой ниже.

Рассмотрим один шаг итерационного процесса. Исходя из найденного на предыдущем шаге значения x_{n-1}, вычисляется $y=\varphi(x_{n-1})$. Если $|y-x_{n-1}|>\varepsilon$, то полагается $x_n=y$ и выполняется очередная итерация. Если же $|y-x_{n-1}|<\varepsilon$, то вычисления заканчиваются и за приближенное значение корня принимается величина $x_n=y$. Погрешность результата вычислений зависит от знака производной $\varphi'(x)$: при $\varphi'(x)>0$ погрешность определения корня составляет $\frac{q\varepsilon}{1-q}$, а при $\varphi'(x)<0$, погрешность не превышает ε. Здесь q - число, такое, что $|\varphi'(x)|<q<1$ на отрезке $[a,b]$. Существование числа q является условием сходимости метода в соответствии с отмеченной выше теоремой.

Для применения метода простых итераций определяющее значение имеет выбор функции $\varphi(x)$ в уравнении $x=\varphi(x)$, эквивалентном исходному. Функцию $\varphi(x)$ необходимо подбирать так, чтобы $|\varphi'(x)|<q<1$. Это обусловливается тем, что если $\varphi'(x)<0$, то последовательные приближения будут сходиться к корню с монотонно. Следует также помнить, что скорость сходимости последовательности $\{x_n\}$ к корню функции $f(x)$ тем выше, чем меньше число q.

Преобразование уравнения $f(x)=0$ к виду $x=\varphi(x)$ изменяет обусловленность задачи. Поэтому оценка обусловленности в данном случае выглядит так:

$$\Delta(x^*) \le \nu\Delta(\varphi^*), \tag{1}$$

где

$$\nu = \frac{1}{1-q}. \tag{2}$$

Для оценки величины q вблизи корня используется следующее соотношение:

$$q \approx \alpha_{n-1} = \frac{x_n - x_{n-1}}{x_{n-1} - x_{n-2}}. \tag{3}$$

Метод итераций сходится со скоростью геометрической прогрессии со значением q и имеет место оценка:

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

где $\bar{x}$ - корень уравнения $f(x)$.

Постановка задачи.

Порядок выполнения:

1) Графически или аналитически отделить корень уравнения $f(x)=0$.

2) Преобразовать уравнение $f(x)=0$ к виду $x=\varphi(x)$ так, чтобы в некоторой окрестности $[Left, Right]$ корня производная $\varphi'(x)$ удовлетворяла условию $|\varphi'(x)| < q < 1$. При этом следует иметь в виду, что чем меньше величина q, тем быстрее последовательные приближения сходятся к корню.

3) Выбрать начальное приближение, лежащее на $[Left, Right]$.

4) Составить подпрограмму для вычисления значений $\varphi(x)$, $\varphi'(x)$, предусмотрев округление вычисленных значений с точностью `Delta`.

5) Составить головную программу, вычисляющую корень уравнения и содержащую обращение к программам $\varphi(x)$, $\varphi'(x)$ и ITER и индикацию результатов.

6) Провести вычисления по программе. Исследовать скорость сходимости и обусловленность метода.

Выполнение работы.

Файлы программ methods.cpp и methods.h взяты из директории LIBR1. В текст программы methods.cpp добавлена подпрограмма вычисления функции $f(x)$. 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. 1 представлен график функции (5). По нему можно найти границы отрезка, на котором расположен корень функции $\bar{x} = 1$. Левая граница *a* равна 0.3, правая граница *b* равна 1.2. В качестве начального приближения корня x_0 взята правая граница *b*.

Функция (5) была преобразована к функции $\varphi(x)$ вида

$$x = e^{2x^2 - x^4 - 1}, \quad (6)$$

график которой представлен на рис. 2.

Также в качестве вспомогательной функции на рис. 2 присутствует функция:

$$x = y. \quad (7)$$

В качестве теоретических данных выступает число обусловленности ν_Δ, рассчитываемое по формулам (2) и (3). Обусловленность задачи определяется выражением (1).

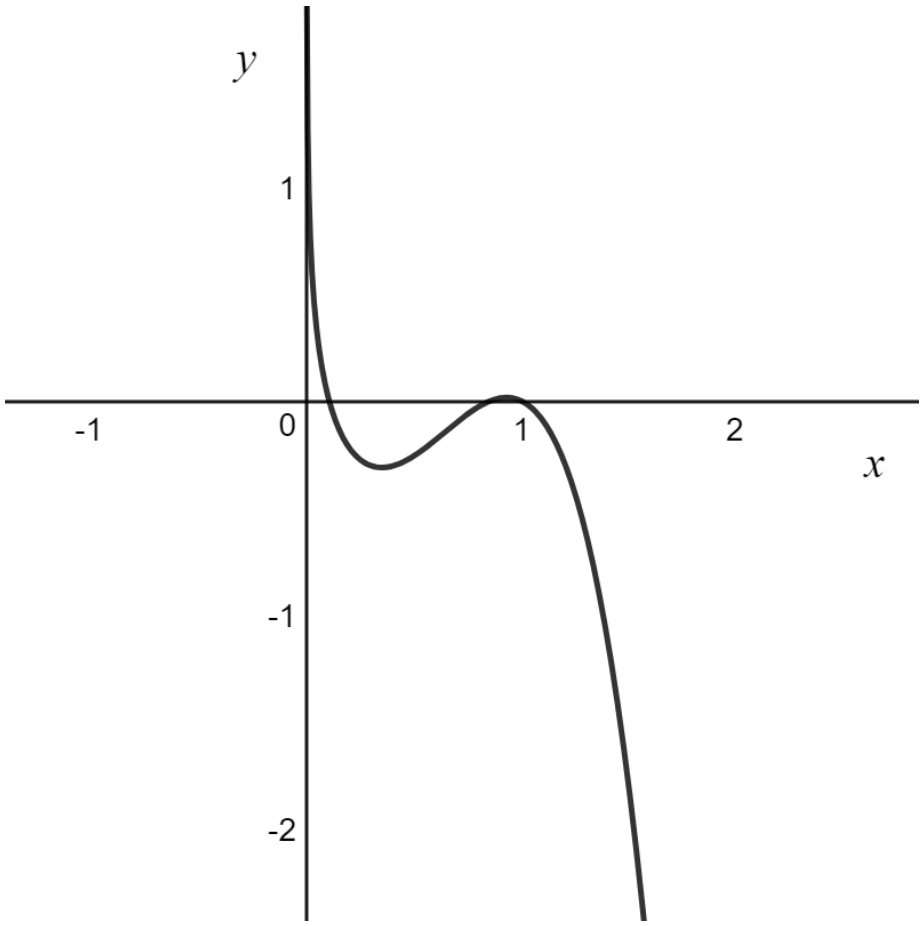

Рисунок 1 – График функции (5)

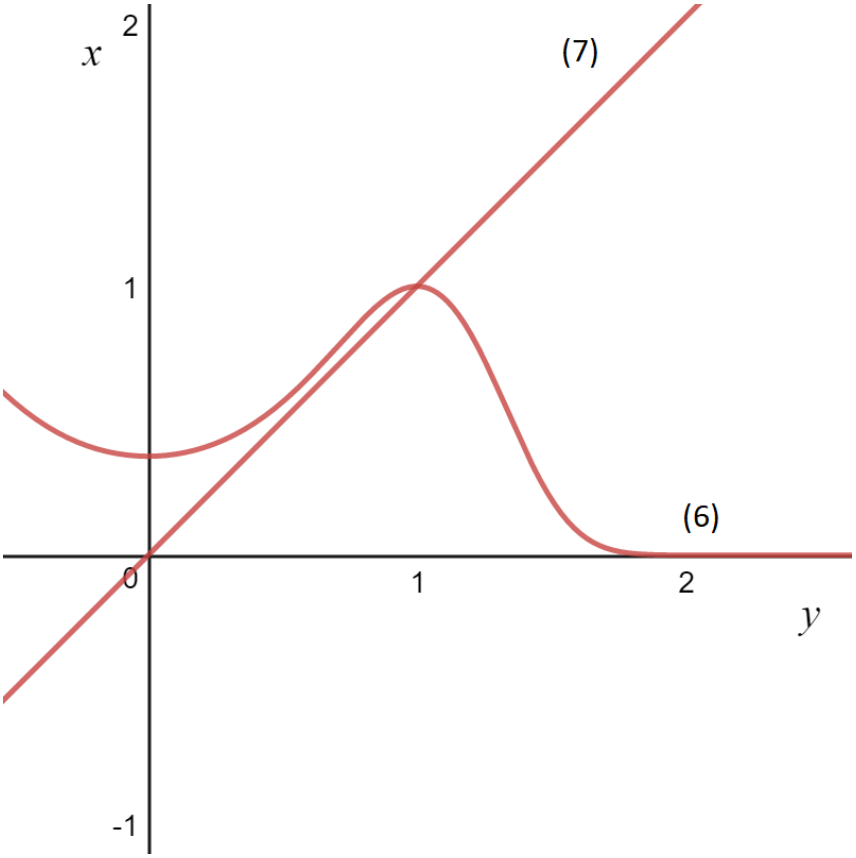

Рисунок 2 – График функций (6) и (7)

В табл. 1 представлены экспериментальные значения корня и числа итераций, теоретические значения абсолютного числа обусловленности и обусловленность задачи при меняющемся от 0.00001 до 0.1 значении `delta`, `Eps` = 0.001.

В табл. 2 представлены экспериментальные значения корня и числа итераций, теоретические значения абсолютного числа обусловленности и обусловленность задачи при меняющемся от 0.00001 до 0.1 значении `Eps`, `delta` = 0.001.

Таблица 1 – Результаты вычислений при меняющемся `Delta`

Значение `delta`	Значение x	Значение k	Значение ν_Δ	Обусловленность
0.00001	0.999960	7	1.13	Хорошая
0.0001	1.000000	7	1.13	Хорошая
0.001	1.000000	7	1.13	Хорошая
0. 01	0.990000	7	1.13	Хорошая
0.1	1.038380	5	1.13	Плохая

Таблица 2 – Результаты вычислений при меняющемся `Eps`

Значение `Eps`	Значение x	Значение k	Значение ν_Δ	Обусловленность
0.00001	1.000000	8	1.13	Хорошая
0.0001	1.000000	8	1.13	Хорошая
0.001	1.000261	7	1.13	Хорошая
0.01	0.757000	2	1.2	Плохая
0.1	0.757000	2	1.2	Плохая

Для проверки априорной оценки скорости сходимости 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	0.686845	0.167168	0.874621
2	0.756509	0.0896035	0.485021
3	0.832832	0.0288768	0.219019
4	0.910397	0.00323459	0.0187411
5	0.971123	4.16883e-005	0.000454117
6	0.996765	1.16861e-008	5.51818e-007
7	0.999958	1.16861e-008	1.543923e-008

Выводы.

В работе изучен метод простых итераций 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).

Скорость сходимости метода постоянна при неизменяющемся отрезке поиска корня, при этом из априорной оценки $|x_n - \bar{x}| \le q^n |x_0 - \bar{x}|$ следует, что имеет место скорость сходимости геометрической прогрессии. Это подтверждается выполнимостью неравенства (4) на каждом шаге итерации (см. табл. 3).

Скорость сходимости метода простых итераций выше скорости сходимости и метода бисекции, но ниже метода Ньютона. Недостатком метода является требование к выполнению условия $|\varphi'(x)| < q < 1$.

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 = 0.3;
    b = 1.2;
    printf(" delta:");
    scanf("%f",&delta1);
    delta = delta1;
    x = ITER(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;
#define e 2,7182818284
double alpha, nu;

double Round (double X,double Delta)
{
 if (Delta<=1E-9) {puts("Incorrect accuracy");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 = exp(2*pow(x,2)-pow(x,4)-1);//sqrt(sqrt(2*pow(x,2)-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 ITER(double X0,double Eps,int &N)
{
    double x[100];
  if (Eps<=0.0) {puts("Incorrect accuracy setting");exit(1);}
  double X1=F(X0);
  x[0]=X1;
  double X2=F(X1);
  x[1]=X2;
  N = 2;
```

```
  while( (X1 - X2)*(X1 - X2) > fabs((2*X1-X0-X2)*Eps) )
  {
     X0 = X1;
     X1 = X2;
     X2 = F(X1);
    x[N]=X2;
     N++;
  }
  cout << N << endl;
  alpha = 0.7;
  for(int i = 0; i<N-2; i++){
      alpha = (x[i+2]-x[i+1])/(x[i+1]-x[i]);
      cout << i << " nu: " << 1/(1-alpha) << " l: " << abs(x[i+2]-1) << "
<= " << abs(alpha/(1-alpha)*(x[i+2]-x[i+1]));
      printf(" x=%f    k=%d\n",x[i],N);
  }

   return(X2);
}
```

ПРИЛОЖЕНИЕ В

ИСХОДНЫЙ КОД ЗАГОЛОВОЧНОГО ФАЙЛА

```
extern double F(double);
double Round (double X,double Delta);
double ITER(double X0,double Eps,int &N);
```