**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра математического обеспечения и применения ЭВМ**

отчет

**по практическому заданию №5**

**по дисциплине «Вычислительная математика»**

Тема: **Решение нелинейных уравнений. Метод простых итераций**

|  |  |  |
| --- | --- | --- |
| Студент гр. 6304 |  | Виноградов К.А. |
| Преподаватель |  | Лисс А.Р. |

Санкт-Петербург

2017

**Цель работы.**

Изучение метода простых итераций как итерационного метода приближенного решения нелинейных уравнений.

**Основные теоретические положения.**

Задача нахождения корней нелинейных уравнений вида

, (1)

где - некоторая непрерывная функция, встречается в различных областях инженерной и научной деятельности. Нелинейные уравнения делятся на два класса - алгебраические и трансцендентные. Алгебраическими называются уравнения, содержащие только алгебраические функции (целые, рациональные, иррациональные). Уравнения, которые содержат другие функции (тригонометрические, показательные, логарифмические и т.п.), называются трансцендентными.

Методы решения нелинейных уравнений подразделяются на прямые и итерационные. Прямые методы позволяют записать корни в виде некоторого конечного соотношения. Такие методы для решения ряда трансцендентных, а также простейших алгебраических уравнений известны из школьного курса алгебры. Однако встречающиеся на практике уравнения не удается решить столь простыми методами. Для их решения применяются итерационные методы, при которых алгоритм нахождения корня уравнения в общем случае включает два этапа:

* отыскания приближенного значения корня или содержащего его отрезка;
* уточнения приближенного значения до некоторой заданной степени точности.

Пусть задана непрерывная функция  вещественного аргумента  и требуется численным методом решить уравнение (1), т.е. найти приближение к вещественному корню этого уравнения. Если уравнение имеет несколько вещественных корней, то сначала производят их отделение (изоляцию), а затем уточняют положение отдельного корня. Считается, что отделение корня произведено, если выделен такой интервал [] области определения функции , на концах которого значения функции  и  имеют разные знаки и внутри которого имеется ровно один корень уравнения (1).

Метод простых итераций решения уравнения (1) состоит в замене исходного уравнения эквивалентным ему уравнением

 (2)

и построении последовательности , сходящейся при n→∞ к точному решению.

Рассмотрим один шаг итерационного процесса. Исходя из найденного на предыдущем шаге значения , вычисляется . Если , то полагается  и выполняется очередная итерация. Если же , то вычисления заканчиваются и за приближенное значение корня принимается величина . Погрешность результата вычислений зависит от знака производной : при  погрешность определения корня составляет , а при  погрешность не превышает . Здесь  - число, такое, что

 (3)

на отрезке []. Существование числа  является условием сходимости метода в соответствии с отмеченной выше теоремой.

Для применения метода простых итераций определяющее значение имеет выбор функции  в уравнении , эквивалентном исходному. Функцию  необходимо подбирать так, чтобы . Это обусловливается тем, что если  на отрезке [], то последовательные приближения  будут колебаться около корня c, если же , то последовательные приближения будут сходиться к корню c монотонно. Следует также помнить, что скорость сходимости последовательности {} к корню c функции  тем выше, чем выше число .

**Выполнение работы.**

Дана функция 

Графически определим отрезок, на котором находится корень уравнения. График представлен на рис. 2.

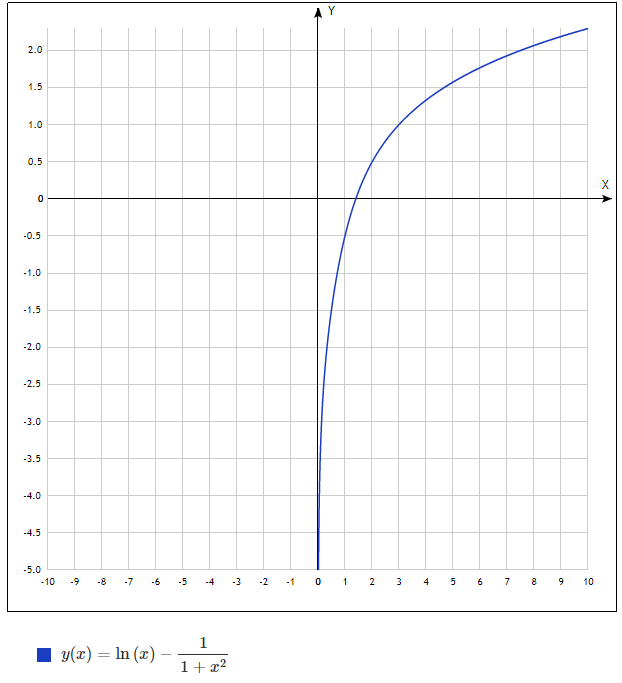


Рисунок 2 – График уравнения

Нужный отрезок ­­– [1, 2].

Преобразуем уравнение (1) к виду (2) так, чтобы в некоторой окрестности (Left, Right) корня производная  удовлетворяла условию (3):

 (4)

Тогда значение производной  при  удовлетворяет условию (3) на интервале [1, 2].

Пусть начальное приближение из [1, 2] будет равно 1.5.

Составим подпрограмму вычисления функции ,, предусмотрев округление вычисленных значений с точностью Delta.. Код подпрограммы находится в приложении А.

Составим головную программу вычисляющую корень уравнения и содержащую обращение к программам ,  и ITER и индикацию результатов. Код программы находится в приложении А.

Для нахождения числа абсолютной обусловленности возьмём производную от :

 (5)

Соответственно, число обусловленности можно легко получить из (5), как модуль обратной величины:

 (6)

Исследуем чувствительность метода к ошибкам в исходных данных. Для этого зафиксируем значение погрешности результата Eps = 0.0001. А значение погрешности исходных данных будем изменять. Кроме того, зафиксируем количество итераций, которое понадобилось для вычисления корня. Число обусловленности будем вычислять по формуле (6). Результаты вычислений представлены в табл. 1.

Таблица 1 – Вычисления при постоянном Eps и переменном Delta

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Eps | Delta | x\* | k |  |
| 0.00001 | 0.1 | 1.400000 | 2 | 0.967247 |
| 0.00001 | 0.01 | 1.400000 | 2 | 0.970859 |
| 0.00001 | 0.001 | 1.401000 | 3 | 0.968717 |
| 0.00001 | 0.0001 | 1.401300 | 2 | 0.968307 |
| 0.00001 | 0.00001 | 1.401290 | 2 | 0.968340 |

окончание таблицы 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0.00001 | 0.000001 | 1.401292 | 2 | 0.968341 |

Для расчёта скорости сходимости метода воспользуемся формулой:

, (7)

где *С*(0, 1), *k* .

Возьмём  из последнего опыта в табл. 1, полагая **, и оценим значение *p*. Результаты представлены в табл. 2.

Таблица 2 – Скорость сходимости метода простых итераций

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| 1 | 1.402230 | 0,000938 | 0,000002 |
| 2 | 1.401290 | 0,000002 | 0,000028 |
| 3 | 1,401320 | 0,000028 | 0,000028 |

Из этого следует, что  при . Значит скорость сходимости линейная.

**Выводы.**

В ходе данной практической работы был высчитан корень функции  методом простых итераций с заданным значением точности.

Также из табл. 2 было получено, что  при **, т.е. скорость сходимости метода простых итераций – линейная.

Кроме того, исследовав чувствительность метода к ошибкам в исходных данных, можно сделать вывод, что данная задача обладает хорошей обусловленностью, т.к.  не превышает 1.

Приложение А

Код программы

#include <math.h>

#include <stdio.h>

#include <stdlib.h>

extern double F(double);

extern double F1(double);

double Round (double X,double Delta);

double ITER(double X0,double Eps,int &N);

double delta;

int main()

{

int k;

long int s;

float eps1,delta1;

double a,b,eps,x,x1,f\_der;

double F(double);

printf("Insert eps:");

scanf("%f",&eps1);

eps = eps1;

printf("Insert x0:");

scanf("%f",&x1);

x = x1;

printf("Insert delta:");

scanf("%f",&delta1);

delta = delta1;

x = ITER(x,eps,k);

f\_der=F1(x);

printf("x=%f k=%d derivative x=%f",x,k,f\_der);

return 0;

}

double F(double x)

{

extern double delta;

double s;

long int S;

s = x-(log(x)-1/(1+x\*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 = -2\*x/((x\*x+1)\*(x\*x+1))-1/x+1;

if( s/delta < 0 )

S = s/delta - .5;

else

S = s/delta + .5;

s = S\*delta;

s = Round( s,delta );

return(s);

}

double Round (double X,double Delta)

{

if (Delta<=1E-9) {puts("Неверное задание точности округления\n");exit(1);}

if (X>0.0) return (Delta\*(long((X/Delta)+0.5)));

else return (Delta\*(long((X/Delta)-0.5)));

}

double ITER(double X0,double Eps,int &N)

{

if (Eps<=0.0) {puts("Неверное задание точности\n");exit (1);}

double X1=F(X0);

double X2=F(X1);

N = 2;

while( (X1 - X2)\*(X1 - X2) > fabs((2\*X1-X0-X2)\*Eps) )

{

X0 = X1;

X1 = X2;

X2 = F(X1);

N++;

}

return(X2);

}