**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра математического обеспечения и применения ЭВМ**

отчет

**по практическому заданию №4**

**по дисциплине «Вычислительная математика»**

Тема: **Решение нелинейных уравнений. Метод Ньютона**

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

Санкт-Петербург

2017

**Цель работы.**

Найти корень уравнения  методом Ньютона с заданной точностью Eps, исследовать обусловленность метода (чувствительность к ошибкам в исходных данных) и скорость сходимости метода.

**Основные теоретические положения.**

Пусть задана непрерывная функция  вещественного аргумента x и требуется численным методом решить уравнение , т.е. найти приближение x\* 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 [a0, b0] области определения функции , на концах которого значения функции (a0) и (b0) имеют разные знаки и внутри которого имеется ровно один корень уравнения . В случае, когда известно хорошее начальное приближение решения уравнения , эффективным методом повышения точности является метод Ньютона. Он состоит в построении итерационной последовательности



сходящейся к корню уравнения .

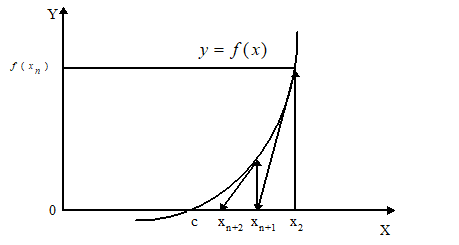


Рисунок 1 – Метод Ньютона

Метод Ньютона допускает простую геометрическую интерпретацию (рис. 1). Если через точку с координатами  провести касательную, то абсцисса точки пересечения этой касательной с осью Ох будет очередным приближением xn+1 корня уравнения .

Для оценки погрешности n-го приближения корня предлагается пользоваться неравенством

 (1)

где М2-наибольшее значение модуля второй производной  на отрезке [a,b]; m1-наименьшее значение модуля первой производной  на отрезке [a,b].

Таким образом, если



то



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

 (2)

Рассмотрим один шаг итераций. Если на (n-1)-м шаге очередное приближение xn-1 неудовлетворяет условию окончания процесса, то вычисляются величины  и следующие приближение корня  При выполнении условия (2) величина xn принимается за приближенное значение корня с, вычисленное с точностью ε.

**Выполнение работы.**

Дана функция 

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

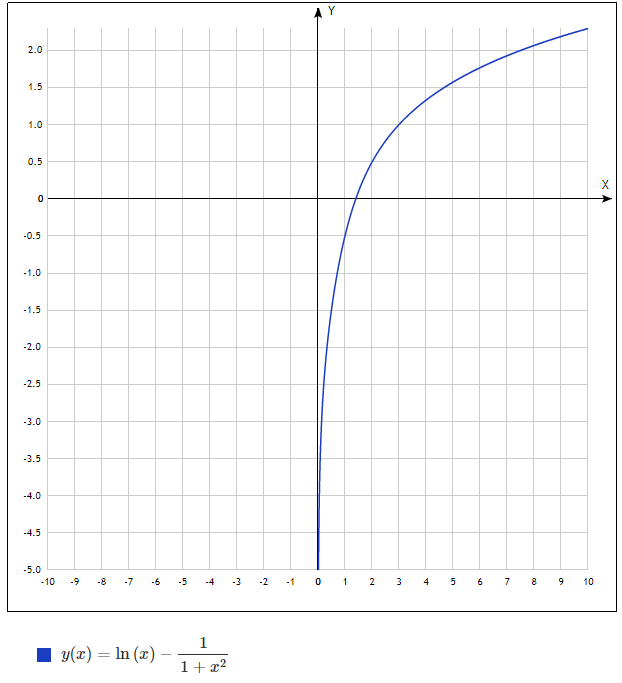


Рисунок 2 – График уравнения

Нужный отрезок ­­– [1.0; 2.0].

1. Составим подпрограммы вычисления функции и f’(x). Код подпрограммы находится в приложении А.
2. Составим головную программу содержащую обращение к функциям , NEWTON, Round и индикации результатов. Код программы находится в приложении А.
3. Выберем начальное приближение  из [1,2] так, чтобы выполнялось условие f(x)f’(x)>0. Пусть x
4. Проведем вычисления по программе, варьируя значения параметра  
   Eps (точность вычисления корня) и Delta (точность задания исходных данных). Результаты приведены в табл. 1 и табл. 2 соответственно.

Таблица 1 – Вычисления при постоянном Delta и переменном Eps

|  |  |  |  |
| --- | --- | --- | --- |
| Eps | Delta | x\* | k |
| 0.1 | 0.001 | 1.418906 | 2 |
| 0.01 | 0.001 | 1.404349 | 3 |
| 0.001 | 0.001 | 1.401135 | 4 |
| 0.0001 | 0.001 | 1.401135 | 4 |
| 0.00001 | 0.001 | 1.401135 | 4 |
| 0.000001 | 0.001 | 1.401135 | 4 |

Для нахождения числа абсолютной обусловленности возьмём производную от :

 (3)

Соответственно, число обусловленности можно легко получить из (3), как модуль обратной величины:

 (4)

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 (4). Результаты вычислений представлены в табл. 2.

Таблица 2 – Вычисления при постоянном Eps и переменном Delta

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Eps | Delta | x\* | k |  |
| 0.001 | 0.1 | 1.416667 | 1 | 0.705882 |
| 0.001 | 0.01 | 1.404688 | 2 | 0.711902 |
| 0.001 | 0.001 | 1.401135 | 4 | 0.713707 |
| 0.001 | 0.0001 | 1.401828 | 4 | 0.713354 |
| 0.001 | 0.00001 | 1.401834 | 4 | 0.713351 |
| 0.001 | 0.000001 | 1.401834 | 4 | 0.713351 |

Для расчёта скорости сходимости метода воспользуемся формулой:

, (5)

где *С*(0, 1), *k* .

Возьмём  из четвёртого и пятого опытов в табл. 2, полагая  и *k*=1,2, оценим значение *p*. Результаты представлены в табл. 3.

Таблица 3 – Результаты вычисления скорости сходимости метода хорд

|  |  |
| --- | --- |
|  |  |
| 0,00286 | 0,0029678 |
| 0,000699 | 0,0005708 |

Из этого следует, что p ~ 1. Значит скорость сходимости линейная.

**Выводы.**

В ходе выполненной работы был вычислен корень данного уравнения и зависимость обусловленности от точности входных данных. Согласно результатам вычислений, данный метод позволяет поучить результат с требуемой точностью за меньшее число итераций, чем метод бисекции. Также была исследована скорость сходимости данного метода. Так как он итеративный, то скорость сходимости линейна.

Приложение А

Код программы

#include <math.h>

#include <stdio.h>

#include <stdlib.h>

double Round (double x,double delta)

{

if (x>0.0) return (delta\*(long((x/delta)+0.5)));

else return (delta\*(long((x/delta)-0.5)));

}

double F(double x, double delta)

{

double s;

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

s = Round(s,delta );

return(s);

}

double HORDA(double Left,double Right,double Eps,double delta, int &N)

{

double FLeft = F(Left, delta);

double FRight = F(Right, delta);

double X,Y;

if (FLeft\*FRight>0.0) {puts("Error\n");exit(1);}

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

if (FLeft==0.0) return Left;

if (FRight==0.0) return Right;

do

{

X = Left-(Right-Left)\*FLeft/(FRight-FLeft);

Y = F(X, delta);

if (Y == 0.0) return (X);

if (Y\*FLeft < 0.0)

{ Right=X; FRight=Y; }

else

{ Left=X; FLeft=Y; }

N++;

}

while ( fabs(Y) >= Eps );

return(X);

}

int main()

{

float eps, delta;

double x;

int k=0;

double F(double);

printf("Enter Epsilon:");

scanf("%f",&eps);

printf("Enter delta:");

scanf("%f",&delta);

x = HORDA(1, 2, eps, delta, k);

printf("x=%f\n",x);

printf("k=%d\n", k);

return 0;

}