**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 |  | Пискунов Я.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 .

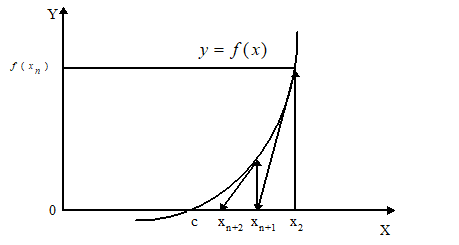


Рисунок 1 – Иллюстрация метода Ньютона

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.

**Ход работы.**

В соответствии с номером варианта выбрана функция:

.

Графически определим отрезок, на котором находится корень уравнения. Для этого построим график данной функции. Он представлен на рис. 2.

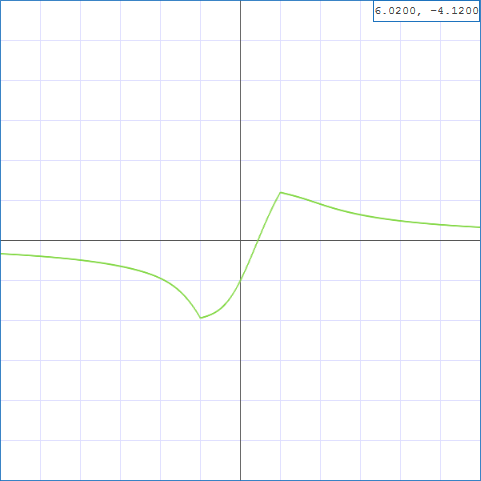
Как видно из этого рисунка, корень находится на отрезке [0, 1]. Именно эти значения мы и используем в качестве начальных границ отрезка локализации корня.

Рисунок 2 – График функции.

Составлены подпрограммы вычисления функции  и .

Составлена головная программа, содержащая обращение к подпрограмме , , NEWTON, Round и индикацию результатов. Программа представлена в приложении А.

Выберем начальное приближение  из [0,1] так, чтобы выполнялось условие:

. (3)

Пусть 

Для нахождения числа абсолютной обусловленности возьмём производную от f(x):

 (4)

Соответственно, число обусловленности можно легко получить отсюда, как модуль обратной величины:

 (5)

Исследуем чувствительность метода к ошибкам в исходных данных. Для этого зафиксируем значение погрешности результата Eps = 0.00001. А значение погрешности исходных данных будем изменять. Кроме того, зафиксируем количество итераций, которое понадобилось для вычисления корня. Число обусловленности будем вычислять по формуле (5). Результаты вычислений представлены в табл. 2.

Таблица 1 – Результаты вычисления программы при изменении Delta

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Eps | Delta | x\* | N |  |
| 0.00001 | 0.1 | 0.409524 | 3 | 0.726445 |
| 0.00001 | 0.01 | 0.413029 | 4 | 0.730588 |
| 0.00001 | 0.001 | 0.412336 | 6 | 0.723500 |
| 0.00001 | 0.0001 | 0.412119 | 7 | 0.724035 |
| 0.00001 | 0.00001 | 0.412105 | 8 | 0.724065 |
| 0.00001 | 0.000001 | 0.412104 | 8 | 0.724059 |

Для расчёта скорости сходимости метода воспользуемся формулой:

. (6)

Возьмём  из всех, начиная с второго, опытов в табл. 1, оценим значение p при . Результаты представлены в табл. 2.

Таблица 2 – Вычисление скорости сходимости метода Ньютона

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 0.002580 | 0.006656 | 0.000925 |
| 0.000925 | 0.000856 | 0.000232 |
| 0.000232 | 0.000054 | 0.000015 |
| 0.000015 | 0.000003 | 0.000001 |

Из этого следует, что p ~ 2 при . Значит скорость сходимости квадратичная.

**Выводы.**

В ходе данной лабораторной работы был высчитан корень функции *f(x)* методом Ньютона с заданным значением точности. Также из табл. 2 было получено, что p ~ 2 при , т.е. скорость сходимости метода хорд – квадратичная. Кроме того, исследовав чувствительность метода к ошибкам в исходных данных, можно сделать вывод, что данная задача обладает хорошей обусловленностью.

По сравнению с другими методами, методу Ньютона необходимо большее число итераций для нахождения корня, однако при этом точность у него выше.

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 NEWTON (double X,double Eps,int &N);

double delta;

int main()

{

int k;

long int s;

float eps1,delta1;

double a,b,eps,x,x1;

double F(double);

printf("Введите eps:");

scanf("%f",&eps1);

eps = eps1;

printf("Введите x0:");

scanf("%f",&x1);

x = x1;

printf("Введите delta:");

scanf("%f",&delta1);

delta = delta1;

x = NEWTON(x,eps,k);

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

return 0;

}

double F(double x)

{

extern double delta;

double s;

long int S;

s = powl(x,2)-powl(x,3)-1/(4+powl(x,2));

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 = asin(2\*x/(1+powf(x,2)))-exp(-powf(x,2));

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 NEWTON (double X,double Eps,int &N)

{

extern double F1 (double);

double Y,Y1,DX;

N=0;

do

{

Y = F(X);

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

Y1 = F1(X);

if (Y1==0.0) {puts("Производная обратилась в ноль\n");exit(1);}

DX=Y/Y1; X=X-DX; N++;

}

while (fabs(DX)>Eps);

return (X);

}