**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. 8381 |  | Киреев К.А. |
| Преподаватель |  | Щеголева Н.Л. |

Санкт-Петербург

2019

**Цель работы.**

Найти корень уравнения  с заданной точностью *ε* методом Ньютона, исследовать скорость сходимости и обусловленность метода.

**Основные теоретические положения.**

В случае, когда известно хорошее начальное приближение решения уравнения , эффективным методом повышения точности является метод Ньютона. Он состоит в построении итерационной последовательности

, (1)

сходящейся к корню уравнения . Достаточные условия сходимости метода формулируются теоремой.

**Теорема:**

Пусть  определена и дважды дифференцируема на [*a*, *b*] причем , а производные , сохраняют знак на отрезке [*a*, *b*]. Тогда, исходя из начального приближения  [*a*, *b*], удовлетворяющего неравенству , можно построить последовательность

, n = 0, 1, 2, …, (2)

сходящуюся к единственному на [*a*, *b*] решению уравнения .

Метод Ньютона допускает простую геометрическую интерпретацию, пример изображен на рис. 1. Если через точку с координатами  провести касательную, то абсцисса точки пересечения этой касательной с осью абсцисс будет очередным приближением  корня уравнения .

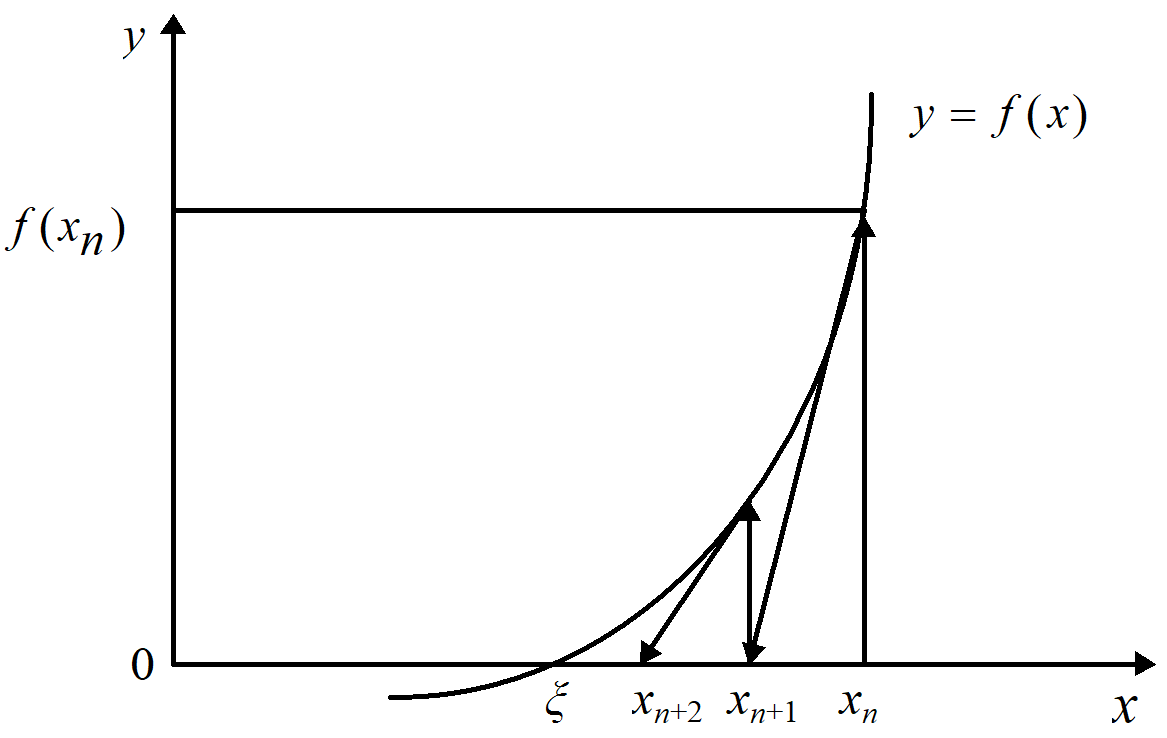


Рисунок 1 – Метод Ньютона

Для оценки погрешности *n*-го приближения корня предлагается пользоваться неравенством

, (3)

где *М*2 – наибольшее значение модуля второй производной  на отрезке [*a*, *b*]; *m*1 – наименьшее значение модуля первой производной  на отрезке [*a*, *b*]. Таким образом, если , то

. (4)

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 квадратическая сходимость). Из указанного следует, что при необходимости нахождения корня с точностью *ε* итерационный процесс можно прекращать, когда

. (5)

Рассмотрим один шаг итераций. Если на (*n*-1)-м шаге очередное приближение *xn*-1 не удовлетворяет условию окончания процесса, то вычисляются величины ,  и следующие приближение корня

. (6)

При выполнении условия (4) величина *xn* принимается за приближенное значение корня , вычисленное с точностью *ε*.

**Постановка задачи.**

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

**Выполнение работы.**

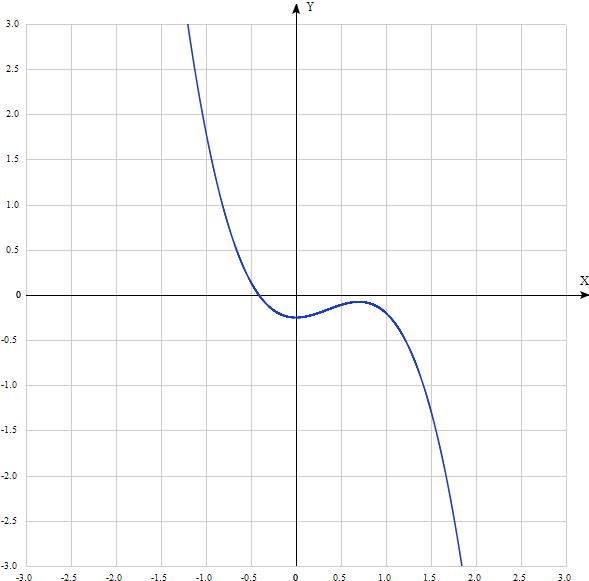


Рисунок 2 – График функции 

Графически функция  удовлетворяет условиям сходимости метода Ньютона на промежутке [-1, 0.5]. На данном промежутке функция имеет единственный вещественный корень *x*\* = -0.412105.

Определим абсолютное число обусловленности задачи вычисления корня

(7)

для этого вычислим производную функции 

, (8)

следовательно, абсолютное число обусловленности имеет вид

, (9)

тогда абсолютное число обусловленности  = 0.724058.

Выберем начальное приближение корня  из промежутка [-1, 0.5] так, чтобы оно удовлетворяло неравенству  и знаки функций  и  в точке *x*0 совпадали. По значениям из табл. 1 начальное приближение корня *x0 =* -0.1.

Таблица 1 – Начальное приближение *x0*

|  |  |  |  |
| --- | --- | --- | --- |
| *x0* |  |  |  |
| -1 | 1.8 | -5.08 | 8.016 |
| -0.7 | 0.610283 | -2.93944 | 6.2559 |
| -0.4 | -0.0163846 | -1.32623 | 4.49779 |
| -0.1 | -0.238377 | -0.242438 | 2.72314 |
| 0.2 | -0.215525 | 0.304507 | 0.917684 |

Проведем ряд вычислений для функции  на промежутке [-1, 0.5], используя код программы и изменяя значения точности вычисления корня и точности задания исходных данных.

Таблица 2 – Метод Ньютона

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | *N* | *x*\* |  |
| 0.1 | 0.1 | 4 | -0.404906 | 0.72406 |
| 0.1 | 0.01 | 4 | -0.418753 | 0.741638 |
| 0.1 | 0.001 | 4 | -0.42017 | 0.725124 |
| 0.1 | 0.0001 | 4 | -0.420421 | 0.723321 |
| 0.1 | 0.00001 | 4 | -0.420385 | 0.724064 |
| 0.1 | 0.000001 | 4 | -0.420387 | 0.724064 |
| 0.01 | 0.1 | 4 | -0.404906 | 0.775822 |
| 0.01 | 0.01 | 5 | -0.411661 | 0.731023 |
| 0.01 | 0.001 | 5 | -0.412413 | 0.736426 |
| 0.01 | 0.0001 | 5 | -0.412248 | 0.736285 |
| 0.01 | 0.00001 | 5 | -0.412218 | 0.736334 |
| 0.01 | 0.000001 | 5 | -0.40705 | 0.736332 |
| 0.001 | 0.1 | 4 | -0.412216 | 0.731023 |
| 0.001 | 0.01 | 5 | -0.411926 | 0.724487 |
| 0.001 | 0.001 | 5 | -0.411657 | 0.725134 |
| 0.001 | 0.0001 | 6 | -0.411643 | 0.725165 |
| 0.001 | 0.00001 | 6 | -0.411643 | 0.725166 |
| 0.001 | 0.000001 | 6 | -0.391668 | 0.775822 |
| 0.0001 | 0.1 | 4 | -0.391668 | 0.775822 |
| 0.0001 | 0.01 | 5 | -0.40922 | 0.731023 |
| 0.0001 | 0.001 | 5 | -0.411926 | 0.724487 |
| 0.0001 | 0.0001 | 6 | -0.412081 | 0.724115 |
| 0.0001 | 0.00001 | 6 | -0.412062 | 0.724162 |
| 0.0001 | 0.000001 | 6 | -0.412062 | 0.72416 |
| 0.00001 | 0.1 | 4 | -0.391668 | 0.775822 |
| 0.00001 | 0.01 | 5 | -0.40922 | 0.731023 |
| 0.00001 | 0.001 | 5 | -0.411926 | 0.724487 |
| 0.00001 | 0.0001 | 6 | -0.412081 | 0.724115 |
| 0.00001 | 0.00001 | 6 | -0.412104 | 0.72406 |
| 0.00001 | 0.000001 | 6 | -0.412098 | 0.724076 |
| 0.000001 | 0.1 | 4 | -0.391668 | 0.775822 |
| 0.000001 | 0.01 | 5 | -0.40922 | 0.731023 |
| 0.000001 | 0.001 | 5 | -0.411926 | 0.724487 |
| 0.000001 | 0.0001 | 6 | -0.412081 | 0.724115 |
| 0.000001 | 0.00001 | 6 | -0.412104 | 0.72406 |
| 0.000001 | 0.000001 | 6 | -0.412104 | 0.72406 |

Из полученных результатов можно сделать следующие выводы: с заданием более высокой точности входных данных возрастает количество итераций, а с ростом ошибок в исходных данных, уменьшается точность выходных данных.

Исследуем скорость сходимости метода Ньютона для функции  с вещественным корнем *x*\* = -0.412105. Начальное приближение для функции *x0 =* -0.1. Так как выполняется условие , то скорость сходимости для функции  определяется неравенством . При выборе достаточно близких начальных приближений метод Ньютона обладает квадратичной скоростью сходимости, т.к. после каждой итерации число верных десятичных знаков в очередном приближении удваивается. Для заданной функции *M*2 = 2, *m*1 = 1, значит формула  для функции  примет вид

, (10)

**Выводы.**

Проанализировав результаты работы, мы можем вывод, что число итераций метода Ньютона возрастает с ростом требуемой точности входных данных. Обусловленность задачи нахождения корня уравнения  для функции  прямо пропорциональна величине  и точности задания исходных данных и обратно пропорциональна точности вычисления корня. Задачу можно считать хорошо обусловленной, так как в данном случае при вычислении приближенного значения корня функции с заданной точностью методом Ньютона количество утерянных верных цифр мало.

Приложение А

исходный код программы

#include <math.h>

#include <stdlib.h>

#include <iostream>

using namespace std;

double Round(double X, double Delta);

double F(double x, double Delta);

double F1(double x, double Delta);

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

void Calculate(double xn, double eps, double Delta) {

    int n = 0;

    double x = NEWTON(xn, eps, n, Delta);

    double ob = 1/fabs(-3\*x\*x+2\*x+(2\*x/(((x\*x)+4)\*((x \* x) + 4))));

    cout << "Delta: " << Delta << "\tEps: "<< eps << endl;

cout << "Функция приимает значение 0 при Х = " << x << endl;

cout << "Число интераций: " << n;

    cout << "\tЧисло обусловленности: " << ob << endl << endl;

}

int main(){

    double x, e = 1, d = 1;

    cout << "Введите начальное приближение корня: ";

cin >> x;

    for (int i = 0; i < 6; i++) {

        e = e \* 0.1;

        for (int j = 0; j < 6; j++){

            d = d \* 0.1;

            Calculate(x, e, d);

        }

        d = 1;

    }

    return 0;

}

double Round(double X, double Delta){

    if (Delta <= 1E-9) {

        cout << "Неверное задание точности округления" << endl;

        return 0;

    }

    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 val = x \* x - x \* x \* x - (1 / (4 + x \* x));

    return Round(val, Delta);

}

double F1(double x, double Delta){

    double valpr = -3\*x\*x+2\*x+(2\*x/(((x\*x)+4)\*((x \* x) + 4)));

return Round(valpr, Delta);

}

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

double Y, Y1, DX;

do{

Y = F(X,Delta);//Значение фунции в очередном прибилижении

if (Y == 0.0)

return X;

Y1 = F1(X, Delta);//Значение производной фунции в очередном прибилижении

if (Y1 == 0.0){

cout << "Производная обратилась в ноль\n";

return 0;

        }

DX = Y / Y1;

X -= DX;//Следующее приближение корня

N++;

}while (fabs(DX) > Eps);//|x[n]-x[n-1] > e|

    return X;

}