**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. 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.**

Метод простых итераций (метод последовательных приближений)

решения уравнения   состоит в замене исходного уравнения эквивалентным ему уравнением   и построении последовательности , сходящейся при  к точному решению. Достаточные условия сходимости метода простых итераций формулируются следующей теоремой.

**Теорема.** Пусть функция   определена и дифференцируема на [a,b], причём все её значения . Тогда, если существует число q, такое, что   на отрезке [a,b] , то последовательность , n = 0, 1, 2,…, сходится к единственному на [a,b] решению уравнения   при любом начальном значении , т.е.

 (1)

При этом если на отрезке [a,b] производная  положительна, то

 (2)

,если  отрицательна, то

 (3)

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 . Существование числа q 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 [a,b], то последовательные приближения  будут колебаться около корня , если же  то последовательные приближения будут сходиться к корню  монотонно. Следует также помнить, что скорость сходимости последовательности  к корню   функции тем выше, чем меньше число q.

**Постановка задачи.**

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

**Выполнение работы.**

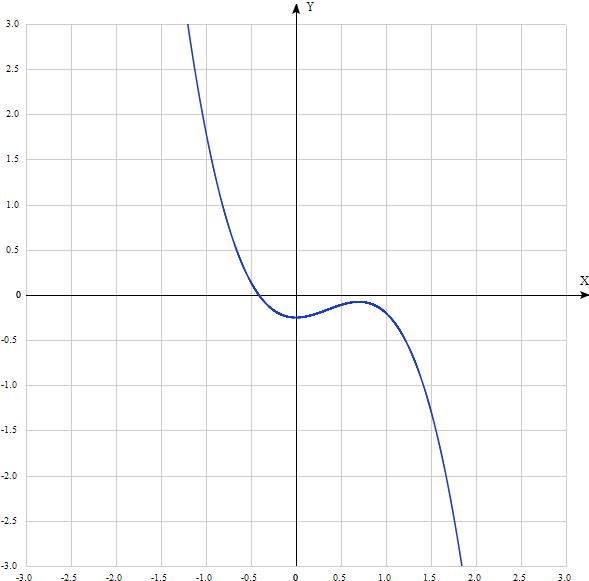


Рисунок 1 – График функции 

Графически функция  удовлетворяет условиям сходимости метода простых итераций на промежутке [-1, 1.5]. На данном промежутке функция имеет единственный вещественный корень *x*\* = -0.412105.

Определим абсолютное число обусловленности задачи вычисления корня

(4)

для этого вычислим производную функции 

, (5)

следовательно, абсолютное число обусловленности имеет вид

, (6)

тогда абсолютное число обусловленности  = 0.724058.

Выберем начальное приближение корня  из промежутка [-1, 1.5] 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 |

Преобразование уравнения f(x) = 0 к виду , на промежутке [-1, 1.5], содержащем корень *x*\*, удовлетворяет условию .  ⬄ 

Так как  на промежутке [-1, 1.5], то 

Проведем ряд вычислений для функции  на промежутке [-1, 1.5], используя код программы и изменяя значения точности вычисления корня и точности задания исходных данных.

Таблица 2 – Метод простых итераций

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | *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 сходимости (1). При этом если на отрезке [a,b] производная  положительна, то . Существование числа q является условием сходимости метода. Скорость сходимости последовательности к корню тем выше, чем меньше число q.

**Выводы.**

Проанализировав результаты работы, мы можем вывод, что число итераций метода простых итераций 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 простых итераций количество утерянных верных цифр мало.

Приложение А

исходный код программы

#include <iostream>

#include <math.h>

#include <stdlib.h>

using namespace std;

double Round(double X, double Delta);

double PHI(double x, double Delta);

double ITER(double x, double Eps, int &N, double Delta);

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

{

int n = 0;

double x = ITER(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 PHI(double x, double Delta)

{

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

    return val;

}

double ITER(double x, double Eps, int &N, double Delta)

{

    if (Eps <= 0.0)

    { cout << "Неверное задание точности" << endl; exit(1); }

    double x1 = PHI(x, Delta);

  double x2 = PHI(x1, Delta);

  for (N = 2; (x1 - x2) \* (x1 - x2) > fabs((2 \* x1 - x - x2) \* Eps); N++)

    {

    x = x1;

    x1 = x2;

    x2 = PHI(x1, Delta);

  }

    return x2;

}