**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра математического обеспечения и применения ЭВМ**

отчет

**по практической работе №4**

**по дисциплине «Вычислительная математика»**

Тема: Метод хорд

|  |  |  |
| --- | --- | --- |
| Студентка гр. 8381 |  | Киреев К.А. |
| Преподаватель |  | Щеголева Н.Л. |

Санкт-Петербург

2019

**Цель работы.**

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

**Основные теоретические положения.**

Пусть найден отрезок [a, b], на котором функция меняет знак. Для определенности положим f(a)>0, f(b)<0. В методе хорд процесс итераций состоит в том, что в качестве приближений к корню уравнения f(x)=0 принимаются значения c0,c1,… точек пересечения хорды с осью абсцисс, как это показано на рис. 1

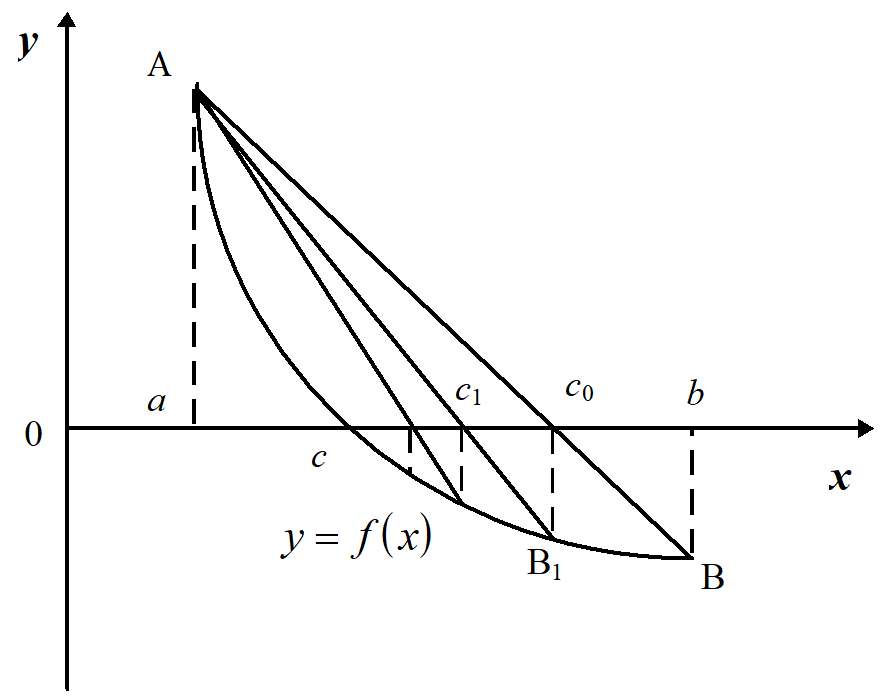


Рисунок 1. Геометрическая интерпретация метода хорд

Сначала находится уравнение хорды AB:

 (1)

Для точки пересечения ее с осью абсцисс (x=c0, y=0) получается уравнение

 (2)

Далее сравниваются знаки величин f(a)и f(c0) и для рассматриваемого случая оказывается, что корень находится в интервале (a,c0), так как f(a)f(c0)<0. Отрезок [c0, b] отбрасывается. Следующая итерация состоит в определении нового приближения c1 как точки пересечения хорды AB1 с осью абсцисс и т.д. Итерационный процесс продолжается до тех пор, пока значение f(cn) не станет по модулю меньше заданного числа ε.

Алгоритмы методов бисекции и хорд похожи, однако метод хорд в ряде случаев дает более быструю сходимость итерационного процесса, причем успех его применения, как и метода бисекции, гарантирован.

**Постановка задачи.**

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

**Выполнение работы.**

График функции  изображен на рис. 2.

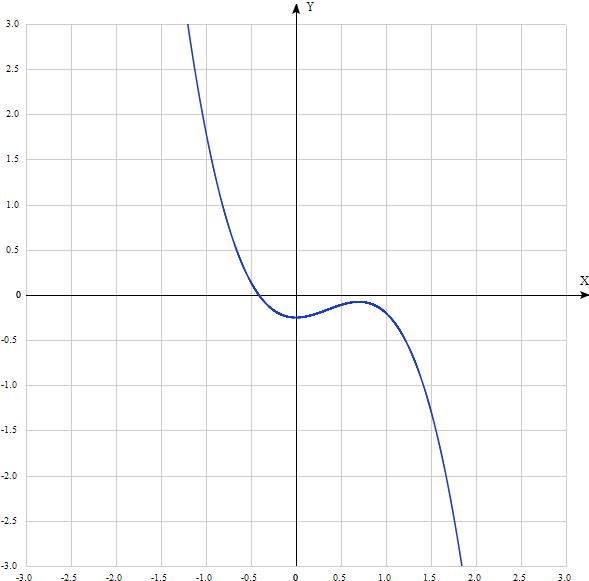


Рисунок 2. График функции 

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

Определим абсолютное число обусловленности задачи вычисления корня

(3)

для этого вычислим производную функции 

 (4)

следовательно, абсолютное число обусловленности имеет вид

 (5)

тогда абсолютное число обусловленности  = 0.724058.

Проведем ряд вычислений для функции  на промежутке [-1, 0.5], изменяя значения точности вычисления корня и точности задания исходных данных. Результаты вычислений предоставлены в табл. 1.

Таблица 1. Метод хорд

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | *N* | *x*\* |  |
| 0.1 | 0.1 | 10 | -0.391668 | 0.775822 |
| 0.1 | 0.01 | 9 | -0.361997 | 0.862408 |
| 0.1 | 0.001 | 9 | -0.359682 | 0.869819 |
| 0.1 | 0.0001 | 9 | -0.359885 | 0.869165 |
| 0.1 | 0.00001 | 9 | -0.359875 | 0.869198 |
| 0.1 | 0.000001 | 9 | -0.359873 | 0.869202 |
| 0.01 | 0.1 | 10 | -0.391668 | 0.775822 |
| 0.01 | 0.01 | 13 | -0.40922 | 0.731023 |
| 0.01 | 0.001 | 13 | -0.407012 | 0.736426 |
| 0.01 | 0.0001 | 13 | -0.407069 | 0.736285 |
| 0.01 | 0.00001 | 13 | -0.407049 | 0.736334 |
| 0.01 | 0.000001 | 13 | -0.40705 | 0.736332 |
| 0.001 | 0.1 | 13 | -0.40922 | 0.731023 |
| 0.001 | 0.01 | 17 | -0.411926 | 0.724487 |
| 0.001 | 0.001 | 17 | -0.411657 | 0.725134 |
| 0.001 | 0.0001 | 17 | -0.411643 | 0.725165 |
| 0.001 | 0.00001 | 17 | -0.411643 | 0.725166 |
| 0.001 | 0.000001 | 10 | -0.391668 | 0.775822 |
| 0.0001 | 0.1 | 10 | -0.391668 | 0.775822 |
| 0.0001 | 0.01 | 13 | -0.40922 | 0.731023 |
| 0.0001 | 0.001 | 17 | -0.411926 | 0.724487 |
| 0.0001 | 0.0001 | 21 | -0.412081 | 0.724115 |
| 0.0001 | 0.00001 | 21 | -0.412062 | 0.724162 |
| 0.0001 | 0.000001 | 21 | -0.412062 | 0.72416 |
| 0.00001 | 0.1 | 10 | -0.391668 | 0.775822 |
| 0.00001 | 0.01 | 13 | -0.40922 | 0.731023 |
| 0.00001 | 0.001 | 17 | -0.411926 | 0.724487 |
| 0.00001 | 0.0001 | 21 | -0.412081 | 0.724115 |
| 0.00001 | 0.00001 | 25 | -0.412104 | 0.72406 |
| 0.00001 | 0.000001 | 24 | -0.412098 | 0.724076 |
| 0.000001 | 0.1 | 10 | -0.391668 | 0.775822 |
| 0.000001 | 0.01 | 13 | -0.40922 | 0.731023 |
| 0.000001 | 0.001 | 17 | -0.411926 | 0.724487 |
| 0.000001 | 0.0001 | 21 | -0.412081 | 0.724115 |
| 0.000001 | 0.00001 | 25 | -0.412104 | 0.72406 |
| 0.000001 | 0.000001 | 28 | -0.412104 | 0.72406 |

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

**Выводы.**

Проанализировав результаты работы, мы можем вывод, что число итераций метода хорд 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 вычисленные методом хорд корни не дают большой разницы с корнем, определенным графически практически во всех ситуациях.

Приложение А

исходный код программы

#include <iostream>

#include <cmath>

using namespace std;

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 HORDA(double Left, double Right, double Eps, int& N, double Delta) {

double FLeft = F(Left, Delta);//Значение функции в точке a

double FRight = F(Right, Delta);//Значение функции в точке b

double X, Y;

if (FLeft \* FRight > 0.0) {

cout << "Неверное задание интервала" << endl;

return 0;

}

if (Eps <= 0.0) {

cout << "Неверное задание интервала" << endl;

return 0;

}

if (FLeft == 0.0)

return Left;

if (FRight == 0.0)

return Right;

do {

X = Left - (Right - Left) \* FLeft / (FRight - FLeft);//Точка пересечения хорды с осью абсцисс

Y = F(X, Delta);//Значение фукнции в точке с[n]

if (Y == 0.0)

return X;

if (Y \* FLeft < 0.0) {//f(c) и f(a) имеют разные знаки

Right = X;

FRight = Y;

}

else {//f(c) и f(b) имеют разные знаки

Left = X;

FLeft = Y;

}

N++;

} while (fabs(Y) >= Eps);

return X;

}

void Calculate(double a, double b, double eps, double Delta) {

int n = 0;

double x = HORDA(a, b, eps, n, Delta);

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

cout << "Eps: " << eps << endl;

cout << "Delta: " << Delta << endl;

cout << "N: " << n << endl;

cout << "X: " << x << endl;

cout << "O: " << ob << endl << endl;

}

int main() {

setlocale(0, "");

double e = 1, d = 1;

for (int i = 0; i < 6; i++) {

e = e \* 0.1;

for (int j = 0; j < 6; j++){

d = d \* 0.1;

Calculate(-1, 0.5, e, d);

}

d = 1;

}

return 0;

}