# МИНОБРНАУКИ РОССИИ

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра МО ЭВМ**

**ОТЧЁТ**

**по лабораторной работе №3**

**по дисциплине «Вычислительная математика» Тема: Метод бисекции**

Студент гр. 8381 Киреев К.А.

Преподаватель Щеголева Н.Л.

Санкт-Петербург 2019

# Цель работы

Формирование практических навыков нахождения корней алгебраических и трансцендентных уравнений методом бисекции.

# Основные теоретические положения

Если найден отрезок *[a, b]*, такой, что *f (a)f (b) < 0*, существует точка *c*, в которой значение функции равно нулю, т.е. *f (c) = 0*, *c ∈ (a, b).* Метод бисекции состоит в построении последовательности вложенных друг в друга отрезков *{[an, bn] | [an, bn] ⊂ [an−1, bn−1] ⊂ [a, b]}*, на концах которых функция имеет разные знаки. Каждый последующий отрезок получается делением пополам предыдущего. Процесс построения последовательности отрезков позволяет найти нуль функции *f (x)* (корень уравнения *f (x) = 0*) с любой заданной точностью.

Рассмотрим один шаг итерационного процесса. Пусть на *(n − 1)-*м шаге найден отрезок *[an-1, bn-1]⊂ [a, b],* такой, что *f (an-1)f (bn-1) < 0*. Разделим его пополам точкой ***ξ*** = *(an-1+ bn-1)/2* и вычислим *f(****ξ****).* Если *f(****ξ****) = 0*, то ***ξ*** = *(an-1+ bn-1)/2* - корень уравнения. Если *f (****ξ****) ≠ 0*, то из двух половин отрезка выбирается та, на концах которой функция имеет противоположные знаки, поскольку искомый корень лежит на этой половине, т.е.

*an = an-1, bn = ξ, если f (ξ)f (an-1) < 0*

*an = ξ, bn = bn-1, если f (ξ)f (an-1) > 0*

Если требуется найти корень с точностью ε, то деление пополам продолжается до тех пор, пока длина отрезка не станет меньше 2ε. Тогда координата середины отрезка есть значение корня с требуемой точностью ε.

Метод бисекции является простым и надёжным методом поиска простого корня уравнения *f (x) = 0* (простым называется корень *x = c* дифференци-

руемой функции *f (x)*, если *f (c) = 0* и *f ‘ (c) ≠ 0*). Скорость его сходимости невысока. Для достижения точности ε необходимо совершить

*N ≈ log2((b−a)/ε)* итераций. Это означает, что для получения каждых трёх верных десятичных знаков необходимо совершить около 10 итераций.

# Постановка задачи

Из программ-функций BISECT и Round из файла methods.cpp (файл заголовков metods.h), найти корень уравнения методом бисекции с заданной точностью Eps, исследовать зависимость числа итераций от точности Eps при изменении Eps от 0.1 до 0.000001, исследовать обусловленность метода (чувствительность к ошибкам в исходных данных).

Порядок выполнения:

1. Графически или аналитически отделить корень уравнения f(x)=0
2. Составить подпрограмму вычисления функции f(x).
3. Составить головную программу, содержащую обращение к подпрограмме *F*, *BISECT*, *Round* и индикацию результатов.
4. Провести вычисления по программе. Построить график зависимости числа итераций от *Eps*, сопоставить его с графиком по формуле выше.
5. Исследовать чувствительность метода к ошибкам в исходных данных. Ошибки в исходных данных моделировать с использованием программы *Round*, округляющей значения функции с заданной точностью *Delta*.

# Выполнение работы.

* 1. В начале работы графически был отделён корень уравнения *f* (*x*) = 0, т.е. найден отрезок [*a, b*], на котором функция удовлетворяет условиям теоремы Больцано-Коши.

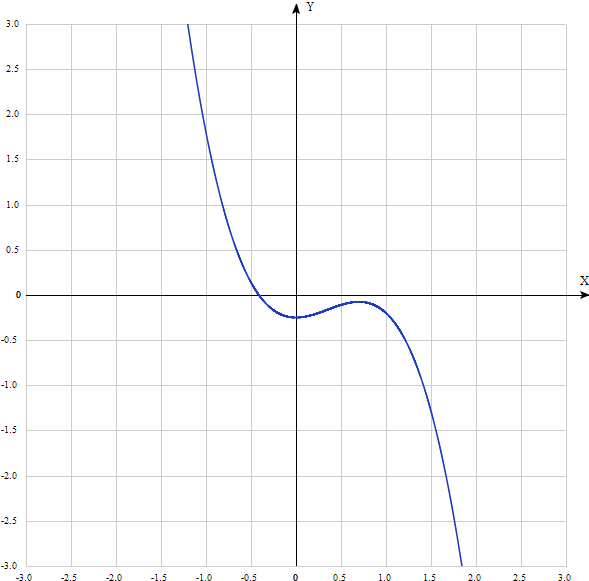


График функции

Заданная функция удовлетворяет условиям (непрерывна и монотонна)

применения метода бисекции на отрезке [-2*,* 3].

Согласно графику, изображённому на рисунке и аналитическому решению, *f (x) = 0* при **x ≈ -0.412105**

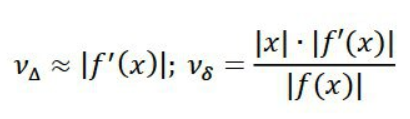


Графическое решение уравнения

Для проверки задачи были найдены абсолютное и относительное числа обусловленности данного уравнения, для этого была вычислена производная функции.

-3 x^2 + (2 x)/(x^2 + 4)^2 + 2 x

Под обусловленностью вычислительной задачи понимают чувствительность её решения к малым погрешностям входных данных. Для вычисления числа обусловленности функции от одной переменной были использованы формулы: *ν∆ ≈ 1.3811 и νδ ≈ 1,186646* *<* 10 задача хорошо обусловлена, т.е. устойчива к малым погрешностям входных данных.



* 1. Далее была разработана подпрограмма, производящая непосредственное вычисление функции *f* (*x*).
  2. Далее была составлена головная программа, содержащая обращение к подпрограмме F, BISECT, Round и выводящая необходимы результаты.

# Таблицы и графики

Создание таблицы зависимости числа итераций от точности вычисления корня:

* + - delta - точность задания исходных данных;
    - ***ξ*** - точность вычисления корня;
    - X – приближенное значение корня уравнения;
    - N ------ число итераций метода бисекции.

|  |  |  |  |
| --- | --- | --- | --- |
| № | ***ξ*** | Практ.  кол-во  операций | Теор.  кол-во  операций |
| 1 | 0.1 | 2 | ≈ 2.23 |
| 2 | 0.01 | 5 | ≈ 5.64 |
| 3 | 0.001 | 8 | ≈ 8.97 |
| 4 | 0.0001 | 12 | ≈ 12.29 |
| 5 | 0.00001 | 15 | ≈ 15.60 |
| 6 | 0.000001 | 18 | ≈ 18.93 |

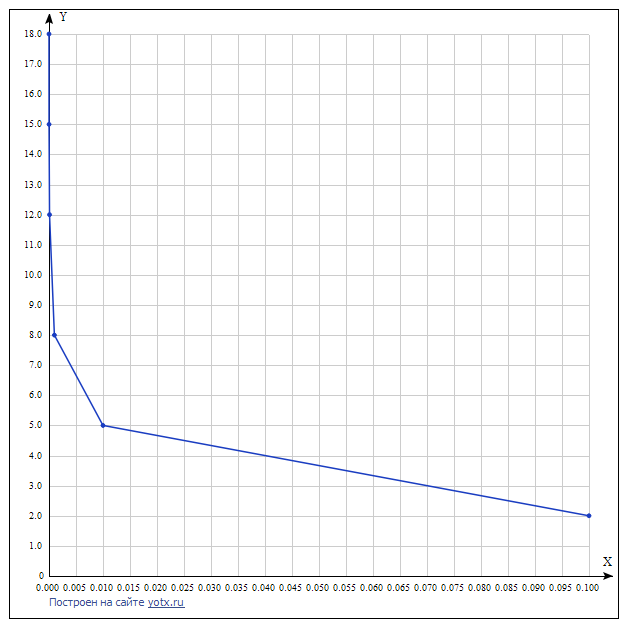
Зависимость числа итераций от Eps

Из полученных в ходе анализа таблицы, данных видно, что с уменьшением коэффициента eps, число итераций N и точность вычисления корня X растёт.

Также данную зависимость можно проследить на графике, изображённом на следующем рисунке.

Ось Х – Eps

Ось Y – Количество операций



Зависимость количества итераций от точности

Далее было произведено тестирования устойчивости метода к ошибкам входных данных, которые были смоделированы с использованием функции Round, округляющей значения функции с заданной точностью delta.

За обозначения левой и правой границ отрезка взяты a и b. Для проверки обусловленности задачи был использован корень x = -0.412105. При *ν*∆ >> 1, где ***ν*∆** - абсолютное число обусловленности, алгоритм плохо обусловлен.

*ν* = |*x* − *x*∗| *,*∆ = *Delta*

∆

*Delta*

На основе анализа таблицы можно сделать вывод, что при допущении небольшой неточности исходных данных и большой длины отрезка [*a, b*] вне

зависимости от значения eps может быть получен неверный результат ввиду

того, что функция на концах отрезков может иметь значение, близкое к 0, но при этом при высокой заданной точности даёт одинаковую обусловленность.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **a** | **b** | ***ξ*** | **Delta** | **x**∗ | ***ν∆*** | ***ν∆(+/-)*** |
| -2 | 3 | 0.1 | 0.1 | -0.28125 | 1.30855 | **+** |
| 0.01 | 0.01 | -0.417969 | 0.586375 | **+** |
| 1e-03 | 1e-03 | -0.411865 | 0.239766 | **+** |
| 1e-05 | 1e-05 | -0.412113 | 0.81897 | **+** |
| -5 | 5 | 0.1 | 0.1 | -0.46875 | 0.56645 | **+** |
| 0.01 | 0.01 | -0.410156 | 0.194875 | **+** |
| 1e-03 | 1e-03 | -0.411377 | 0.728047 | **+** |
| 1e-05 | 1e-05 | -0.412121 | 4.58191 | **-** |
| -20 | 20 | 0.1 | 0.1 | -0.46875 | 0.56645 | **+** |
| 0.01 | 0.01 | -0.410156 | 0.194875 | **+** |
| 1e-03 | 1e-03 | -0.411377 | 0.880635 | **+** |
| 1e-05 | 1e-05 | -0.412121 | 4.58191 | **-** |
| -50 | 50 | 0.1 | 0.1 | -0.410156 | 0.0194875 | **+** |
| 0.01 | 0.01 | -0.427246 | 1.51411 | **+** |
| 1e-03 | 1e-03 | -0.411224 | 0.880635 | **+** |
|  |  | 1e-05 | 1e-05 | -0.412109 | 4.389816 | **-** |

# Выводы

На основании анализа данных, полученных в ходе выполнения лабораторной работы, можно сделать вывод, что число итераций метода бисекции возрастает с ростом требуемой точности выходных данных.

Обусловленность задачи нахождения корня уравнения *f* (*x*) = 0 для

функции обратно пропорциональна значению производной в точке xи

точности вычисления корня, прямо пропорциональна точности задания

исходных данных. Т.е. чем меньше единица, деленная на значение

производной и delta, и чем больше eps, тем задача хуже обусловлена, и

наоборот. Задачу можно считать хорошо обусловленной, так как в данном

случае вычисленные методом бисекции корни не дают большой разницы с

корнем, вычисленным 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 == -1)

        return X;

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

double E = fabs(Eps) \* 2.0; //Требуется найти корень с точностью ε => деление пополам продолжается до тех пор, пока длина отрезка не станет меньше 2ε

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 точности" << endl;

return 0;

}

if(FLeft == 0.0)//Искомое значение в точке a

return Left;

if(FRight == 0.0)//Искомое значение в точке b

return Right;

    while((Right - Left) >= E){

        //cout << "[" << Left << "," << Right << "]" << endl;

X = 0.5\*(Right + Left);//Вычисление середины отрезка

Y = F(X, -1);//Значение функции в середине отрезка [a,b]

if(Y == 0.0){//Искомое значение в точке X

return X;

}

if(Y \* FLeft < 0.0){//Середина отрезка становится точкой b

Right = X;

}

else{

Left = X;//Середина отрезка становится точкой a

FLeft = Y;//Значение функции в точке X(a)

}

        N++;

}

return X;

}

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

double nt = log2((b - a)/eps);

    int n = 0;

    double value = BISECT(a, b, eps, n, Delta);

    double ob = (fabs(-0.412105 - value))/Delta;

    cout << "Значение: " << value << endl;

    cout << "Теор. число итераций: " << nt << endl;

    cout << "Практ. число итераций: " << n << endl;

    cout << "Число обусловленности: " << ob << endl;

    if(ob < 5)

        cout << "Хорошая обусл." << endl << endl;

    else

        cout << "Плохая обусл." << endl << endl;

}

int main(){

        Calculate(-2, 3, 0.1, 0.1);

        Calculate(-2, 3, 0.01, 0.01);

        Calculate(-2, 3, 0.001, 0.001);

        Calculate(-2, 3, 0.00001, 0.00001);

return 0;

}