**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра МО ЭВМ**

**ОТЧЕТ**

**по практической работе №3**

# **по дисциплине «Вычислительная математика»**

# **Тема: Метод бисекции**

| Студентка гр. 1304 |  | Виноградова М.О. |
| --- | --- | --- |
| Преподаватель |  | Попова Е.В. |

Санкт-Петербург

2022

**Вариант 5**

**Цель работы.**

Нахождение корня уравнения методом *бисекции* с заданной точностью *Eps*, исследование зависимости числа итераций от точности *Eps* при заданном изменении *Eps*, исследование обусловленности метода (чувствительность к ошибкам в исходных данных); исследование обусловленности метода без ограничения итераций заданной точностью *Eps*, когда рассчитывается членов последовательности приближенного значения корня.

**Теоретические положения**

Пусть заданная функция - непрерывна на отрезке , тогда существует точка *c*, в которой значение функции равно нулю, т.е. =0 (следствие из теоремы Больцано-Коши) [8]. В методе *бисекции* строят последовательность вложенных друг в друга отрезков, на концах которых функция имеет разные знаки. Каждый последующий отрезок получается делением пополам предыдущего. Процесс построения последовательности отрезков позволяет найти корень уравнения =0 с любой заданной точностью.

Опишем *(n-1)-*ю итерацию метода. На *(n-1)-*м шаге найден отрезок такой, что . Разделим данный отрезок пополам точкой и вычислим (рисунок 1).



Рисунок 1 – Деление отрезка пополам

Если =0, то - корень уравнения, при этом ошибка вычислений равна нулю, и задача хорошо обусловлена. Если 0, то из двух половин отрезка выбирается та, на концах которой функция имеет противоположные знаки, поскольку искомый корень лежит на этой половине. Таким образом, , , если ; или , , если .

Пусть во входных данных задачи задан *ε,* являющийся точностью искомого корня, тогда процесс деления пополам продолжается до тех пор, пока длина отрезка не станет меньше *2ε*. За значение корня функции с требуемой точностью *ε* принимается координата середины отрезка.

Другой подход вычислительного эксперимента заключается в отказе от ограничения *2ε*-окрестностью количества итераций. Строятся две последовательности приближений корня с двойной точностью и с наложенным шумом. Вычисляются значение числа обусловленности и абсолютная погрешность в точке , полученной без искажения функции при большом числе итераций . Проверяется неравенство и фиксируется при каких значениях итерационной переменной метод можно считать хорошо обусловленным.

Метод *бисекции* является несложным и надежным методом поиска простого корня уравнения (простым называется корень дифференцируемой функции , если и ).

Этот метод сходится для любых непрерывных функций , в том числе недифференцируемых. Скорость его сходимости невысока. Для достижения точности *ε* необходимо совершить порядка итераций. Это означает, что для получения каждых трех верных десятичных знаков необходимо совершить около 10 итераций.

Плюсы метода бисекции:

* Всегда сходится.
* Не требует гладкости.

Минусы метода:

* Сходимость относительно медленная.
* Не применяется для поиска кратных корней.

Для определения скорости сходимости обозначим середину *n*-го отрезка за точку . Погрешность приближения к корню определяется по формуле |. Из этой оценки видно, что метод *бисекции* сходится со скоростью геометрической прогрессии, знаменатель которой . По сравнению с другими методами метод *бисекции* сходится довольно медленно. Критерий окончания итерационного процесса определяется неравенством. При его выполнении, за приближенное значение корня принимают которое приближено к корню с точностью .

В случае, когда попадает в интервал неопределенности корня , знак вычисленного приближенного значения функции может быть неверным, и последующие итерации не имеют смысла. Однако этот метод следует признать очень надежным; он гарантирует точность приближения, примерно равную радиусу интервала неопределенности.

,

где x\* и y\* - приближённые входные данные и приближённое решение соответственно. Тогда величина называется абсолютным числом обусловленности. Если же установлено неравенство

между относительными ошибками данных и решения, то величину называют относительным числом обусловленности. Для плохо обусловленной задачи 𝜈 ≫ 1. Грубо говоря, если , где 𝜈 − относительное число обусловленности, то порядок N показывает число верных цифр, которое может быть утеряно в результате по сравнению с числом верных цифр входных данных. Ответ на вопрос о том, при каком значении 𝜈 задачу следует признать плохо обусловленной, зависит, с одной стороны, от предъявляемых требований 3 к точности решения и, с другой, – от уровня обеспечиваемой точности исходных данных. Например, если требуется найти решение с точностью 0.1%, а входная информация задается с точностью 0.02%, то уже значение 𝜈 = 10 сигнализирует о плохой обусловленности. Однако, при тех же требованиях к точности результата, гарантия, что исходные данные задаются с точностью не ниже 0.0001%, означает, что при задача хорошо обусловлена. Если рассматривать задачу вычисления корня уравнения 𝑦 = 𝑓(𝑥), то роль числа обусловленности будет играть величина

,

где – корень уравнения.

**Постановка задачи.**

1) Графически или аналитически отделить корень уравнения (найти отрезки [Left, Right], на которых функция . удовлетворяет условиям теоремы Коши).

2) Составить подпрограмму вычисления функции

3) Составить подпрограмму вычисления функции , для вычисления .

4) Составить головную программу, содержащую обращение к подпрограммам , BISECT, Round и вывод результатов.

5) Провести вычисления по программе. Построить график зависимости числа итераций от *Eps*.

6) Исследовать чувствительность метода к ошибкам в исходных данных. Ошибки в исходных данных моделировать с использованием программы Round, округляющей значения функции с заданной точностью Delta.

В коде прописывается вывод и (листинг 2).

Листинг 2 – Вывод значений и

std::cout << "V delta: " <<………………………. << "\n";

std::cout << "V delta max: " << eps / delta << "\n";

Во второй части работы модули программы, вычисляющие функции BISECT и F, модифицируются с новым условием прохождения цикла (листинг 2) и округления функции методом усечения.

1. Методом бисекции построить итерационную последовательность приближений к корню последовательности , где и задано в варианте. Вычисления проводить с двойной точностью.
2. Вычислить значение числа обусловленности по формуле , принимая за значение корня приближение полученное при числе итераций равное
3. Внести неточности в значения вычисляемой функции методом усечения точных значений функции до знака после запятой, заданного параметром . Получить приближенные значения итерационной последовательности .
4. Вычислить абсолютные величины разностей между значениями и приближенной итерационной последовательности где
5. Определить при каких значениях выполняется соотношение и задача является хорошо обусловленной.

Листинг 2 – Фрагмент модифицированной программы «BISECT»

while((\*N)<80)

{x = (left + right) / 2;

**Ход работы**

1. Корень уравнения c\*cth(x)-6 = 0 при с от 1 до 5.9 находится в (0.1, 2.5).

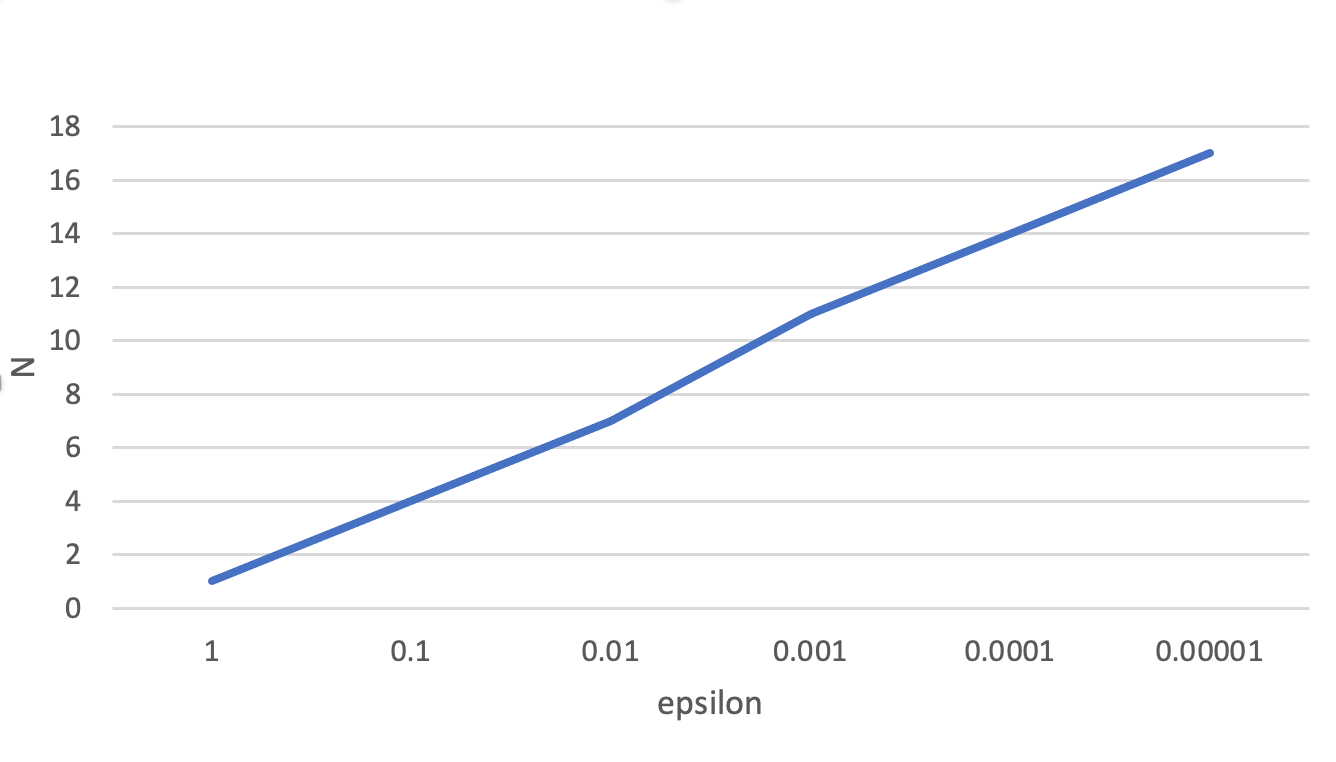
Область определения функции – (-inf, 0)(0,+inf)

Пример графика, где с = 1



График 1 f(x) при с=1

1. Функция F – принимает значения x и с типа double. Функция возвращает значение заданной функции ( c\*cth(x)-6 = 0 ) в точке x.
2. Функция Derivative – принимает значения x и с типа double. Функция возвращает значение производной заданной функции в точке x.
3. Код программы находится в приложении А
4. Вычисления и анализ

График 1. Зависимость количества итераций от eps (delta=0.01 c=3)

Вывод: чем меньше эпсилон, тем больше количество итераций.

Таблица 2. Вычисления по условиям из пункта 6

| c | x | delta | eps | N | ny | Ny max | conditioning |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 4 | 0.2 | 0.1 | 0.1 | 4 | 0.0231832 | 1 | good |
| 4 | 0.24 | 0.01 | 0.1 | 4 | 0.0159532 | 10 | good |
| 4 | 0.244 | 0.001 | 0.1 | 4 | 0.0159532 | 100 | good |
| 4 | 0.2438 | 0.0001 | 0.1 | 4 | 0.0159532 | 1000 | good |
| 4 | 0.24375 | 0.00001 | 0.1 | 4 | 0.0159532 | 10000 | good |

Вывод: обусловленность везде хорошая

Таблица 3. итерационная последовательность приближений к корню (eps=0.01 delta=0.01 с=1)

| | X0: |  | | --- | --- | | X1: |  | | X2: |  | | X3: |  | | X4: |  | | X5: |  | | X6: |  | | X7: |  | | X8: |  | | X9: |  | | X10: |  | | X11: |  | | X12: |  | | X13: |  | | X14: |  | | X15: |  | | X16: |  | | X17: |  | | X18: |  | | X19: |  | | X20: |  | | X21: |  | | X22: |  | | X23: |  | | X24: |  | | X25: |  | | X26: |  | | X27: |  | | X28: |  | | X29: |  | | X30: |  | | X31: |  | | X32: |  | | X33: |  | | X34: |  | | X35: |  | | X36: |  | | X37: |  | | X38: |  | | X39: |  | | X40: |  | | X41: |  | | X42: |  | | X43: |  | | X44: |  | | X45: |  | | X46: |  | | X47: |  | | X48: |  | | X49: |  | | X50: |  | | X51: |  | | X52: |  | | X53: |  | | X54: |  | | X55: |  | | X56: |  | | X57: |  | | X58: |  | | X59: |  | | X60: |  | | X61: |  | | X62: |  | | X63: |  | | X64: |  | | X65: |  | | X66: |  | | X67: |  | | X68: |  | | X69: |  | | X70: |  | | X71: |  | | X72: |  | | X73: |  | | X74: |  | | X75: |  | | X76: |  | | X77: |  | | X78: |  | | X79: |  | | X80: |  | | | 1.30 | | | --- | --- | | 0.75 | | | 0.475 | | | 0.3375 | | | 0.26875 | | | 0.234375 | | | 0.217188 | | | 0.208594 | | | 0.204297 | | | 0.202148 | | | 0.201074 | | | 0.200537 | | | 0.200269 | | | 0.200134 | | | 0.200067 | | | 0.200034 | | | 0.200017 | | | 0.200008 | | | 0.200004 | | | 0.200002 | | | 0.200001 | | | 0.200001 | | | 0.2 | | | 0.2 | | | 0.2 | | | 0.2 | | | 0.2 | | | 0.2 | | | 0.2 | | | 0.2 | | | 0.2 | | | 0.2 | | | 0.2 | | | 0.2 | | | 0.2 | | | 0.2 | | | 0.2 | | | 0.2 | | | 0.2 | | | 0.2 | | | 0.2 | | | 0.2 | | | 0.2 | | | 0.2 | | | 0.2 | | 0.2 | | 0.2 | | 0.2 | | 0.2 | | 0.2 | | 0.2 | | 0.2 | | 0.2 | | 0.2 | | 0.2 | | 0.2 | | 0.2 | | 0.2 | | 0.2 | | 0.2 | | 0.2 | | 0.2 | | 0.2 | | 0.2 | | 0.2 | | 0.2 | | 0.2 | | 0.2 | | 0.2 | | 0.2 | | 0.2 | | 0.2 | | 0.2 | | 0.2 | | 0.2 | | 0.2 | | 0.2 | | 0.2 | | 0.2 | | 0.2 | | 0.2 | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

=249,97367

Таблица 4. Вычисления по условиям из пунктов 3-5

|  | X | X\* | Δi | |  | | --- | | ν\_Δ | обусловленность |
| --- | --- | --- | --- | --- | --- | --- | --- |
| X0: | 1,300000 | 1,30 | 1,100000 | 0,01 | 249,97 | good |
| X1: | 0,750000 | 0,70 | 0,500000 |  |  | good |
| X2: | 0,475000 | 0,40 | 0,200000 |  |  | good |
| X3: | 0,337500 | 0,30 | 0,100000 |  |  | good |
| X4: | 0,268750 | 0,20 | 0,000000 |  |  | good |
| X5: | 0,234375 | 0,20 | 0,000000 |  |  | good |
| X6: | 0,217188 | 0,20 | 0,000000 |  |  | good |
| X7: | 0,208594 | 0,20 | 0,000000 |  |  | good |
| X8: | 0,204297 | 0,20 | 0,000000 |  |  | good |
| X9: | 0,202148 | 0,20 | 0,000000 |  |  | good |
| X10: | 0,201074 | 0,20 | 0,000000 |  |  | good |
| X11: | 0,200537 | 0,20 | 0,000000 |  |  | good |
| X12: | 0,200269 | 0,20 | 0,000000 |  |  | good |
| X13: | 0,200134 | 0,20 | 0,000000 |  |  | good |
| X14: | 0,200067 | 0,20 | 0,000000 |  |  | good |
| X15: | 0,200034 | 0,20 | 0,000000 |  |  | good |
| X16: | 0,200017 | 0,20 | 0,000000 |  |  | good |
| X17: | 0,200008 | 0,20 | 0,000000 |  |  | good |
| X18: | 0,200004 | 0,20 | 0,000000 |  |  | good |
| X19: | 0,200002 | 0,20 | 0,000000 |  |  | good |
| X20: | 0,200001 | 0,20 | 0,000000 |  |  | good |
| X21: | 0,200001 | 0,20 | 0,000000 |  |  | good |
| X22: | 0,200000 | 0,20 | 0,000000 |  |  | good |
| X23: | 0,200000 | 0,20 | 0,000000 |  |  | good |
| X24: | 0,200000 | 0,20 | 0,000000 |  |  | good |
| X25: | 0,200000 | 0,20 | 0,000000 |  |  | good |
| X26: | 0,200000 | 0,20 | 0,000000 |  |  | good |
| X27: | 0,200000 | 0,20 | 0,000000 |  |  | good |
| X28: | 0,200000 | 0,20 | 0,000000 |  |  | good |
| X29: | 0,200000 | 0,20 | 0,000000 |  |  | good |
| X30: | 0,200000 | 0,20 | 0,000000 |  |  | good |
| X31: | 0,200000 | 0,20 | 0,000000 |  |  | good |
| X32: | 0,200000 | 0,20 | 0,000000 |  |  | good |
| X33: | 0,200000 | 0,20 | 0,000000 |  |  | good |
| X34: | 0,200000 | 0,20 | 0,000000 |  |  | good |
| X35: | 0,200000 | 0,20 | 0,000000 |  |  | good |
| X36: | 0,200000 | 0,20 | 0,000000 |  |  | good |
| X37: | 0,200000 | 0,20 | 0,000000 |  |  | good |
| X38: | 0,200000 | 0,20 | 0,000000 |  |  | good |
| X39: | 0,200000 | 0,20 | 0,000000 |  |  | good |
| X40: | 0,200000 | 0,20 | 0,000000 |  |  | good |
| X41: | 0,200000 | 0,20 | 0,000000 |  |  | good |
| X42: | 0,200000 | 0,20 | 0,000000 |  |  | good |
| X43: | 0,200000 | 0,20 | 0,000000 |  |  | good |
| X44: | 0,200000 | 0,20 | 0,000000 |  |  | good |
| X45: | 0,200000 | 0,20 | 0,000000 |  |  | good |
| X46: | 0,200000 | 0,20 | 0,000000 |  |  | good |
| X47: | 0,200000 | 0,20 | 0,000000 |  |  | good |
| X48: | 0,200000 | 0,20 | 0,000000 |  |  | good |
| X49: | 0,200000 | 0,20 | 0,000000 |  |  | good |
| X50: | 0,200000 | 0,20 | 0,000000 |  |  | good |
| X51: | 0,200000 | 0,20 | 0,000000 |  |  | good |
| X52: | 0,200000 | 0,20 | 0,000000 |  |  | good |
| X53: | 0,200000 | 0,20 | 0,000000 |  |  | good |
| X54: | 0,200000 | 0,20 | 0,000000 |  |  | good |
| X55: | 0,200000 | 0,20 | 0,000000 |  |  | good |
| X56: | 0,200000 | 0,20 | 0,000000 |  |  | good |
| X57: | 0,200000 | 0,20 | 0,000000 |  |  | good |
| X58: | 0,200000 | 0,20 | 0,000000 |  |  | good |
| X59: | 0,200000 | 0,20 | 0,000000 |  |  | good |
| X60: | 0,200000 | 0,20 | 0,000000 |  |  | good |
| X61: | 0,200000 | 0,20 | 0,000000 |  |  | good |
| X62: | 0,200000 | 0,20 | 0,000000 |  |  | good |
| X63: | 0,200000 | 0,20 | 0,000000 |  |  | good |
| X64: | 0,200000 | 0,20 | 0,000000 |  |  | good |
| X65: | 0,200000 | 0,20 | 0,000000 |  |  | good |
| X66: | 0,200000 | 0,20 | 0,000000 |  |  | good |
| X67: | 0,200000 | 0,20 | 0,000000 |  |  | good |
| X68: | 0,200000 | 0,20 | 0,000000 |  |  | good |
| X69: | 0,200000 | 0,20 | 0,000000 |  |  | good |
| X70: | 0,200000 | 0,20 | 0,000000 |  |  | good |
| X71: | 0,200000 | 0,20 | 0,000000 |  |  | good |
| X72: | 0,200000 | 0,20 | 0,000000 |  |  | good |
| X73: | 0,200000 | 0,20 | 0,000000 |  |  | good |
| X74: | 0,200000 | 0,20 | 0,000000 |  |  | good |
| X75: | 0,200000 | 0,20 | 0,000000 |  |  | good |
| X76: | 0,200000 | 0,20 | 0,000000 |  |  | good |
| X77: | 0,200000 | 0,20 | 0,000000 |  |  | good |
| X78: | 0,200000 | 0,20 | 0,000000 |  |  | good |
| X79: | 0,200000 | 0,20 | 0,000000 |  |  | good |
| X80: | 0,200000 | 0,20 | 0,000000 |  |  | good |

Вывод по таблицам в файле Math3 лист 3 и лист 4:

Для метода Бисекции потребовалось меньше итераций, чем для метода хорд. Обусловленность для бисекции при каждом из трех возмущений становилась хорошей с 2 шага, для метода хорд, только при третьем возмущении с 5 шага обусловленность стала хорошей, при других двух обусловленность всегда хорошая.

Скорость сходимости метода хорд:

Если х корень уравнения f(x) = 0 находится на отрезке [a , b], первая и вторая производные непрерывны и сохраняют знак и f’’(b)\*f(b)>0 , то метод сходится со скоростью геометрической прогрессии.

График 2 зависимости N, с которого задача стала хорошо обусловленной от разряда, подвергшегося возмущению(метод хорд).

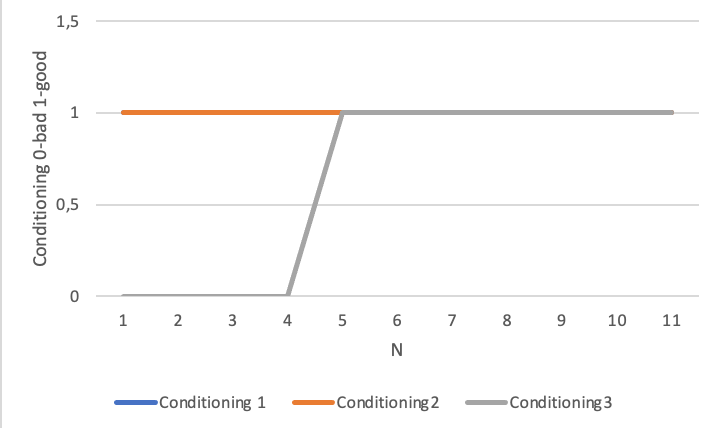
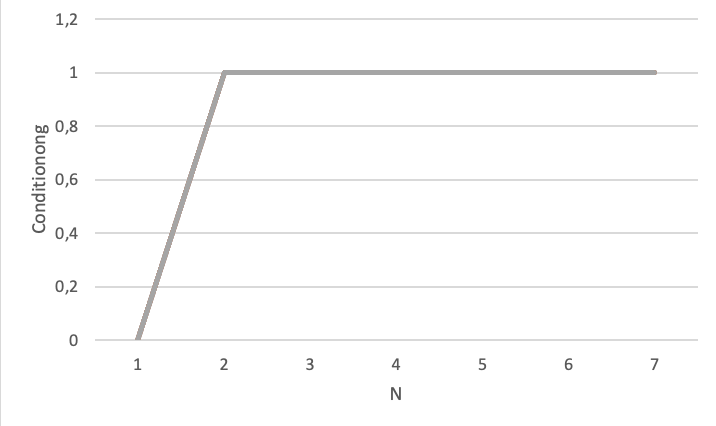
****

График 3 зависимости N, с которого задача стала хорошо обусловленной от разряда, подвергшегося возмущению (метод бисекции).

****

**Приложение А**

#include<iostream>  
#include<cmath>  
  
double Round\_Truncation(double Y) {  
 if (Y >= 0) {  
 return floor(Y);*//round down* } else {  
 return ceil(Y);*//round up* }  
}  
  
*//c\*cth(x)-6*double F(double x, double c = 1) {  
 double cot = 1 / tanh(x);  
 return cot \* c - 6;  
}  
  
double Derivative(double c, double x) {  
 double derivative = -c / (sinh(x) \* sinh(x));  
 return derivative;  
}  
  
std::string Conditioning(double c, double x, double eps, double delta, double &ny) {  
 double derivative = Derivative(c, x);  
 ny = fabs(1 / derivative);  
 if (ny <= (eps / delta)) {  
 return "ok";  
 } else {  
 return "not ok";  
 }  
}  
  
double BISECT(double Left, double Right, double Eps, int &N, double c) {  
  
  
 double E = fabs(Eps) \* 2.0;  
 double FLeft = F(Left, c);  
 double FRight = F(Right, c);  
 double X = (Left + Right) / 2.0;  
 double Y;  
 if (FLeft \* FRight > 0.0) {  
 puts("neverno zadan interval\n");  
  
 exit(1);  
 }  
 if (Eps <= 0.0) {  
 puts("neverno zadana tochnost\n");  
 exit(1);  
 }  
 N = 0;  
 if (FLeft == 0.0) return Left;  
 if (FRight == 0.0) return Right;  
 while ((Right - Left) >= E) {  
 X = 0.5 \* (Right + Left);  
 Y = F(X);  
 if (Y == 0.0) return (X);  
 if (Y \* FLeft < 0.0) Right = X;  
 else {  
 Left = X;  
 FLeft = Y;  
 }  
 *//std::cout<<"X"<<N<<": "<<Round\_Truncation(X\*1000)/1000<<" f(X):"<<Round\_Truncation(F(X)\*1000)/1000<<"\n";  
 //std::cout<<"X"<<N<<": "<<Round\_Truncation(X\*100)/100<<" f(X):"<<Round\_Truncation(F(X)\*100)/100<<"\n";* std::cout<<"X"<<N<<": "<<Round\_Truncation(X\*10000)/10000<<" f(X):"<<Round\_Truncation(F(X)\*10000)/10000<<"\n";  
 N++;  
 };  
 return X;  
}  
  
double Round(double X, double Delta) {  
 if (Delta <= 1E-9) {  
 puts("Неверно задана точность округления\n");  
  
 exit(1);  
 }  
 if (X > 0.0) return (Delta \* (long((X / Delta) + 0.5)));  
 else return (Delta \* (long((X / Delta) - 0.5)));  
}  
  
  
  
double HORDA(double Left, double Right, double Eps, int &N) {  
 double FLeft = F(Left);  
 double FRight = F(Right);  
 double X, Y;  
 if (FLeft \* FRight > 0.0) {  
 puts("Неверное задание интервала\n");  
 exit(1);  
 }  
 if (Eps <= 0.0) {  
 puts("Неверное задание точности\n");  
 exit(1);  
 }  
 N = 0;  
 if (FLeft == 0.0) {  
 return Left;  
 }  
 if (FRight == 0.0) {  
 return Right;  
 }  
 *//std::cout<<Left<<" "<<Right<<" "<<FLeft<<" "<<FRight<<"\n";* do {  
 X = Left - (Right - Left) \* FLeft / (FRight - FLeft);  
 Y = F(X);  
 if (Y == 0.0) {  
 return X;  
 }  
 if (Y \* FLeft < 0.0) {  
 Right = X;  
 FRight = Y;  
 } else {  
 Left = X;  
 FLeft = Y;  
 }  
 std::cout<<"X"<<N<<": "<<Round\_Truncation(X\*100)/100<<" F(X): "<<Round\_Truncation(Y\*100)/100<<"\n";  
 N++;  
 } while (fabs(Y) >= Eps);  
  
 return X;  
}  
  
double BISECT\_2(double Left, double Right, int \*N) {  
  
 double FLeft = F(Left);  
 double FRight = F(Right);  
 double X, Y;  
 if (FLeft \* FRight > 0.0) {  
 puts("Неверное задание интервала\n");  
 exit(1);  
 }  
  
 \*N = 0;  
 if (FLeft == 0.0) {  
 return Left;  
 }  
 if (FRight == 0.0) {  
 return Right;  
 }  
 *//std::cout<<Left<<" "<<Right<<" "<<FLeft<<" "<<FRight<<"\n";* double e;  
 while ((\*N) < 80) {  
 e = Left + Right / 2;  
 double Fe = F(e);  
 if (Fe \* F(Left) < 0) {  
 Right = e;  
 }  
 if (Fe \* F(Left) > 0) {  
 Left = e;  
 }  
 if (Fe == 0) {  
 return Round\_Truncation(e \* 10) / 10;  
 }  
 std::cout << "X" << \*N << ": " << e << " f(x):"<<F(e)<<"\n";  
 ++\*N;  
 }  
  
 return e;  
}  
  
int main() {  
  
 int N;  
 double e, d, c;  
 std::cin >> e;  
 std::cin >> d;  
 std::cin >> c;  
 int &N2 = N;  
 double ny;  
 *//Round(BISECT(0.1, 2.4, e, N2, c), d);* int N3 = 0;  
 std::cout<< 1/Derivative(3,0.168236);  
 *//BISECT\_2(0.1,2.4,&N3);  
 // std::cout << "x: " << Round(BISECT(0.1, 2.4, e, N2, c), d) << std::endl;  
 // std::cout << "f(x): " << F(Round(BISECT(0.1, 2.4, e, N2, c), d)) << std::endl;  
 // int N3 = 0;  
 //std::cout << BISECT\_2(0.1, 2.4, &N3);  
 //std::cout << "N" << N2 << "\n";  
 //std::cout << "f'(x): " << Derivative(c, BISECT(0.1, 2.5, e, N2, c)) << std::endl;  
 //std::cout << "N" << N2 << "\n";* HORDA(0.1, 2.5, e, N2);  
 *//std::cout << "N" << N2 << "\n";  
 //std::cout << "Conditioning: " << Conditioning(c, Round(BISECT(0.1, 2.5, e, N2, c), d), e, d, ny) << "\n";  
 //std::cout << "V delta: " << ny << "\n";  
 //std::cout << "V delta max: " << e / d << "\n";* return 0;  
}