МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №2

по дисциплине «Вычислительная математика»

Тема: Исследование обусловленности задачи нахождения корня уравнения на примере линейной функции

Студент гр. 0304 _____ Алексеев Р.В.

Преподаватель _____ Попова Е.В.

Санкт-Петербург

2021

Вариант 1

Цель работы.

Исследование обусловленности задачи нахождения корня уравнения на примере линейной функции.

Основные теоретические положения.

Под обусловленностью вычислительной задачи понимают чувствительность ее решения к малым погрешностям входных данных. Задачу называют хорошо обусловленной, если малым погрешностям входных данных отвечают малые погрешности решения, и плохо обусловленной, если возможны сильные изменения решения. Количественной мерой степени обусловленности вычислительной задачи является число обусловленности, которое можно интерпретировать как коэффициент возможного возрастания погрешностей в решении по отношению к вызвавшим их погрешностям входных данных. Пусть между абсолютными погрешностями входных данных x и решения y установлено неравенство:

$$\Delta(y^*) \le \nu_\Delta \cdot \Delta(x^*),$$

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

$$\delta(y^*) \le \nu_\delta \cdot \delta(x^*)$$

между относительными ошибками данных и решения, то величину ν_δ называют относительным числом обусловленности. Для плохо обусловленной задачи $\nu \gg 1$. Грубо говоря, если $\nu_\delta = 10^N$, 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 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 $\nu = 10$ 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 $\nu = 10^3$ задача хорошо обусловлена. Если рассматривать задачу вычисления корня уравнения $y = f(x)$, то роль числа обусловленности будет играть величина

$$\nu_\Delta = \frac{1}{|f'(x_0)|},$$

где x_0 – корень уравнения.

Постановка задачи.

d = 26

Используя программы-функции BISECT и Round, исследовать обусловленность задачи нахождения корня уравнения $f(x) = 0$ для линейной функции $f(x) = c(x - d)$. Значения функции $f(x)$ следует вычислить приближенно с точностью Δ, варьируемой в пределах от 0.1 до 0.000001. Порядок выполнения работы следующий:

1) Отделение корня уравнения $f(x) = 0$.

2) Составление подпрограммы вычисления функции $f(x) = c(x - d)$ для параметров c и d вводимых с клавиатуры.

3) Составление головной программы, вычисляющей корень уравнения с заданной точностью ε, и содержащую обращение к подпрограмме F, программам-функциям BISECT, Round и представление результатов.

4) Проведение вычислений по программе, варьируя значения параметров.

5) Анализ результатов.

1. Параметр c варьируется от 0.01 до 1000. Параметры eps и delta постоянны и равны значению 0.01.

2. Параметр c постоянен и равен 1, eps постоянен и равен 0.01, delta варьируется от 0.00001 до 0.1.

3. Параметр c постоянен и равен 10, delta постоянна и равна 0.01, eps варьируется от 0.000001 до 10.

4. Параметр c постоянен и равен 10, delta и eps одновременно варьируются от 0.000001 до 1 (а можно сдвинуть влево). Построить график зависимости eps от количества итераций.

5. Параметр eps постоянен и равен 0.01, c и delta варьируются независимым друг от друга образом.

Выполнение работы.

1. Функция $f(x) = c(x-d)$ линейная, т. е. имеет один корень, равный, по условию, d = 26. Корень уравнения всегда находится внутри некоторого интервала (d - ε1; d + ε2), где ε1, ε2 > 0. В данной работе выбраны значения ε1 = 7 и ε2 = 3. Границы отрезка будут передаваться в функцию *BISECT*.

2. Напишем функцию

```
double F(double x){
  return c*(x-d);
}
```

которая будет вычислять значение функции $f(x) = c(x-d)$.

3. При постоянных eps и delta, равным 0.01, и параметре c, варьирующися от 0.01 до 1000 значение абсолютной обусловленности имело вид $\frac{1}{c}$, т. к. $f'(x) = c$, максимальное же значение абсолютного числа обусловленности вычислялось по формуле $\frac{eps}{delta}$ и оставалось равном 1. Поэтому при значениях $c \leq 1$ задача обусловлена плоха, а при $c > 1$ — хорошо. Данный вывод подтвержадется результатми работы программы для данных параметров, представленных на рис. 1.

c	eps	delta	x	nu	nu_max	iter	Вывод
0.010000	0.010000	0.010000	26.010000	100.000000	1.000000	9	Плохо
0.100000	0.010000	0.010000	26.010000	10.000000	1.000000	9	Плохо
1.000000	0.010000	0.010000	26.010000	1.000000	1.000000	9	Плохо
10.000000	0.010000	0.010000	26.010000	0.100000	1.000000	9	Хорошо
100.000000	0.010000	0.010000	26.010000	0.010000	1.000000	9	Хорошо
1000.000000	0.010000	0.010000	26.010000	0.001000	1.000000	9	Хорошо

Рисунок 1 — eps = 0.01, delta = 0.01, c варьируется от 0.01 до 1000.

4. При постоянном значении параметров c = 1 и eps = 0.01, и при варьирующимся от 0.00001 до 0.1 значении параметра delta значение абсолютной обсуловленности остается равным 1, а значение абсолютного числа обусловленности уменьшается с увеличением delta. Низкая точность исходных данных (при большом значении delta) приводит к тому, что нахождение значения функции с достаточной функцией становится невозможным. Данный вывод подтверждается результатом работы программы, представленном на рис. 2.

c	eps	delta	x	nu	nu_max	iter	Вывод
1.000000	0.010000	0.000010	26.011720	1.000000	1000.000000	9	Хорошо
1.000000	0.010000	0.000100	26.011700	1.000000	100.000000	9	Хорошо
1.000000	0.010000	0.001000	26.012000	1.000000	10.000000	9	Хорошо
1.000000	0.010000	0.010000	26.010000	1.000000	1.000000	9	Плохо
1.000000	0.010000	0.100000	26.000000	1.000000	0.100000	9	Плохо

Рисунок 2 — c = 1, eps = 0.01, delta варьируется от 0.00001 до 0.1.

5. При постоянном значении параметров c = 10 и delta = 0.01, и при eps варьирующимся от 0.000001 до 10 значение абсолютного числа обсуловленности постоянно и равно 0.1, значение максимального абсолютного числа обусловленности увеличиватеся с увеличением значения eps. Следовательно чем выше точность результатов задачи необходима, тем выше должна быть точность входных данных. Данный вывод подтверждается результатами работы программы, представленными на рис. 3.

c	eps	delta	x	nu	nu_max	iter	Вывод
10.000000	0.000001	0.010000	26.000000	0.100000	0.000100	23	Плохо
10.000000	0.000010	0.010000	26.000000	0.100000	0.001000	19	Плохо
10.000000	0.000100	0.010000	26.000000	0.100000	0.010000	16	Плохо
10.000000	0.001000	0.010000	26.000000	0.100000	0.100000	13	Плохо
10.000000	0.010000	0.010000	26.010000	0.100000	1.000000	9	Хорошо
10.000000	0.100000	0.010000	26.030000	0.100000	10.000000	6	Хорошо
10.000000	1.000000	0.010000	25.250000	0.100000	100.000000	3	Хорошо
10.000000	10.000000	0.010000	24.000000	0.100000	1000.000000	0	Хорошо

Рисунок 3 — c = 10, delta = 0.01, eps ворьируется от 0.000001 до 10.

6. При постоянном значении параметра c = 10 и при значениях delta и eps одновременно варьирующихся от 0.000001 до 1 значение абсолютного числа обусловленности постоянно и равно 0.1, значение максимального абсолютного числа обсуловленности также постоянно и равно 1. Следовательно при одновременном изменении точности исходных данных и точности результата допустимое число обусловленности не изменяется. Данный вывод подтверждается результатами работы программы, представленными на рис. 4.

c	eps	delta	x	nu	nu_max	iter	Вывод
10.000000	0.000001	0.000001	25.999999	0.100000	1.000000	23	Хорошо
10.000000	0.000010	0.000010	25.999990	0.100000	1.000000	19	Хорошо
10.000000	0.000100	0.000100	26.000000	0.100000	1.000000	16	Хорошо
10.000000	0.001000	0.001000	26.001000	0.100000	1.000000	13	Хорошо
10.000000	0.010000	0.010000	26.010000	0.100000	1.000000	9	Хорошо
10.000000	0.100000	0.100000	26.000000	0.100000	1.000000	6	Хорошо
10.000000	1.000000	1.000000	25.000000	0.100000	1.000000	3	Хорошо

Рисунок 4 — c = 10, delta и eps одновременно варьируются от 0.000001 до 1.

На основании результатов работы программы посторен график зависимости N от eps, где N — количество итераций, необходимых для вычисления уравнения с заданной точностью. График представлен на рис. 5.

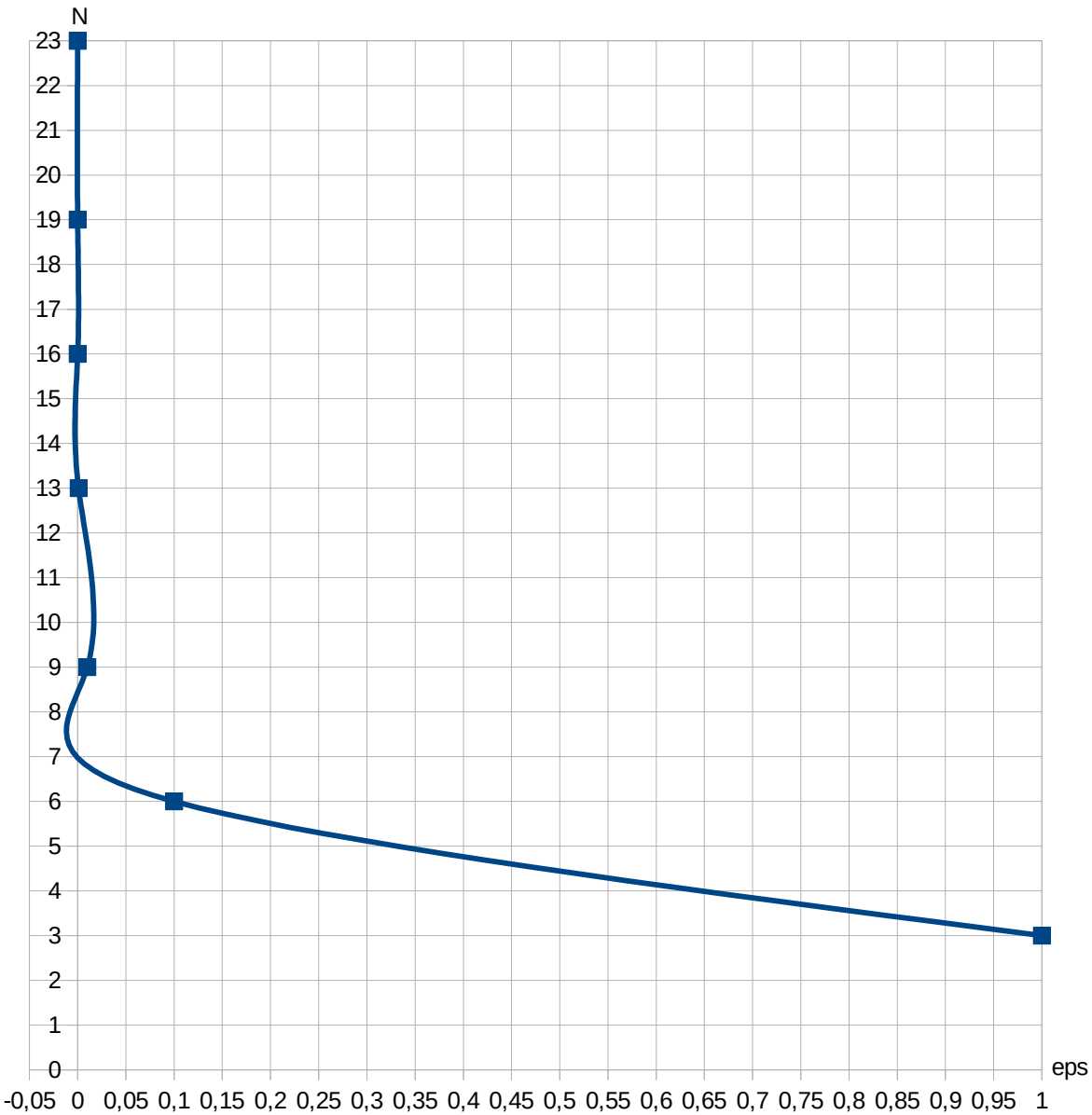

Рисунок 5 — график зависимости eps от количества итераций N.

На графике видна обратная логарифмическая зависимость N от eps, т. е. $N \sim \log(\frac{1}{eps})$. Это обусловлено тем, что алгоритм бисекции имеет логарифмическую сложность, а входной параметр имеет обратную зависимость от eps, т. к. с увеличением eps увеличивается диапозон значений, следовательно происходит меньше итераций.

7. При постоянном значении параметра eps = 0.01 и независимо друг от друга варьирующихся значениях c и delta значение абсолютного числа обусловеленности увеличивается с увеличением значения c, а значением максимального абсолютного числа обусловеленности уменьшается при уменьшении значения delta. Данный вывод подтвержадется резутатами работы программы, представленными на рис. 6.

c	eps	delta	x	nu	nu_max	iter	Вывод
0.010000	0.010000	0.000100	26.011700	100.000000	100.000000	9	Плохо
0.010000	0.010000	0.001000	26.012000	100.000000	10.000000	9	Плохо
0.010000	0.010000	0.010000	26.010000	100.000000	1.000000	9	Плохо
0.010000	0.010000	0.100000	26.000000	100.000000	0.100000	9	Плохо
0.100000	0.010000	0.000100	26.011700	10.000000	100.000000	9	Хорошо
0.100000	0.010000	0.001000	26.012000	10.000000	10.000000	9	Плохо
0.100000	0.010000	0.010000	26.010000	10.000000	1.000000	9	Плохо
0.100000	0.010000	0.100000	26.000000	10.000000	0.100000	9	Плохо
1.000000	0.010000	0.000100	26.011700	1.000000	100.000000	9	Хорошо
1.000000	0.010000	0.001000	26.012000	1.000000	10.000000	9	Хорошо
1.000000	0.010000	0.010000	26.010000	1.000000	1.000000	9	Плохо
1.000000	0.010000	0.100000	26.000000	1.000000	0.100000	9	Плохо
10.000000	0.010000	0.000100	26.011700	0.100000	100.000000	9	Хорошо
10.000000	0.010000	0.001000	26.012000	0.100000	10.000000	9	Хорошо
10.000000	0.010000	0.010000	26.010000	0.100000	1.000000	9	Хорошо
10.000000	0.010000	0.100000	26.000000	0.100000	0.100000	9	Плохо
100.000000	0.010000	0.000100	26.011700	0.010000	100.000000	9	Хорошо
100.000000	0.010000	0.001000	26.012000	0.010000	10.000000	9	Хорошо
100.000000	0.010000	0.010000	26.010000	0.010000	1.000000	9	Хорошо
100.000000	0.010000	0.100000	26.000000	0.010000	0.100000	9	Хорошо
1000.000000	0.010000	0.000100	26.011700	0.001000	100.000000	9	Хорошо
1000.000000	0.010000	0.001000	26.012000	0.001000	10.000000	9	Хорошо
1000.000000	0.010000	0.010000	26.010000	0.001000	1.000000	9	Хорошо
1000.000000	0.010000	0.100000	26.000000	0.001000	0.100000	9	Хорошо

Рисунок 6 — eps = 0.01, c и delta независимо варьируются.

Разработанный программный код см. в приложении А.

Выводы.

Была исследованы обусловеленность задачи нахождения корня уравнения. В ходе работы было подтверждено, что обусловленность задачи зависит от точности исходных данных, желаемой точности результата решения, от выбранной функции, а именно от её производной. Если производная функции влияет на относительное число обсуловленности, то точность исходных данных и результата решения влияет на максимально допустимое значение числа обусловленности, т. е. все эти параметры влияют на обусловеленность задачи.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ НА ЯЗЫКЕ C++

```
#include <cstdio>
#include <cmath>

#define d 26

double c;

double F(double x){
  return c*(x-d);
}

double BISECT(double Left, double Right, double Eps, int &N){
  double E = fabs(Eps)*2.0;
  double FLeft = F(Left);
  double FRight = F(Right);
  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;
      }
      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)));
}

int main(){
  double eps = 0.01;
  double delta = 0.01;
  int n = 0;


puts("-------------------------------------------------------------------
-------------------------------------------------");
  printf("%6s\t%14s\t%14s\t%14s\t%14s\t%14s\t%10s\t%10s\n",   "c",
"eps", "delta", "x", "nu", "nu_max", "iter", "Вывод");

puts("-------------------------------------------------------------------
-------------------------------------------------");
  for(c = 0.01; c <= 1000; c *= 10){
      double x = Round(BISECT(d-7, d+3, eps, n), delta);
      printf("%6f\t%6f\t%6f\t%6f\t%6f\t%6f\t%d\t", c, eps, delta,
x, 1/c, eps/delta, n);
      if(1/c < eps/delta)
          printf("%6s\n", "Хорошо");
      else
          printf("%6s\n", "Плохо");
  }

puts("-------------------------------------------------------------------
-------------------------------------------------");

  c = 1;
  eps = 0.01;


puts("-------------------------------------------------------------------
-------------------------------------------------");
  printf("%6s\t%14s\t%14s\t%14s\t%14s\t%14s\t%10s\t%10s\n",   "c",
"eps", "delta", "x", "nu", "nu_max", "iter", "Вывод");

puts("-------------------------------------------------------------------
-------------------------------------------------");
  for(delta = 0.00001; delta <= 0.1; delta *= 10){
      double x = Round(BISECT(d-7, d+3, eps, n), delta);
      printf("%6f\t%6f\t%6f\t%6f\t%6f\t%6f\t%d\t", c, eps, delta,
x, 1/c, eps/delta, n);
      if(1/c < eps/delta)
```

```
            printf("%6s\n", "Хорошо");
        else
            printf("%6s\n", "Плохо");
    }

puts("-------------------------------------------------------------------
-------------------------------------------------");

    c = 10;
    delta = 0.01;


puts("-------------------------------------------------------------------
-------------------------------------------------");
    printf("%6s\t%14s\t%14s\t%14s\t%14s\t%14s\t%10s\t%10s\n",   "c",
"eps", "delta", "x", "nu", "nu_max", "iter", "Вывод");

puts("-------------------------------------------------------------------
-------------------------------------------------");
    for(eps = 0.000001; eps <= 10; eps *= 10){
        double x = Round(BISECT(d-7, d+3, eps, n), delta);
        printf("%6f\t%6f\t%6f\t%6f\t%6f\t%6f\t%d\t", c, eps, delta,
x, 1/c, eps/delta, n);
        if(1/c < eps/delta)
            printf("%6s\n", "Хорошо");
        else
            printf("%6s\n", "Плохо");
    }

puts("-------------------------------------------------------------------
-------------------------------------------------");

    c = 10;


puts("-------------------------------------------------------------------
-------------------------------------------------");
    printf("%6s\t%14s\t%14s\t%14s\t%14s\t%14s\t%10s\t%10s\n",   "c",
"eps", "delta", "x", "nu", "nu_max", "iter", "Вывод");

puts("-------------------------------------------------------------------
-------------------------------------------------");
    for(eps = 0.000001, delta = 0.000001; eps <= 1 && delta <= 1;
eps *= 10, delta *= 10){
        double x = Round(BISECT(d-7, d+3, eps, n), delta);
        printf("%6f\t%6f\t%6f\t%6f\t%6f\t%6f\t%d\t", c, eps, delta,
x, 1/c, eps/delta, n);
        if(1/c < eps/delta)
            printf("%6s\n", "Хорошо");
        else
            printf("%6s\n", "Плохо");
    }

puts("-------------------------------------------------------------------
-------------------------------------------------");
```

```
        eps = 0.01;


puts("-------------------------------------------------------------------
-------------------------------------------------");
        printf("%6s\t%14s\t%14s\t%14s\t%14s\t%14s\t%10s\t%10s\n",   "c",
"eps", "delta", "x", "nu", "nu_max", "iter", "Вывод");

puts("-------------------------------------------------------------------
-------------------------------------------------");
        for(c = 0.01; c <= 1000; c *= 10){
            for(delta = 0.0001; delta <= 0.1; delta *= 10){
                double x = Round(BISECT(d-7, d+3, eps, n), delta);
                printf("%6f\t%6f\t%6f\t%6f\t%6f\t%6f\t%d\t",   c,   eps,
delta, x, 1/c, eps/delta, n);
                if(1/c < eps/delta)
                    printf("%6s\n", "Хорошо");
                else
                    printf("%6s\n", "Плохо");
            }
        }

puts("-------------------------------------------------------------------
-------------------------------------------------");

        return 0;
      }
```