Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное
образовательное учреждение высшего образования
Санкт-Петербургский университет аэрокосмического приборостроения

КАФЕДРА № 2

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ
ПРЕПОДАВАТЕЛЬ

доц., канд. техн. наук	Зачтено 28.02.2023	С.Л. Козенко
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ПРАКТИЧЕСКОЙ РАБОТЕ

Нелинейные уравнения

по дисциплине: ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА

РАБОТУ ВЫПОЛНИЛ
СТУДЕНТ ГР. № 4134к

	подпись, дата	инициалы, фамилия

Санкт – Петербург, 2023

Цель работы:

а) освоение методов решения нелинейных уравнений;
б) совершенствование навыков по алгоритмизации и программированию вычислительных задач.

Постановка задачи:

19.	$\frac{a}{x} + be^{cx} = 0$	Хорд $\varepsilon = 5 \cdot 10^{-4}$	$a = 2.37;\ b = -0.99;$ $c = 0.56$

Математическая часть:

Общее

Уравнением называется равенство справедливое при некоторых значениях $x=x^*$, называемыми корнями этого уравнения или нулями функции (x). Решение уравнения заключается в определении его корней. Среди корней x^* могут быть и комплексные, однако в данной работе вычисляются только действительные корни.

Вычисление каждого из действительных корней складывается из двух этапов:

1) отделение корня, т.е. нахождение возможно малого интервала a, b, в пределах которого находится один и только один корень x^* уравнения;

2) уточнение значения корня, т.е. вычисление с заданной степенью точности.

При использовании рассматриваемых ниже методов решения уравнения (2.1) к функции (x) на интервале a,b предъявляются следующие требования:

a) функция (x) непрерывна и дважды дифференцируема (т.е. существует первая и вторая производные);

b) первая производная $'(x)$ непрерывна, сохраняет знак и не обращается в нуль;

c) вторая производная $''(x)$ непрерывна и сохраняет знак.

Отделение корней может производиться аналитическим или графическим способами. Аналитический способ основывается на теореме Коши, утверждающей, что для непрерывной функции (x) (первое требование “a”), принимающей на концах интервала a, b разные знаки, т.е. $(a)\ (b)<0$, уравнение (2.1) имеет внутри этого интервала хотя бы один корень (рис. 1). Если к этому добавить второе требование “b”, означающее монотонность функции (x), то этот корень оказывается единственным.

В этих условиях отделение корня сводится к вычислению значений функции (x) для последовательности точек $_1, _2, \ldots, _n$ и сопоставлению знаков $(_k)$, $(_{k+1})$ в соседних точках $_k$ и $_{k+1}$. Каждый интервал $_k, _{k+1}$, для которого $(_k)\ (_{k+1})<0$, содержит, по крайней мере, один корень уравнения. Этот корень является единственным, если на этом интервале выполняется второе требование “b”. В противном случае следует интервал $_k, _{k+1}$ разделить на меньшие интервалы, повторяя для каждого из них указанные действия.

При использовании графического способа уравнение (2.1) можно также представить в виде

$$_1(x) = _2(x) \qquad (2.2)$$

и построить графики функций $y= _1(x)$ и $y= _2(x)$. Абсцисса точки пересечения этих графиков дает приближенное значение x^0 корня x уравнения

$$(x) = _1(x) - _2(x) = 0.$$

Представление уравнения (2.1) в форме (2.2) не является, естественно, однозначным и его следует подбирать так, чтобы построение графиков было возможно простым.

Из того же чертежа следует определить и тот интервал a, b, в пределах которого данный корень является единственным (если это необходимо для выбранного метода последующего уточнения значения корня x^0);

Метод хорд

Пусть определен интервал a, b, в котором лежит один корень x уравнения (2.1) $(x)=0$.

Учитывая, что (a) $(b)<0$, определяем первое приближение как точку пересечения с осью абсцисс хорды A_0B_0, соединяющей точки $A_0[a, (a)]$ и $B_0[b, (b)]$ (рис. 2.6,а).

Для нахождения последующего приближения вычислим значение (x_1) и сопоставим со значениями (a) и (b). Выберем тот из интервалов a,x_1 или x_1,b, на концах которого функция (x) имеет разные знаки (именно внутри этого интервала лежит искомый корень x). Применим предыдущий прием к этому интервалу, получая последующее приближение – точку x_2.

Заметим, что для случая, изображенного на рис. 9а, производные $'(x)$ и $''(x)$ сохраняют положительный знак ($'(x)>0$, $''(x)>0$; $'(x)$ $''(x)$ 0) и все приближения $x_1,x_2,$ образуют возрастающую последовательность, ограниченную значением x=x. Следовательно, $\lim_{n\to\infty} x_n = x^*,$ и при этом в любом из приближений соответствующая хорда проходит через начальную точку $B_0[b, (b)]$.

Для получения формулы, определяющей последующие приближения, рассмотрим переход от x_n и x_{n+1}. В этом случае уравнение хорды B_nB_0 как прямой, проходящей через точки B_n,B_0, имеет вид

$$\frac{y - f(x_n)}{f(b) - f(x_n)} = \frac{x - x_n}{b - x_n}.$$

Если для определения x_{n+1} положить $y(x_{n+1})=0$, то получим

$$x_{n+1} = x_n - f(x_n)\frac{b - x_n}{f(b) - f(x_n)} \qquad (n=0,1,2,\ldots) \qquad (2.15)$$

Согласно требованиям “a”,”b”,”c” (см. п.2.1), наложенным на функцию (x) для оценки погрешностей вычислений используется неравенство

$$\left|x_{n+1} - x^*\right| \le \frac{M - m}{m}\left|x_{n+1} - x_n\right|, \quad \text{где} \quad 0 < m \le |'(x)| \le M < 1.$$

Если при этом $M \quad 2m$, то $| x_{n+1}-x^{\tilde{}}| \quad | x_{n+1}-x_n|$, и для заданной погрешности ε вычисления прекращаются при $| x_{n+1}-x_n| \quad \varepsilon$ (как это имело место и для методов последовательных приближений и метода касательных).

При выполнении упомянутых требований *(“a”,”b”,”c”)* возможны и иные картины построений для метода хорд, определяемые сочетаниями знаков производных *'(x)* и *"(x).*

Рис. 2.6,а соответствует рассмотренному уже случаю *'(x)>0, "(x)>0* (функция *(x)* монотонно возрастает и выпукла вниз). Случай *'(x), "(x)◦*0* (рис. 2.6,б) приводит к аналогичным построениям, и последовательность $x_1, x_2,$ оказывается так же возрастающей.

Однако в случаях *'(x)>0, "(x)◦*0* (рис. 9,в) и *'(x) ◦*0, "(x)>0* (рис. 2.6,г) после определения каждого x_n различными оказываются знаки значений функций *(a)* и (x_n) (а не (x_n) и *(b),* как ранее). Поэтому “неподвижной” для всех хорд оказывается точка $A_0[a, (a)]$ (а не $B_0[b, (b)]$). В результате расчетными являются формулы

$$x_{n+1} = x_n - f(x_n)\frac{x_n - a}{f(x_n) - f(a)} \quad (n=0,1,2,\ ), \qquad (2.16)$$

а последовательность $x_1, x_2,$ оказывается убывающей.

Таким образом, если *'(x) "(x)◦*0*, то следует использовать формулы (2.15), выбирая за начальное значение $x_0=a$, если же *'(x) "(x)◦*0, то используются формулы (2.16) и начальным является $x_0=b$.

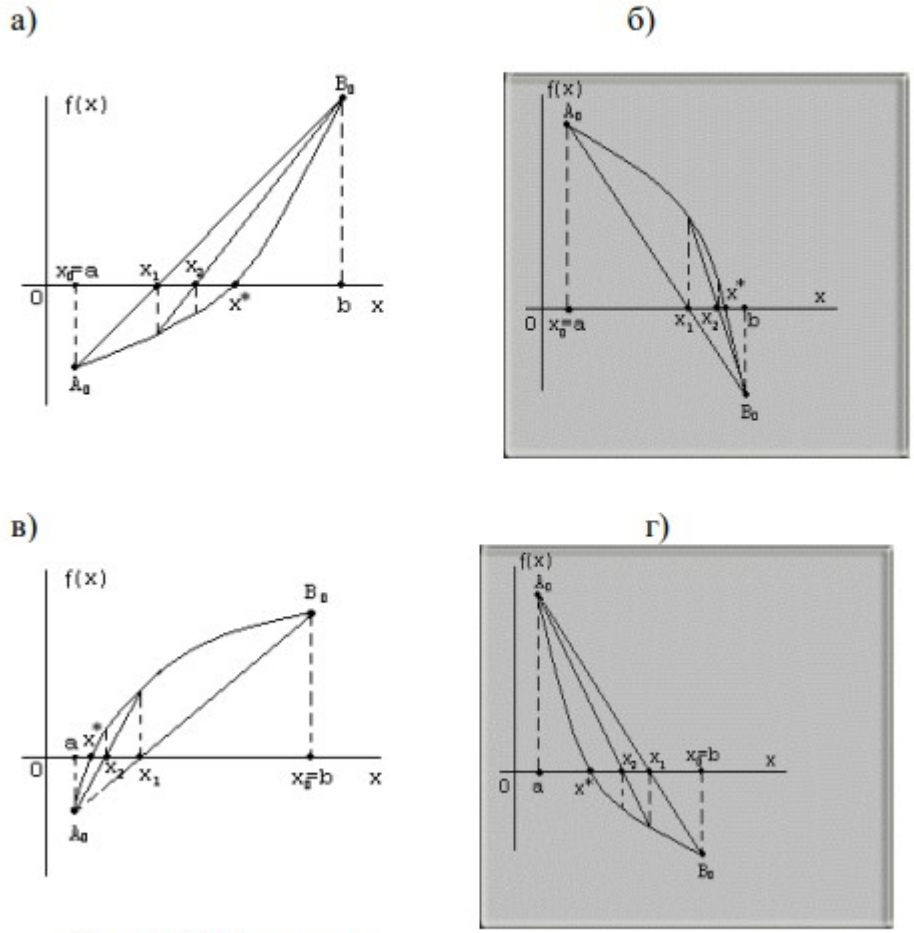


Рис. 2.6. Метод хорд

Аналитические расчёты:

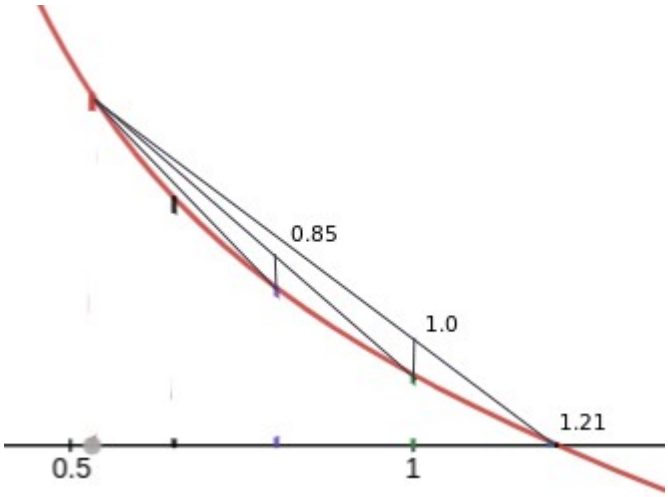
F(0.1)=22.653; F(10)=-267.485
Поскольку F(0.1)*F(10)<0 (т.е. значения функции на его концах имеют противоположные знаки), то корень лежит в пределах [0.1;10].
Вычисляем значения функций в точке a = 0.1
f(0.1) = 22.653
f''(0.1) = 4739.672
Поскольку f(a)•f''(a) > 0, то x_0 = a = 0.1
Остальные расчеты сведем в таблицу.

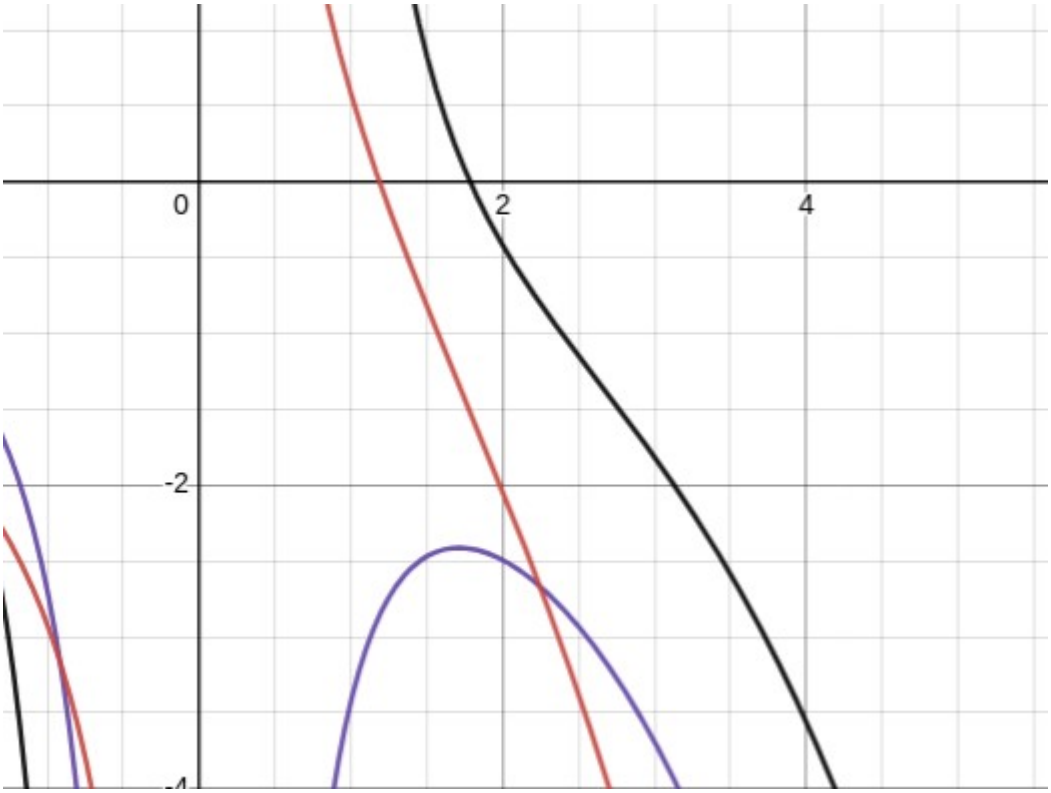
N	x	F(x)	h = F(x)*(x-a)/(f(x)-f(a))
1	10	-267.4851	0.873
2	0.873	1.1008	0.9124
3	0.9124	0.9472	0.9479
4	0.9479	0.817	0.9796

5	0.9796	0.7058	1.0079
6	1.0079	0.6106	1.0331
7	1.0331	0.5286	1.0553
8	1.0553	0.458	1.0751
9	1.0751	0.397	1.0924
10	1.0924	0.3442	1.1078
11	1.1078	0.2985	1.1212
12	1.1212	0.2589	1.133
13	1.133	0.2245	1.1434
14	1.1434	0.1948	1.1524
15	1.1524	0.169	1.1603
16	1.1603	0.1466	1.1672
17	1.1672	0.1271	1.1733
18	1.1733	0.1103	1.1785
19	1.1785	0.09566	1.1831
20	1.1831	0.08297	1.1871
21	1.1871	0.07197	1.1905
22	1.1905	0.06242	1.1935
23	1.1935	0.05414	1.1962
24	1.1962	0.04695	1.1984
25	1.1984	0.04072	1.2004
26	1.2004	0.03532	1.2021
27	1.2021	0.03063	1.2036
28	1.2036	0.02656	1.2049
29	1.2049	0.02303	1.206

30	1.206	0.01997	1.207
31	1.207	0.01732	1.2079
32	1.2079	0.01502	1.2086
33	1.2086	0.01303	1.2092
34	1.2092	0.0113	1.2098
35	1.2098	0.0098	1.2103

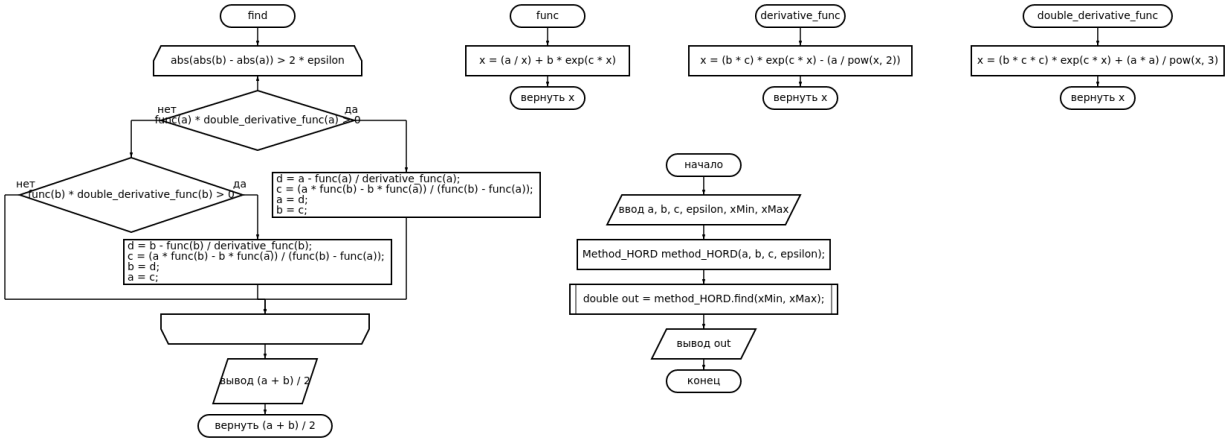
Ответ: x = 1.21-(1.21) = 1.210268911944; F(x) = 0.00849





из приведённых графиков условие коши выполнятся для выбранного интервала где находится корень

Схема алгоритма:



Текст программы:

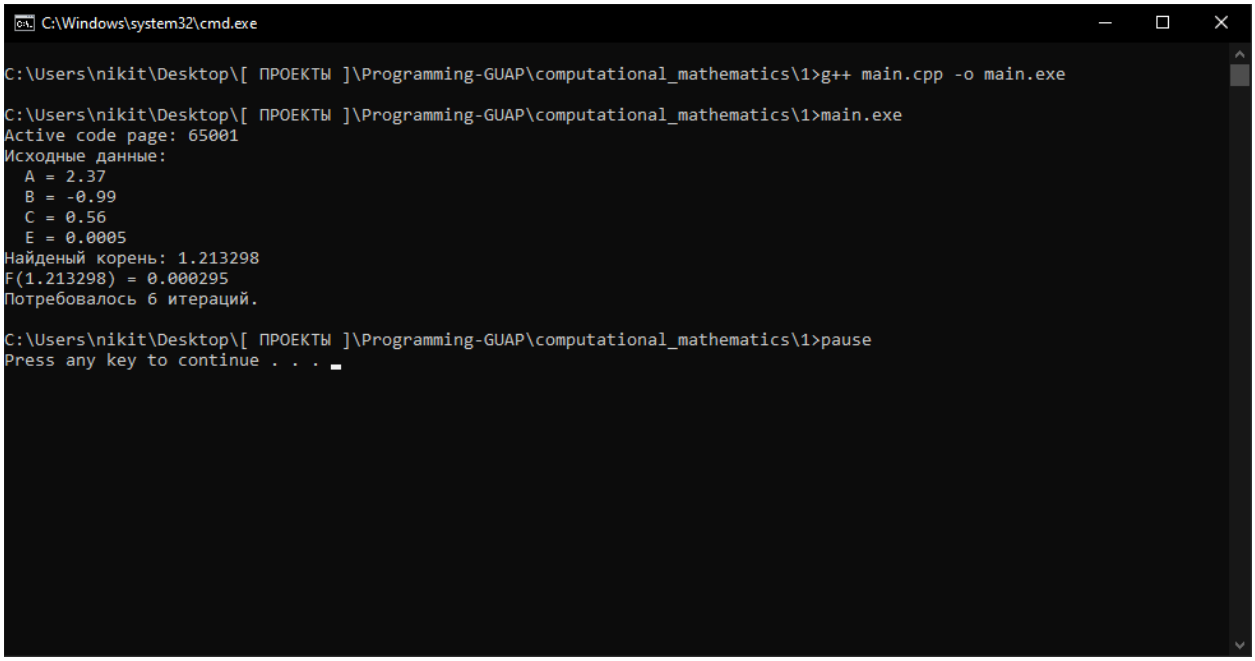
```
/*
  19 Вариант

  (a / x) + b * pow(M_E, c * x) = 0

  a = 2.37; b = -0.99; c = 0.56
  ε = 5·10-4
*/
```

```cpp
#include <iostream>
using namespace std;

#include <cmath>
#include <iomanip>

#define _USE_MATH_DEFINES

#define DEFAULT_VAREBLES false
#define DEFAULT_RANGE false

// проверка ввода
double read_double(const char* promt = ""){
  double x;
  cout << promt;
  while ( (scanf("%lf",&x) ) != 1 ) {
    cout << "Неверное введенное значение, попробуйте еще." << endl << promt;
    while(getchar() != '\n');
  }
  return x;
}

// класс реализующий метод ХОРД
class Method_HORD {
public:
  Method_HORD(double, double, double, double);
  double func(double);
  double derivative_func(double);
  double double_derivative_func(double);
  double find(double, double);

  int iterCount;
private:
  double a, b, c;
  double epsilon;
};

// конструктор
Method_HORD::Method_HORD(double _a, double _b, double _c, double _epsilon) {
  a = _a;
  b = _b;
  c = _c;
  epsilon = _epsilon;

  cout << "Исходные данные: " << endl;
  cout << "  A = " << a << endl;
  cout << "  B = " << b << endl;
  cout << "  C = " << c << endl;
  cout << "  E = " << epsilon << endl;
}

// функция
double Method_HORD::func(double x) {
  return (a / x) + b * exp(c * x);
}

// функция
double Method_HORD::derivative_func(double x) {
  return (b * c) * exp(c * x) - (a / pow(x, 2));
}

// функция
double Method_HORD::double_derivative_func(double x) {
  return (b * c * c) * exp(c * x) + (a * a) / pow(x, 3);
```

```
}


double Method_HORD::find(double a, double b) {
  iterCount = 0;
  double d, c;
  while (abs(abs(b) - abs(a)) > 2 * epsilon) {
    if (func(a) * double_derivative_func(a) > 0) {
      d = a - func(a) / derivative_func(a);
      c = (a * func(b) - b * func(a)) / (func(b) - func(a));
      a = d;
      b = c;
    } else if (func(b) * double_derivative_func(b) > 0) {
      d = b - func(b) / derivative_func(b);
      c = (a * func(b) - b * func(a)) / (func(b) - func(a));
      b = d;
      a = c;
    }
    iterCount++; // узнать кол-во итераций
    //cout << iterCount << " " << a << " " << b << endl;
  }
  return (a + b) / 2;
}



int main() {
  // смена кодировки
  system("chcp 65001");

  // переменные для подстановки в функцию
  double a, b, c, epsilon;
  if (DEFAULT_VAREBLES) {
    a = 2.37;
    b = -0.99;
    c = 0.56;
    epsilon = 5 * pow(10, -4);
  } else {
    a = read_double("A = ");
    b = read_double("B = ");
    c = read_double("C = ");
    epsilon = read_double("E = ");
  }

  // диапазон
  double xMin, xMax;
  if (DEFAULT_RANGE) {
    xMin = 0.1;
    xMax = 10;
  } else {
    while (true) {
      xMin = read_double("xMin = ");
      xMax = read_double("xMax = ");

      if (xMax <= xMin) {
        cout << "xMin не может быть больше или равен xMax." << endl;
      } else if (xMin <= 0) {
        cout << "xMin не может быть меньше или равен 0." << endl;
      } else break;
    }
  }

  Method_HORD method_HORD(a, b, c, epsilon);
```

```
  // рассчёт
  cout << endl << "Результат: " << method_HORD.find(xMin, xMax) << endl;
  cout << "Потребовалось " << method_HORD.iterCount << " итериций." << endl;

  return 0;
}
```

Скриншоты программы:



Сравнение результатов: Результаты вычислений программы и моего кода с учетом погрешности совпадают. Можно сделать вывод, что результаты достоверны.

С калькулятора	программа	разница
1.2103	1.213298	0.002998

Вывод: Мы освоили методы решения нелинейных уравнений и усовершенствовали навыки алгоритмизации вычислительных задач.