МИНИСТЕРСТВО НАУКИ
И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

НГТУ
НЭТИ | Факультет прикладной
математики и информатики

Практическое задание № 1

по дисциплине «Методы оптимизации»

ПРЯМЫЕ МЕТОДЫ РЕШЕНИЯ СЛАУ

Вариант 8

Группа ПМ-91

БАРСУКОВА НАТАЛЬЯ
ГРИБОВА АЛЕКСАНДРА
ЗАТОЛОЦКАЯ ЮЛИЯ

Преподаватели ФИЛИПОВА ЕЛЕНА ВЛАДИМИРОВНА

Новосибирск, 2022

1. Цель работы

Ознакомиться с методами одномерного поиска [3, 12], используемыми в многомерных методах минимизации функций n переменных. Сравнить различные алгоритмы по эффективности на тестовых примерах.

2. Постановка задачи

- Реализовать методы дихотомии, золотого сечения, исследовать их сходимость и провести сравнение по числу вычислений функции для достижения заданной точности ε от 10^{-1} до 10^{-7} . Построить график зависимости количества вычислений минимизируемой функции от десятичного логарифма задаваемой точности ε.
- Реализовать алгоритм поиска интервала, содержащего минимум функции.
- Реализовать метод Фибоначчи, сравнить его с методами дихотомии и золотого сечения

Тестовая функция:

$$f(x) = (x-8)^2,\ x \in [-2, 20].$$

3. Результаты исследований

Для точности $\varepsilon=10^{-7}$

1) Метод дихотомии

i	x_1	x_2	$f(x_1)$	$f(x_2)$	a_i	b_i	b_i-a_i	$\frac{b_{i-1}-a_{i-1}}{b_i-a_i}$
0	8,9999999950E+00	9,0000000050E+00	9,9999999000E-01	1,0000000100E+00	-2,0000000000E+00	9,0000000050E+00	1,1000000005E+01	1,9999999991E+00
1	3,4999999975E+00	3,5000000075E+00	2,0250000023E+01	2,0249999933E+01	3,4999999975E+00	9,0000000050E+00	5,5000000075E+00	1,9999999982E+00
2	6,2499999963E+00	6,2500000063E+00	3,0625000131E+00	3,0624999781E+00	6,2499999963E+00	9,0000000050E+00	2,7500000088E+00	1,9999999964E+00
3	7,6249999956E+00	7,6250000056E+00	1,4062500328E-01	1,4062499578E-01	7,6249999956E+00	9,0000000050E+00	1,3750000094E+00	1,9999999927E+00
4	8,3124999953E+00	8,3125000053E+00	9,7656247070E-02	9,7656253320E-02	7,6249999956E+00	8,3125000053E+00	6,8750000969E-01	1,9999999855E+00
5	7,9687499955E+00	7,9687500055E+00	9,7656278320E-04	9,7656215820E-04	7,9687499955E+00	8,3125000053E+00	3,4375000984E-01	1,9999999709E+00
6	8,1406249954E+00	8,1406250054E+00	1,9775389329E-02	1,9775392141E-02	7,9687499955E+00	8,1406250054E+00	1,7187500992E-01	1,9999999418E+00
7	8,0546874954E+00	8,0546875054E+00	2,9907221564E-03	2,9907232501E-03	7,9687499955E+00	8,0546875054E+00	8,5937509961E-02	1,9999998836E+00
8	8,0117187455E+00	8,0117187555E+00	1,3732899490E-04	1,3732922928E-04	7,9687499955E+00	8,0117187555E+00	4,2968759981E-02	1,9999997673E+00
9	7,9902343705E+00	7,9902343805E+00	9,5367520332E-05	9,5367325020E-05	7,9902343705E+00	8,0117187555E+00	2,1484384990E-02	1,9999995346E+00
10	8,0009765580E+00	8,0009765680E+00	9,5366543772E-07	9,5368496898E-07	7,9902343705E+00	8,0009765680E+00	1,0742197495E-02	1,9999990691E+00
11	7,9956054642E+00	7,9956054742E+00	1,9311944840E-05	1,9311856949E-05	7,9956054642E+00	8,0009765680E+00	5,3711037476E-03	1,9999981382E+00
12	7,9982910111E+00	7,9982910211E+00	2,9206431276E-06	2,9206089479E-06	7,9982910111E+00	8,0009765680E+00	2,6855568738E-03	1,9999962764E+00
13	7,9996337845E+00	7,9996337945E+00	1,3411377983E-07	1,3410645562E-07	7,9996337845E+00	8,0009765680E+00	1,3427834369E-03	1,9999925528E+00
14	8,0003051712E+00	8,0003051812E+00	9,3129483073E-08	9,3135586598E-08	7,9996337845E+00	8,0003051812E+00	6,7139671845E-04	1,9999851057E+00
15	7,9999694779E+00	7,9999694879E+00	9,3160002689E-10	9,3098968437E-10	7,9999694779E+00	8,0003051812E+00	3,3570335922E-04	1,9999702118E+00
16	8,0001373246E+00	8,0001373346E+00	1,8858033693E-08	1,8860780285E-08	7,9999694779E+00	8,0001373346E+00	1,6785667961E-04	1,9999404254E+00

17	8,0000534012E+00	8,0000534112E+00	2,8516898961E-09	2,8527580205E-09	7,9999694779E+00	8,0000534112E+00	8,3933339807E-05	1,9998808578E+00
18	8,0000114396E+00	8,0000114496E+00	1,3086322041E-10	1,3109211136E-10	7,9999694779E+00	8,0000114496E+00	4,1971669903E-05	1,9997617441E+00
19	7,9999904587E+00	7,9999904687E+00	9,1036188404E-11	9,0845462616E-11	7,9999904587E+00	8,0000114496E+00	2,0990834952E-05	1,9995236016E+00
20	8,0000009491E+00	8,0000009591E+00	9,0084559785E-13	9,1992817669E-13	7,9999904587E+00	8,0000009591E+00	1,0500417477E-05	1,9990476568E+00
21	7,9999957039E+00	7,9999957139E+00	1,8456302300E-11	1,8370480695E-11	7,9999957039E+00	8,0000009591E+00	5,2552087393E-06	1,9980971257E+00
22	7,9999983265E+00	7,9999983365E+00	2,8005202712E-12	2,7671507584E-12	7,9999983265E+00	8,0000009591E+00	2,6326043701E-06	1,9962014798E+00
23	7,9999996378E+00	7,9999996478E+00	1,3116951549E-13	1,2402604846E-13	7,9999996378E+00	8,0000009591E+00	1,3213021859E-06	1,9924317073E+00
24	8,0000002935E+00	8,0000003035E+00	8,6129201535E-14	9,2098757439E-14	7,9999996378E+00	8,0000003035E+00	6,6565109336E-07	1,9849771135E+00
25	7,9999999657E+00	7,9999999757E+00	1,1797697284E-15	5,9281416456E-16	7,9999999657E+00	8,0000003035E+00	3,3782554709E-07	1,9703989207E+00
26	8,0000001296E+00	8,0000001396E+00	1,6787088436E-14	1,9478388606E-14	7,9999999657E+00	8,0000001396E+00	1,7391277396E-07	1,9424999061E+00
27	8,0000000476E+00	8,0000000576E+00	2,2665797830E-15	3,3187520861E-15	7,9999999657E+00	8,0000000576E+00	9,1956387394E-08	1,8912527872E+00

2) Метод золотого сечения

i	x1	x2	f1	f2	a	b	b-a	b-a/b-a
0	3,1934955050E+00	6,4032522475E+00	2,3102485461E+01	2,5496033851E+00	-2,0000000000E+00	1,1596747753E+01	1,3596747753E+01	1,3596747753E+01
1	6,4032522475E+00	8,3869910100E+00	2,5496033851E+00	1,4976204181E-01	3,1934955050E+00	1,1596747753E+01	8,4032522475E+00	8,4032522475E+00
2	8,3869910100E+00	9,6130089900E+00	1,4976204181E-01	2,6017980019E+00	6,4032522475E+00	1,1596747753E+01	5,1934955050E+00	5,1934955050E+00
3	7,6292702275E+00	8,3869910100E+00	1,3744056420E-01	1,4976204181E-01	6,4032522475E+00	9,6130089900E+00	3,2097567425E+00	3,2097567425E+00
4	7,1609730300E+00	7,6292702275E+00	7,0396625643E-01	1,3744056420E-01	6,4032522475E+00	8,3869910100E+00	1,9837387625E+00	1,9837387625E+00
5	7,6292702275E+00	7,9186938124E+00	1,3744056420E-01	6,6106961352E-03	7,1609730300E+00	8,3869910100E+00	1,2260179800E+00	1,2260179800E+00
6	7,9186938124E+00	8,0975674251E+00	6,6106961352E-03	9,5194024347E-03	7,6292702275E+00	8,3869910100E+00	7,5772078247E-01	7,5772078247E-01
7	7,8081438402E+00	7,9186938124E+00	3,6808786073E-02	6,6106961352E-03	7,6292702275E+00	8,0975674251E+00	4,6829719755E-01	4,6829719755E-01
8	7,9186938124E+00	7,9870174528E+00	6,6106961352E-03	1,6854653244E-04	7,8081438402E+00	8,0975674251E+00	2,8942358492E-01	2,8942358492E-01

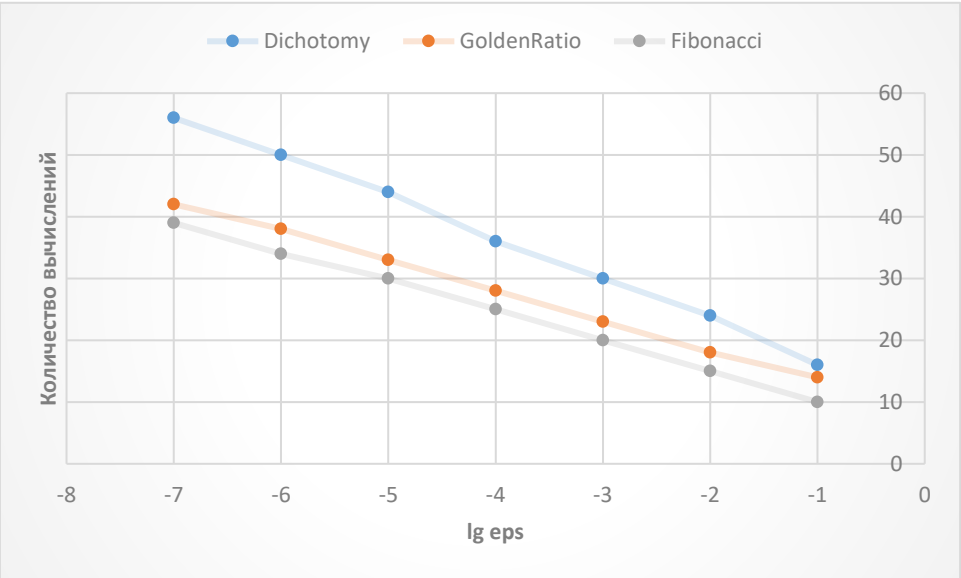
...

30	7,9999974166E+00	7,9999991418E+00	6,6742165076E-12	7,3657031619E-13	7,9999946251E+00	8,0000019332E+00	7,3081227452E-06	1,6180339889E+00
31	7,9999991418E+00	8,0000002080E+00	7,3657031619E-13	4,3265660239E-14	7,9999974166E+00	8,0000019332E+00	4,5166682501E-06	1,6180339889E+00
32	8,0000002080E+00	8,0000008670E+00	4,3265660239E-14	7,5164913157E-13	7,9999991418E+00	8,0000019332E+00	2,7914544942E-06	1,6180339889E+00
33	7,9999998007E+00	8,0000002080E+00	3,9706034786E-14	4,3265660239E-14	7,9999991418E+00	8,0000008670E+00	1,7252137559E-06	1,6180339884E+00
34	7,9999995490E+00	7,9999998007E+00	2,0337306292E-13	3,9706034786E-14	7,9999991418E+00	8,0000002080E+00	1,0662407393E-06	1,6180339883E+00
35	7,9999998007E+00	7,9999999563E+00	3,9706034786E-14	1,9098039633E-15	7,9999995490E+00	8,0000002080E+00	6,5897301749E-07	1,6180339877E+00
36	7,9999999563E+00	8,0000000524E+00	1,9098039633E-15	2,7501176327E-15	7,9999998007E+00	8,0000002080E+00	4,0726772266E-07	1,6180339880E+00
37	7,9999998969E+00	7,9999999563E+00	1,0633912561E-14	1,9098039633E-15	7,9999998007E+00	8,0000000524E+00	2,5170529572E-07	1,6180339849E+00
38	7,9999999563E+00	7,9999999930E+00	1,9098039633E-15	4,8692427602E-17	7,9999998969E+00	8,0000000524E+00	1,5556242783E-07	1,6180339895E+00
39	7,9999999930E+00	8,0000000157E+00	4,8692427602E-17	2,4706360055E-16	7,9999999563E+00	8,0000000524E+00	9,6142867889E-08	1,6180339868E+00

3) Метод Фибоначчи

i	x1	x2	f1	f2	a	b	b-a	b-a/b-a
0	3,1934955050 E+00	6,4032522475 E+00	2,3102485461 E+01	2,5496033851 E+00	-2,0000000000 E+00	1,1596747753 E+01	1,3596747753 E+01	1,6180339888 E+00
1	6,4032522475 E+00	8,3869910100 E+00	2,5496033851 E+00	1,4976204181 E-01	3,1934955050 E+00	1,1596747753 E+01	8,4032522475 E+00	1,6180339888 E+00
2	8,3869910100 E+00	9,6130089900 E+00	1,4976204181 E-01	2,6017980019 E+00	6,4032522475 E+00	1,1596747753 E+01	5,1934955050 E+00	1,6180339888 E+00
3	7,6292702275 E+00	8,3869910100 E+00	1,3744056420 E-01	1,4976204181 E-01	6,4032522475 E+00	9,6130089900 E+00	3,2097567425 E+00	1,6180339888 E+00
4	7,1609730300 E+00	7,6292702275 E+00	7,0396625643 E-01	1,3744056420 E-01	6,4032522475 E+00	8,3869910100 E+00	1,9837387625 E+00	1,6180339888 E+00
5	7,6292702275 E+00	7,9186938124 E+00	1,3744056420 E-01	6,6106961352 E-03	7,1609730300 E+00	8,3869910100 E+00	1,2260179800 E+00	1,6180339888 E+00
6	7,9186938124 E+00	8,0975674251 E+00	6,6106961352 E-03	9,5194024347 E-03	7,6292702275 E+00	8,3869910100 E+00	7,5772078247 E-01	1,6180339888 E+00
7	7,8081438402 E+00	7,9186938124 E+00	3,6808786073 E-02	6,6106961352 E-03	7,6292702275 E+00	8,0975674251 E+00	4,6829719755 E-01	1,6180339888 E+00
8	7,9186938124 E+00	7,9870174528 E+00	6,6106961352 E-03	1,6854653244 E-04	7,8081438402 E+00	8,0975674251 E+00	2,8942358492 E-01	1,6180339888 E+00
					...			
29	7,9999991424 E+00	8,0000019326 E+00	7,3546192262 E-13	3,7349385444 E-12	7,9999946250 E+00	8,0000064501 E+00	1,1825089580 E-05	1,6179775282 E+00
30	7,9999974152 E+00	7,9999991424 E+00	6,6814549121 E-12	7,3546192262 E-13	7,9999946250 E+00	8,0000019326 E+00	7,3076396276 E-06	1,6181818182 E+00
31	7,9999991424 E+00	8,0000002053 E+00	7,3546192262 E-13	4,2163954288 E-14	7,9999974152 E+00	8,0000019326 E+00	4,5174499519 E-06	1,6176470587 E+00
32	8,0000002053 E+00	8,0000008697 E+00	4,2163954288 E-14	7,5632505617 E-13	7,9999991424 E+00	8,0000019326 E+00	2,7901896758 E-06	1,6190476193 E+00
33	7,9999998067 E+00	8,0000002053 E+00	3,7349385376 E-14	4,2163954288 E-14	7,9999991424 E+00	8,0000008697 E+00	1,7272602761 E-06	1,6153846148 E+00
34	7,9999995410 E+00	7,9999998067 E+00	2,1067387780 E-13	3,7349385376 E-14	7,9999991424 E+00	8,0000002053 E+00	1,0629293996 E-06	1,6250000017 E+00
35	7,9999998067 E+00	7,9999999396 E+00	3,7349385376 E-14	3,6474010162 E-15	7,9999995410 E+00	8,0000002053 E+00	6,6433087476 E-07	1,6000000000 E+00
36	7,9999999396 E+00	8,0000000725 E+00	3,6474010162 E-15	5,2522572058 E-15	7,9999998067 E+00	8,0000002053 E+00	3,9859852397 E-07	1,6666666704 E+00

График зависимости количества вычислений целевой функции от логарифма задаваемой точности ε:



Процесс поиска интервала, содержащего минимум:

x0=5			x_0=15		
i	x_i	$f(x_i)$	i	x_i	$f(x_i)$
1	5,0000	9,0000	1	15,0000	49,0000
2	5,0000	9,0000	2	15,0000	49,0000
3	5,0000	9,0000	3	15,0000	49,0000
4	5,0000	9,0000	4	15,0000	49,0000
5	5,0000	9,0000	5	15,0000	49,0000
6	5,0000	9,0000	6	15,0000	49,0000
7	5,0000	9,0000	7	15,0000	49,0000
8	5,0000	9,0000	8	15,0000	48,9999
9	5,0000	8,9999	9	15,0000	48,9999
10	5,0000	8,9999	10	15,0000	48,9997
11	5,0000	8,9998	11	15,0000	48,9994
12	5,0001	8,9995	12	14,9999	48,9989
13	5,0002	8,9990	13	14,9998	48,9977
14	5,0003	8,9980	14	14,9997	48,9954
15	5,0007	8,9961	15	14,9993	48,9908
16	5,0013	8,9921	16	14,9987	48,9817
17	5,0026	8,9843	17	14,9974	48,9633
18	5,0052	8,9686	18	14,9948	48,9266
19	5,0105	8,9372	19	14,9895	48,8533
20	5,0210	8,8746	20	14,9790	48,7068
21	5,0419	8,7501	21	14,9581	48,4146
22	5,0839	8,5037	22	14,9161	47,8326
23	5,1678	8,0215	23	14,8322	46,6793
24	5,3355	7,0993	24	14,6645	44,4150
25	5,6711	5,4238	25	14,3289	40,0551
26	6,3422	2,7484	26	13,6578	32,0110
27	7,6844	0,0996	27	12,3156	18,6248
28	10,3687	5,6108	28	9,6313	2,6611
Интервал, содержащий минимум: [6,3422, 10,3687]			29	4,2626	13,9683
			Интервал, содержащий минимум: [4,2626, 12,3156]		

Вывод:

В ходе лабораторной работы мы ознакомились с методами одномерного поиска. Можем сделать вывод, что у каждого метода есть свои преимущества и недостатки. Метод

дихотомии сходится за меньшее число итераций (27, против 39 у золотого сечения и 36 у Фибоначчи), но при его выполнении требуется чаще считать значение функции в точке, что является более трудозатратной операцией (56 раз, против 42 у золотого сечения и 39 у Фибоначчи для $\varepsilon=10^{-7}$).

Результат работы метода поиск интервала, содержащего минимум функции зависит от входных данных. Чем ближе заданное значение x_0 к минимуму функции, тем уже будет найденный интервал, (для $x_{min}=8$ при $x_0=5$ длина найденного интервала ~4,0265 для $x_0=15$ уже ~8,053).

Код:

```
#include <iostream>
#include<vector>
#include <fstream>
#include <iomanip>
using namespace std;

double a = -2;
double b = 20;

double eps = 0.0001;

double f(double x) {
    return (x - 8) * (x - 8);
}
//-----------------------------------------------------------------------------
/* Метод дихотомии для поиска минимума. */
void calcDichotomy(double a, double b) {
    std::ofstream fout("Dichotomy.txt");
    fout << std::setprecision(11);
    cout << std::setprecision(11);
    double x1, x2, f1, f2;
    double delta = eps / 10;
    int fcount = 0; //сколько раз посчиталась ф-я
    for (int i = 0; abs(b - a) > eps; i++)
    {
        x1 = (b + a - delta) / 2;
        x2 = (b + a + delta) / 2;
        f1 = f(x1);
        f2 = f(x2);
        if (f1 > f2)
            a = x1;
        else
            b = x2;
        fcount += 2;
        fout << "Iteration: " << i << "  a: " << a << "  b: " << b << "  X_min: " <<
(a + b) / 2 << "  fcount: " << fcount << endl;
        cout << "Iteration: " << i << "  a: " << a << "  b: " << b << "  X_min: " <<
(a + b) / 2 << "  fcount: " << fcount << endl;
    }
}
//-----------------------------------------------------------------------------
/* Метод золотого сечения для поиска минимума. */
void calcGoldenRatio(double a, double b) {
    std::ofstream fout("GoldenRatio.txt");

    int i;
    int fcount = 0;
    const double A_COEFF((3 - sqrt(5.0)) / 2);
    const double B_COEFF((sqrt(5.0) - 1) / 2);
    double x1 = a + A_COEFF * (b - a);
```

```
    double x2 = a + B_COEFF * (b - a);
    double  f1 = f(x1);
    double  f2 = f(x2);
    fcount += 2;

    for (i = 0; abs(b - a) > eps; i++)
    {
        if (f1 > f2)
        {
            a = x1;
            x1 = x2;
            f1 = f2;
            x2 = a + B_COEFF * (b - a);
            f2 = f(x2);
        }

        else {
            b = x2;
            x2 = x1;
            f2 = f1;
            x1 = a + A_COEFF * (b - a);
            f1 = f(x1);
        }
        fcount++;
        fout << "Iteration: " << i << "  a: " << a << "  b: " << b << "  X_min: " <<
(a + b) / 2 << "  fcount: " << fcount << endl;
        cout << "Iteration: " << i << "  a: " << a << "  b: " << b << "  X_min: " <<
(a + b) / 2 << "  fcount: " << fcount << endl;
    }
}
//-------------------------------------------------------------------------------
/* Функция для нахождения интервала, содержащего минимум для унимодальной функции.
*/
void findInterval(double a, double b, double x0)
{
    std::ofstream fout("Interval.txt");
    int fcount;
    double delta = eps / 10;
    double f0 = f(x0);
    double f1 = f(x0 + delta);
    double x1, h, x00;
    int k = 0;
    if (f0 > f1) //Если ф-я на участке убывает
    {
        k = 1;
        h = delta;

    }
    else //Если ф-я на участке возрастает
        h = -delta;

    x1 = x0 + h;
    do {
        h *= 2;
        x00 = x0;
        x0 = x1;
        x1 = x0 + h;
        f0 = f1;
        f1 = f(x1);
        k++;
        if (x00 > x1)
        {
            fout << x1 << " " << x00 << " iters " << k << endl;
            cout << x1 << " " << x00 << " iters " << k << endl;
        }
```

```
        else
        {
            fout << x00 << " " << x1 << " iters " << k << endl;
            cout << x00 << " " << x1 << " iters " << k << endl;
        }
    } while (f1 < f0);

    a = x00;
    b = x1;
    if (x00 > x1)
    {
        fout << x1<< " " << x00 << " iters " << k << endl;
        cout << x1 << " " << x00 << " iters " << k << endl;
    }
    else
    {
        fout << x00 << " " << x1 << " iters " << k << endl;
        cout << x00 << " " << x1 << " iters " << k << endl;
    }
}
//-------------------------------------------------------------------------
/** Метод Фибоначчи для поиска минимума. */
void calcFibonacci(double a, double b)
{
    std::ofstream fout("Fibonacci.txt");

    int fcount = 0;
    int i;
    double x1, x2, f1, f2;
    double n = 2, max = (b - a) / eps, new_number = 0;//?
    vector<int> fibonacci_num;//массив чисел фибоначи

    fibonacci_num.push_back(1);
    fibonacci_num.push_back(1);
    for (; max > new_number; n++)//заполняем массив числами фибоначи
    {
        new_number = fibonacci_num[n - 1] + fibonacci_num[n - 2];
        fibonacci_num.push_back(new_number);
    }
    n = fibonacci_num.size() - 3; //1 число привысившее максимум, 2 числа для
использования формулы n+2

    x1 = a + fibonacci_num[n] * (b - a) / fibonacci_num[n + 2];
    x2 = a + fibonacci_num[n + 1] * (b - a) / fibonacci_num[n + 2];

    f1 = f(x1);
    f2 = f(x2);
    fcount += 2;
    for (i = 0; i < n - 2; i++) {
        if (f1 > f2)
        {
            a = x1;
            x1 = x2;
            f1 = f2;
            x2 = a + fibonacci_num[n - i - 1] * (b - a) / fibonacci_num[n - i];
            f2 = f(x2);
        }
        else
        {
            b = x2;
            x2 = x1;
            f2 = f1;
            x1 = a + fibonacci_num[n - i - 2] * (b - a) / fibonacci_num[n - i];
            f1 = f(x1);
        }
```

```
        fcount++;
        fout << "Iteration: " << i << "  a: " << a << "  b: " << b << "  X_min: " <<
(a + b) / 2 << "  fcount: " << fcount << endl;
        cout << "Iteration: " << i << "  a: " << a << "  b: " << b << "  X_min: " <<
(a + b) / 2 << "  fcount: " << fcount << endl;
    }
    fout << "Iteration: " << i << "  a: " << a << "  b: " << b << "  X_min: " << (a
+ b) / 2 << "  fcount: " << fcount << endl;
    cout << "Iteration: " << i << "  a: " << a << "  b: " << b << "  X_min: " << (a
+ b) / 2 << "  fcount: " << fcount << endl;

}
int main()
{
    float x0;
    cout << "Enter the initial value: " << endl << "x0 = ";
    cin >> x0;
    cout << "findInterval: " << endl;
    findInterval(a, b, x0);
    cout << "calcDichotomy: " << endl;
    calcDichotomy(a, b);
    cout << "calcGoldenRatio: " << endl;
    calcGoldenRatio(a, b);
    cout << "calcFibonacci: " << endl;
    calcFibonacci(a, b);
}
```