



**Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Московский государственный технический университет  
имени Н.Э. Баумана  
(национальный исследовательский университет)»  
(МГТУ им. Н.Э. Баумана)**

---

**ФАКУЛЬТЕТ «Информатика и системы управления» (ИУ)**

**КАФЕДРА «Информационная безопасность» (ИУ8)**

**Отчёт**

по лабораторной работе № 1  
по дисциплине «Теория систем и системный анализ»

**Тема: «Исследование методов прямого поиска экстремума унимодальной функции  
одного переменного»**

Вариант 15

Выполнил: Ушаков З. М.,  
студент группы ИУ8-31

Проверил: Коннова Н.С.,  
доцент каф. ИУ8

## 1. Цель работы

Исследовать функционирование и провести сравнительный анализ различных алгоритмов прямого поиска экстремума (пассивный поиск, метод дихотомии, золотого сечения, Фибоначчи) на примере унимодальной функции одного переменного.

## 2. Условие задачи

На интервале  $[9; 12]$  задана унимодальная функция одного переменного  $f(x) = x^2 \sin(x)$ . Используя метод Фибоначчи, найти интервал нахождения минимума  $f(x)$  с заданным количеством итераций. Провести сравнение с методом оптимального пассивного поиска. Результат, в зависимости от числа точек разбиения  $N$ , представить в виде таблицы.

## 3. Ход работы

Построим график заданной функции и определим местонахождение её минимума:

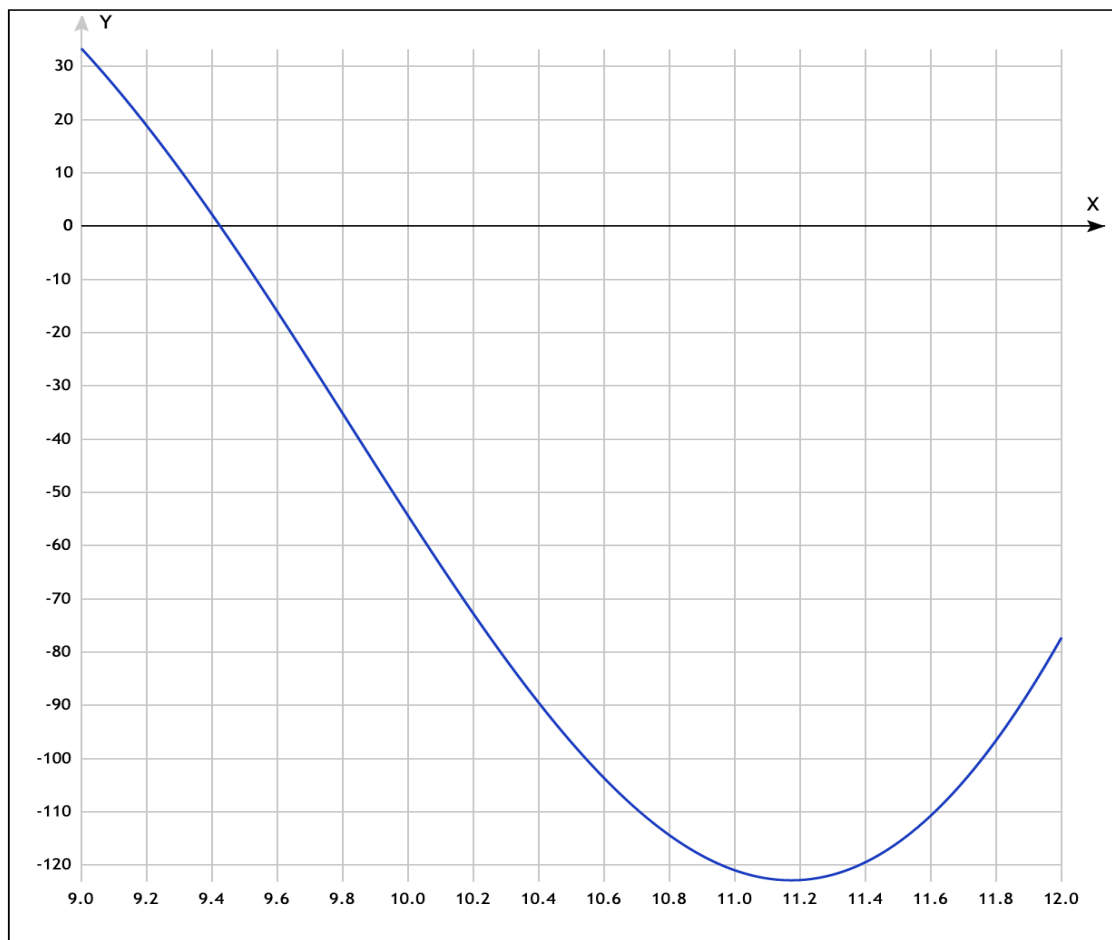


Рисунок 1 - График Функции  $f(x) = x^2 \sin(x)$  на интервале  $[9, 12]$

Как видно из графика, функция достигает своего минимума в точке  $x = 11,1727$ .  
Теперь проведём программный расчет при помощи методов оптимального пассивного поиска и Фибоначчи.

Результат работы программы представлен в таблицах 1 и 2:

**Таблица 1** – результат работы метода пассивного поиска

Количество точек (N)	Значение $x$ в минимуме
-----	-----
1	10.500000+-1.500000
2	11.000000+-1.000000
3	11.250000+-0.750000
4	11.400000+-0.600000
5	11.000000+-0.500000
6	11.142857+-0.428571
7	11.250000+-0.375000
8	11.333333+-0.333333
9	11.100000+-0.300000
10	11.181818+-0.272727
11	11.250000+-0.250000
12	11.076923+-0.230769
13	11.142857+-0.214286
14	11.200000+-0.200000
15	11.250000+-0.187500
16	11.117647+-0.176471
17	11.166667+-0.166667
18	11.210526+-0.157895
19	11.100000+-0.150000
20	11.142857+-0.142857
21	11.181818+-0.136364
22	11.217391+-0.130435

23	11.125000+-0.125000
24	11.160000+-0.120000
25	11.192308+-0.115385
26	11.222222+-0.111111
27	11.142857+-0.107143
28	11.172414+-0.103448
29	11.200000+-0.100000

-----

**Таблица 2** – результат работы метода Фибоначчи

Количество	Значение y	
точек (N)		

-----

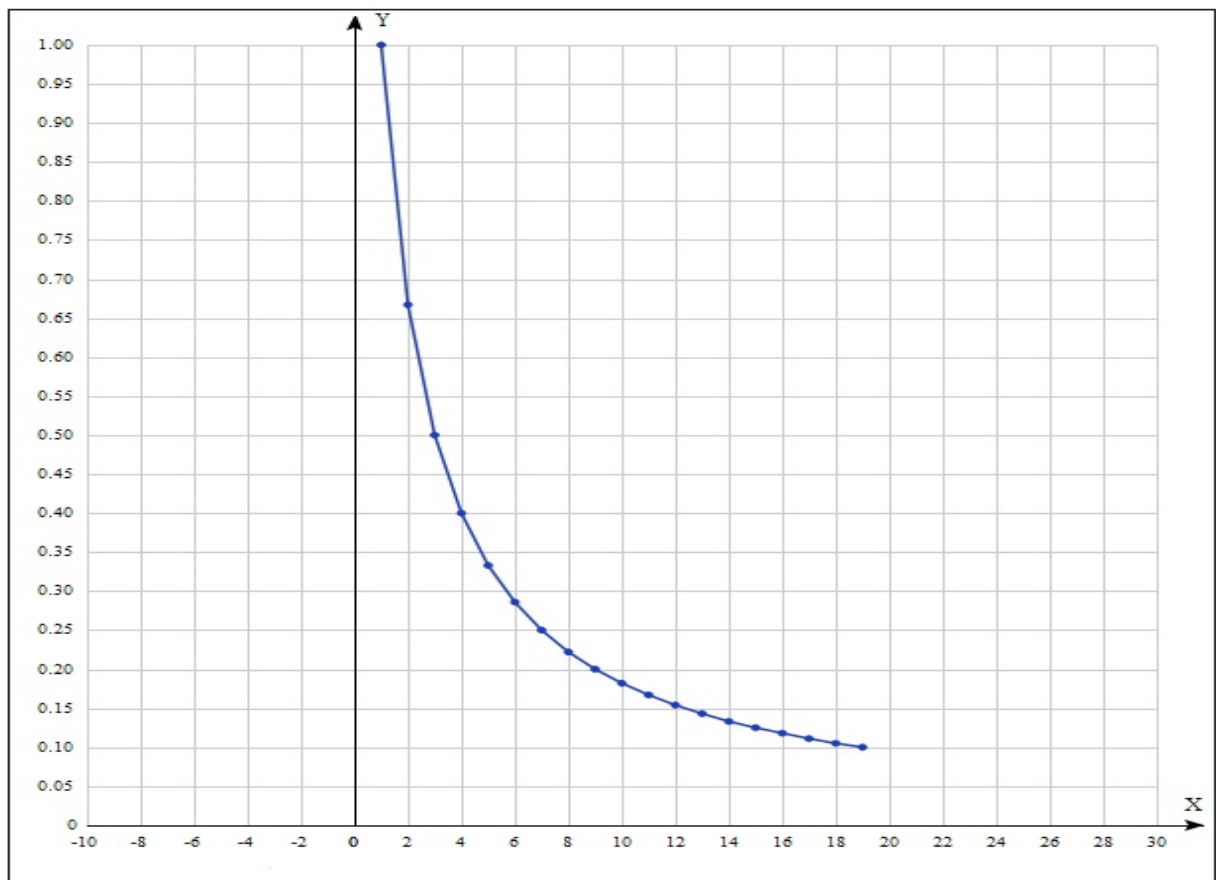
1	11.291796068
2	11.562305899
3	11.124611797
4	11.021286236
5	11.188470506
6	11.227937359
7	11.16407865
8	11.149003654
9	11.17339551
10	11.179153646
11	11.169836786
12	11.175594922
13	11.172036198
14	11.171196098
15	11.172555411
16	11.172876297
17	11.172357085
18	11.172677971
19	11.172753736

20	11.172631175	
21	11.17270694	
22	11.172724767	
23	11.172695799	
24	11.172713626	
25	11.172702484	
26	11.172709169	
27	11.172704712	
28	11.17270694	
29	11.172709169	

-----

Минимальное значение функции в точке: 11.172709169

Построим график зависимостей погрешности от числа точек N (для оптимального пассивного поиска).



Ссылка на git-репозиторий: [https://github.com/HvarZ/tsisa\\_lab01](https://github.com/HvarZ/tsisa_lab01)

## 4. Выводы

В конечном итоге расчета разными методами показали что метод Фибоначчи эффективнее метода оптимально пассивного поиска при нахождении экстремума унимодальной функции одного переменного.

### Приложение 1. Исходный код программы

#### Файл Fibonacci.h:

```
#pragma once
#include <iostream>
#include <cmath>
#include <vector>
#include <string>
#include <iomanip>
#include <algorithm>

using std::cout;
using std::endl;

double f (const double& x){
    return (pow(x ,2) * sin(x));
}

int64_t Fib( const size_t &n )
{
    if (n < 1) return 0;
    int64_t f1 = 0 , f2 = 1, fn = 0;
    for ( size_t i = 1; i < n; ++i )
    {
        fn = f1 + f2;
        f1 = f2;
        f2 = fn;
    }
    return fn;
}

std::vector<double> Fib_values(double& a, double& b, size_t n)
{
    std::vector<double> values;
    double x1 = a + (b - a) * Fib(n) / Fib(n + 2);
    double x2 = a + b - x1;
    double y1 = f(x1);
    double y2 = f(x2);
    while (n-->0) {
        if (y1 > y2) {
            a = x1;
            x1 = x2;
            x2 = b - (x1 - a);
            y1 = y2;
            y2 = f(x2);
            values.push_back(x2);
        }
        else {
            b = x2;
            x2 = x1;
            x1 = a + (b - x2);
            y2 = y1;
            y1 = f(x1);
        }
    }
}
```

```

        values.push_back(x1);
    }
}
return values;
}

void PrintValues (const std::vector<double>& values){
    cout << "Расчет методом Фибоначчи" << endl;
    cout << "|" << std::setw(23) << std::left << "Количество" << "|"
    << std::setw(22) << std::left << "Значение y" << "|" << endl;
    cout << "|" << std::setw(18) << std::left << "точек (N)" << "|"
    << std::string(14, ' ') << "|" << endl;
    cout << std::string (30, '-') << endl;
    for (size_t i = 0; i < values.size(); i++){
        cout << "|" << std::setw(13) << i + 1 << "|" << std::setw(14) <<
std::setprecision(11) << values[i] << "|" << endl;
    }
    cout << std::string (30, '-') << endl;
    cout << "Минимальное значение функции в точке: " << values[values.size()-
1] << endl;
}

```

### Файл Passive\_search.h:

```

#include "Fibonacci.h"

std::vector<std::pair<double, double>> PasValues(const double& a, const
double& b, const double& quantity) {
    std::vector<std::pair<double, double>> values;
    size_t N = 1;
    double delta = (b - a) / (N + 1);
    double minX;
    while (delta > quantity) {
        std::vector<double> valuesY;
        delta = (b - a) / (N + 1);
        for (size_t k = 1; k <= N; ++k) {
            valuesY.push_back(f((b - a) / (N + 1) * k + a));
        }
        size_t kForMinY = std::min_element(valuesY.begin(), valuesY.end()) -
valuesY.begin() + 1;
        minX = (b - a) / (N + 1) * kForMinY + a;
        values.push_back({ minX, delta });
        N++;
    }
    return values;
}

void PrintPasValues (const std::vector<std::pair<double, double>>& values) {
    cout << "Расчет методом оптимально пассивного поиска" << endl;
    cout << "|" << std::setw(23) << std::left << "Количество" << "|"
    << std::setw(28) << std::left << "Значение x" << "|" << endl;
    cout << "|" << std::setw(18) << std::left << "точек (N)" << "|"
    << std::setw(29) << std::left << "в минимуме" << "|" << endl;
    cout << std::string (36, '-') << endl;
    for (size_t i = 0; i < values.size(); i++){
        cout << "|" << std::setw(13) << i + 1 << "|" << std::setw(20)
        << std::to_string(values[i].first) + "+-" +
std::to_string(values[i].second) << "|" << endl;
    }
    cout << std::string (36, '-') << endl;
}

```

### Файл main.cpp:

```
#include "Fibonacci.h"
#include "Passive_search.h"

double N = 29;
double QUANTITY = 0.1;
double LEFT_EDGE = 9.0;
double RIGHT_EDGE = 12.0;

int main()
{
    setlocale(LC_ALL, "Russian");

    cout << "Вариант 15" << endl;
    cout << "Функция:  $y = x^2 * \sin(x)$ " << endl;
    cout << "Интервал: [9, 12]" << endl << endl;

    std::vector<std::pair<double, double>> values = PasValues(LEFT_EDGE,
RIGHT_EDGE, QUANTITY);
    PrintPasValues(values);
    cout << endl;

    std::vector<double> Fibvalues = Fib_values(LEFT_EDGE, RIGHT_EDGE, N);
    PrintValues(Fibvalues);
}
```