**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ**

**Московский государственный технический университет**

**им. Н.Э. Баумана**

**(МГТУ им. Н.Э. Баумана)**

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

Отчёт

Лабораторная работа № 2

По дисциплине: «Теория систем и системный анализ»

# Тема: «Исследование метода случайного поиска экстремума

# функции одного переменного»

# Вариант 8

Выполнил: Забродина М.П.,

студент группы ИУ8-32

Проверил: Коннова Н.С.

Доцент каф. ИУ8

г. Москва 2020 г.

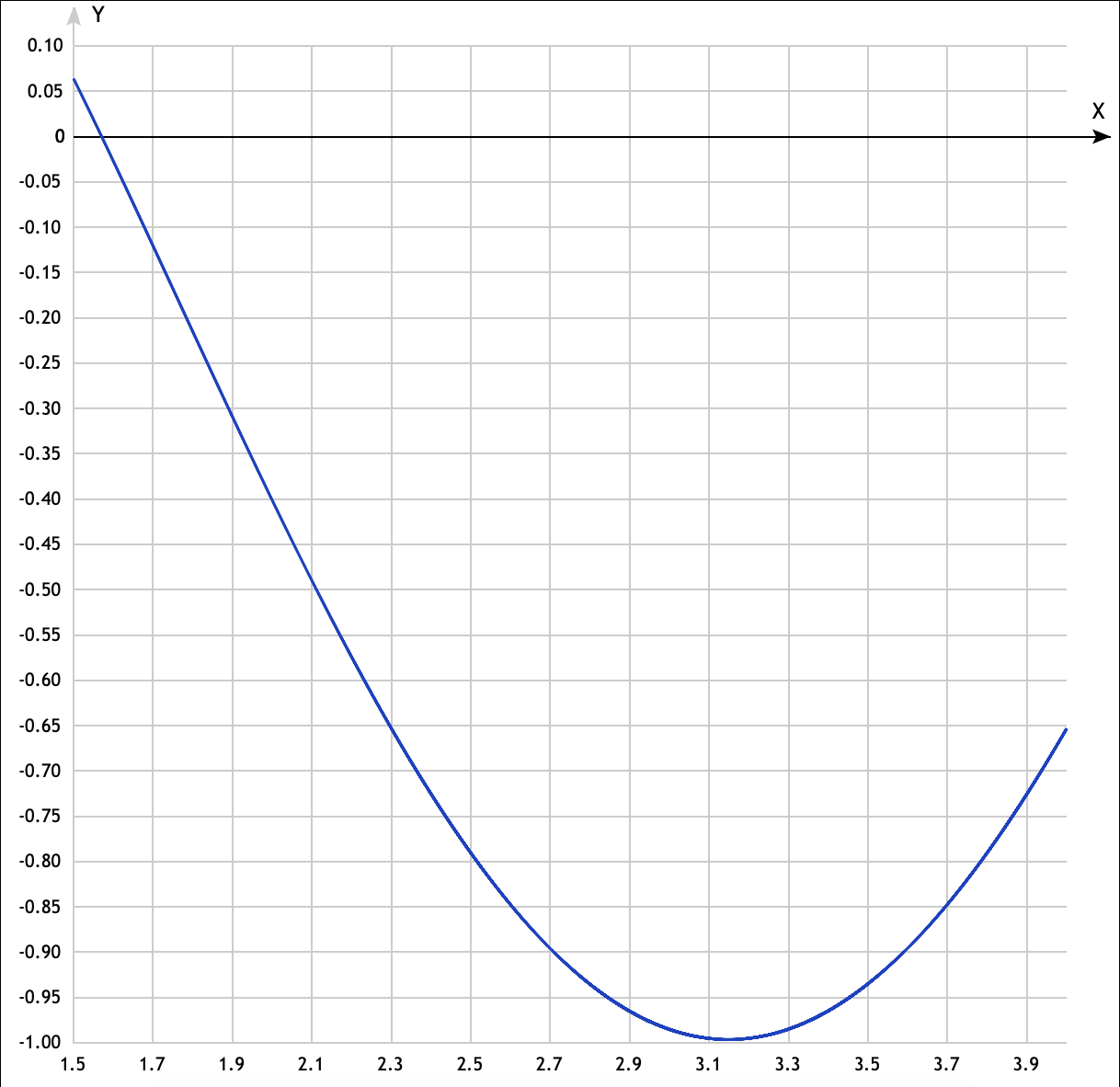
**Цель:**

Изучение метода случайного поиска экстремума на примере унимодальной и мультимодальной функций одного переменного.

**Условие задачи:**

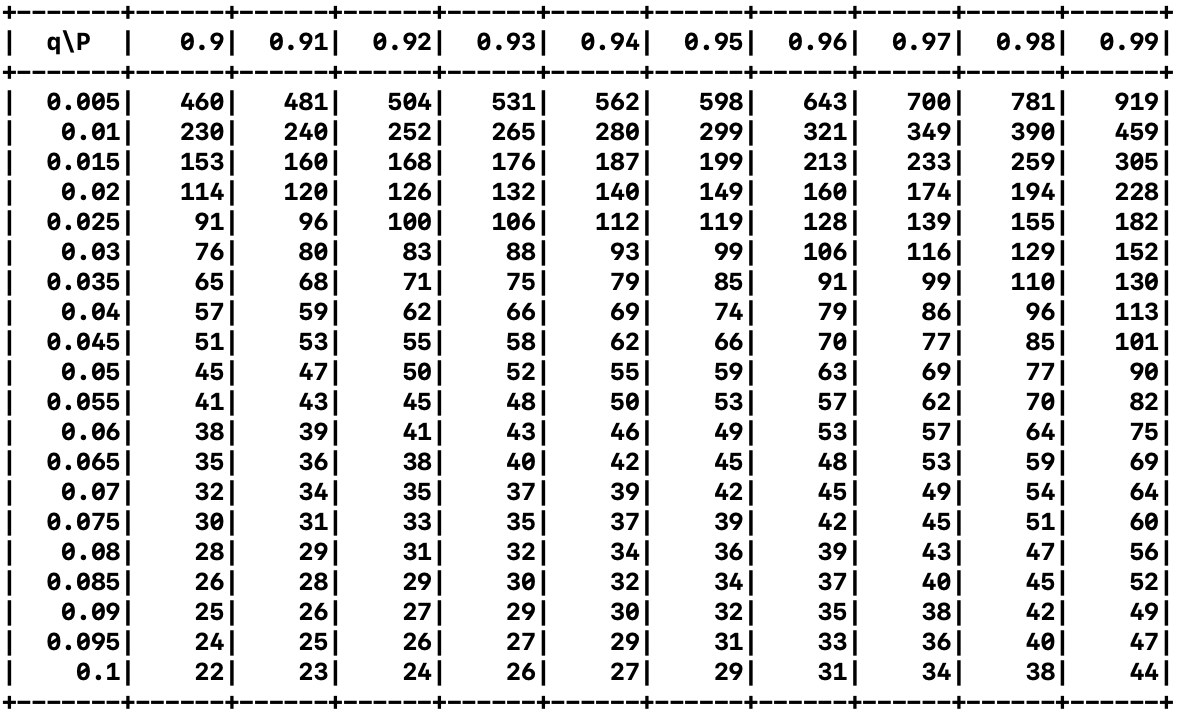
1.На интервале [1.5;4] задана унимодальная функция одного переменного f(x)=cos(x)\*th(x). Используя метод случайного поиска осуществить поиск минимума f(x) с заданной вероятностью попадания в окрестность экстремума P при допустимой длине интервала неопределенности ε. Определить необходимое число испытаний N. Численный эксперимент выполнить для значений P=0,90,0,91,…,0,99 и значений ε=(b-a)q, где q=0,005,0,010,…,0,100.

2. При аналогичных исходных условиях осуществить поиск минимума f(x), модулированной сигналом sin5x, т.е. мультимодальной функции f(x)\*sin5x.

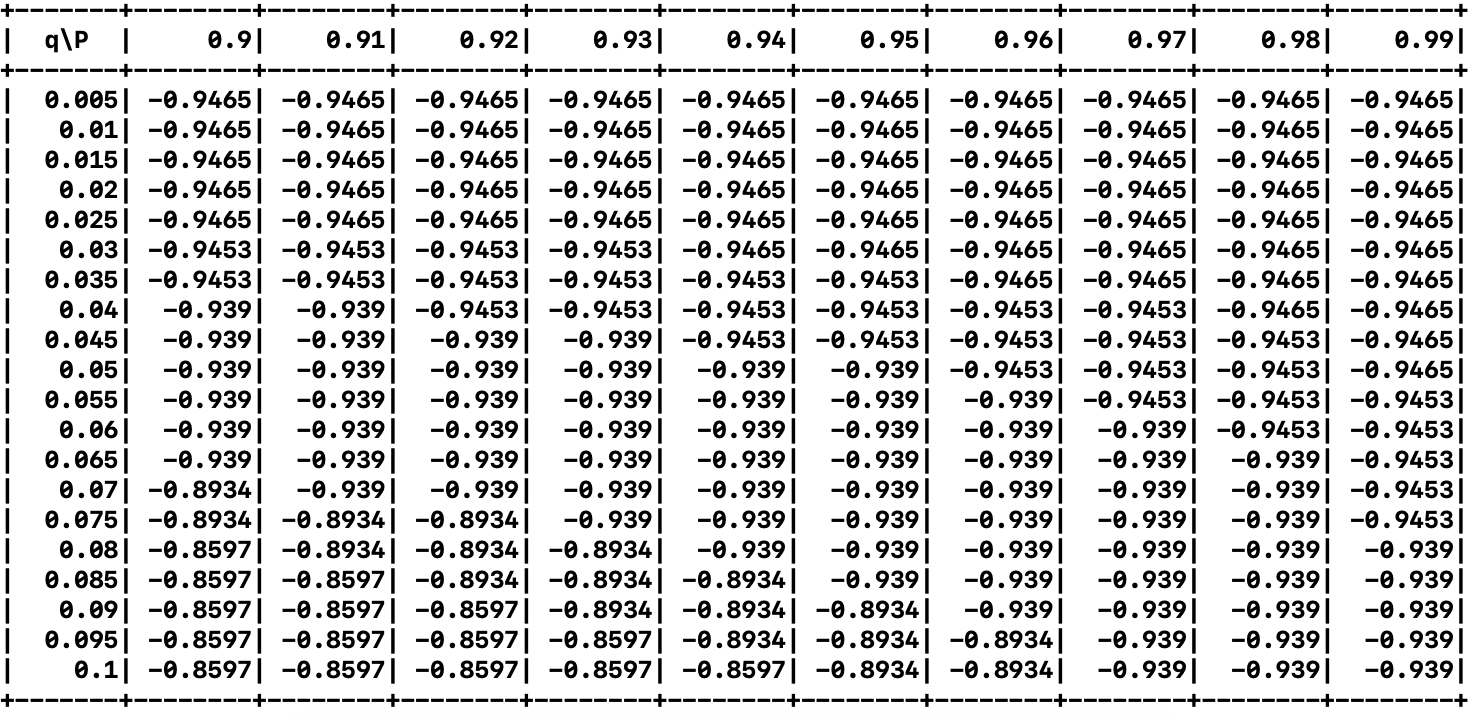


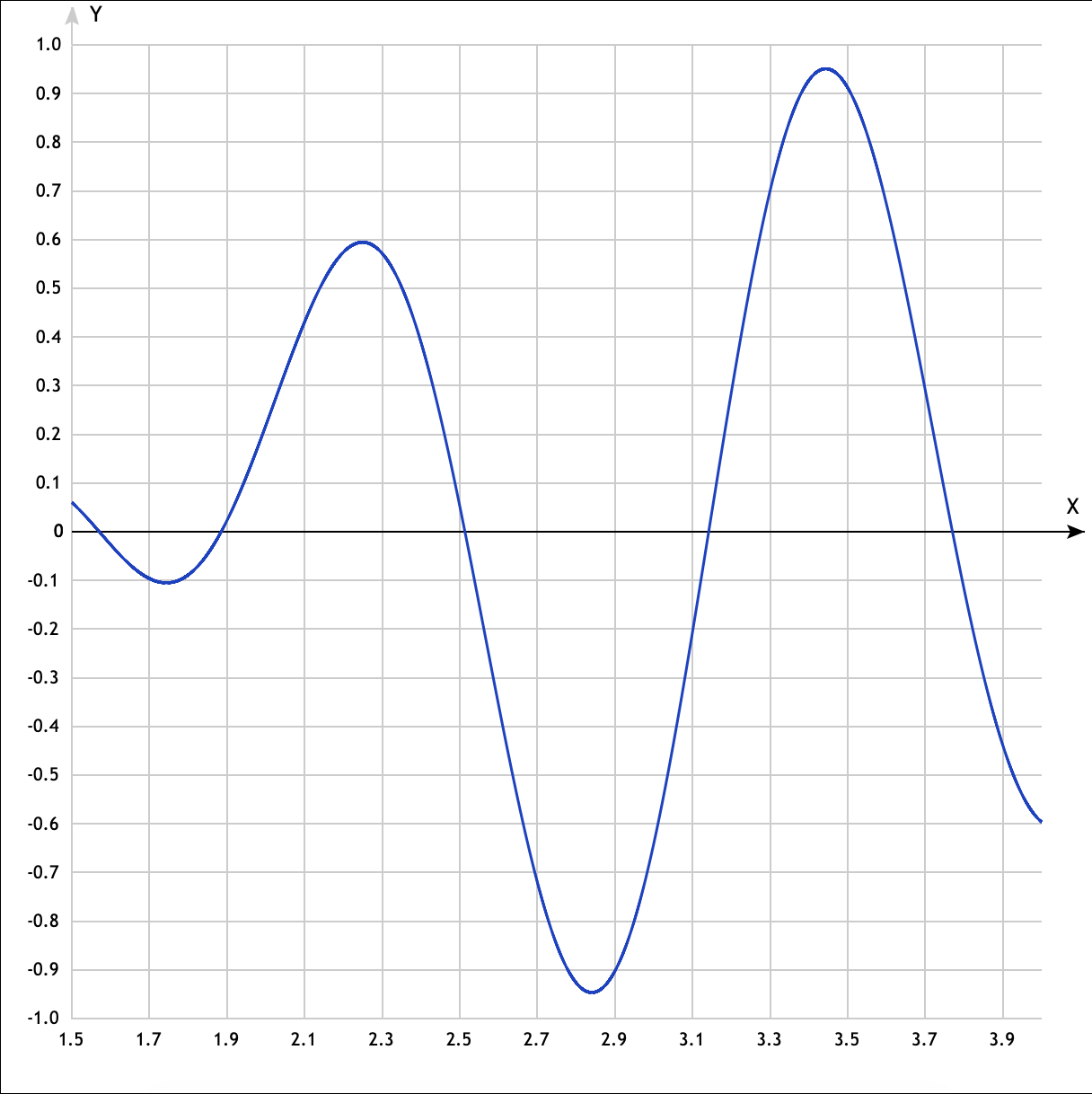
**Рисунок 1.** График функции f(x)=cos(x)\*th(x) на интервале [1.5;4].

**Таблица 1.** Зависимость N от P и q.



**Таблица 2.** Результаты поиска экстремума f(x) в зависимости от P и q.





**Рисунок 2.** График функции f(x)\*sin5x на интервале [1.5;4].

**Код программы:**

//Кафедра "Информационная безопасность"

//ИУ8-32

//Забродина М.П.

//Лабораторная работа №1(Теория систем и системный анализ)

#include <iostream>

#include <math.h>

#include <iomanip>

#include <vector>

**using** **namespace** std;

**const** **double** a = 1.5, b = 4.;

**double** func(**double** x) {

**return** cos(x)\*tanh(x);

}

**void** delimeter(**int** f,**int** l){

cout <<"+";

cout <<setfill('-') <<setw(f) <<"+";

**for** (**int** i = 0; i < 10; i++){

cout <<setw(l) <<"+";

}

cout <<endl;

}

**void** printfirsttable(**double** q, **double** P){

delimeter(8,7);

cout <<"| q\\P |";

cout <<setfill(' ');

**for** (**int** i = 0; i < 10; i++){

cout <<setw(6) <<P + i \* 0.01 <<"|";

}

cout <<endl;

delimeter(8,7);

cout <<setfill(' ');

**for** ( q = 0.005 ; q < 0.105;){

cout <<"|" <<setw(7) <<q <<"|";

**for** ( P = 0.9 ; P < 1;){

**double** N = log(1 - P) / log(1 - q);

cout <<setw(6) <<ceil(N) <<"|";

P += 0.01;

}

cout <<endl;

q += 0.005;

}

delimeter(8,7);

}

**long** **double** min(**double** q, **double** P){

srand(**static\_cast**<**unsigned** **int**>(time(**nullptr**)));

**int** N = ceil(log(1 - P) / log(1 - q));

vector<**long** **double**> values;

**for** (**int** i = 0; i < N; ++i){

**double** x = a + (b - a) \* rand() / RAND\_MAX;

values.push\_back(func(x));

}

**return** \*min\_element(values.begin(), values.end());

}

**void** get\_min(**double** q, **double** P){

delimeter(8,9);

cout <<"| q\\P |";

cout <<setfill(' ');

**for** (**int** i = 0; i < 10; i++){

cout <<setw(8) <<P + i \* 0.01 <<"|";

}

cout <<endl;

delimeter(8,9);

cout <<setfill(' ');

**for** ( q = 0.005 ; q < 0.105;){

cout <<"|" <<setw(7) <<q <<"|";

**for** ( P = 0.9 ; P < 1;){

cout <<setw(8) <<setprecision(4) <<min(q, P) <<"|";

P += 0.01;

}

cout <<endl;

q += 0.005;

}

delimeter(8,9);

}

**long** **double** min(**double** q, **double** P, **int** n){

srand(**static\_cast**<**unsigned** **int**>(time(**nullptr**)));

**int** N = ceil(log(1 - P) / log(1 - q));

vector<**long** **double**> values;

**for** (**int** i = 0; i < N; ++i){

**double** x = a + (b - a) \* rand() / RAND\_MAX;

values.push\_back(func(x) \* sin(5\*x));

}

**return** \*min\_element(values.begin(), values.end());

}

**void** get\_min(**double** q, **double** P, **int** f){

delimeter(8,9);

cout <<"| q\\P |";

cout <<setfill(' ');

**for** (**int** i = 0; i < 10; i++){

cout <<setw(8) <<P + i \* 0.01 <<"|";

}

cout <<endl;

delimeter(8,9);

cout <<setfill(' ');

**for** ( q = 0.005 ; q < 0.105;){

cout <<"|" <<setw(7) <<q <<"|";

**for** ( P = 0.9 ; P < 1;){

cout <<setw(8) <<setprecision(4) <<min(q, P, 2) <<"|";

P += 0.01;

}

cout <<endl;

q += 0.005;

}

delimeter(8,9);

}

**int** main() {

**double** q = 0.005, P = 0.9;

printfirsttable(q, P);

cout <<endl;

get\_min(q, P);

get\_min(q, P, 2);

**return** 0;

}

**Вывод**

Из полученных таблиц и графиков видно, что применимость метода случайного поиска не зависит от того, является ли функция унимодальной и мультимодальной. Для уменьшения интервала неопределенности необходимо увеличивать число случайных точек N.