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 12

Выполнил: Мишков А.О

студент группы ИУ8-32

Проверил: Коннова Н. С

доцент каф. ИУ8

г. Москва, 2020 г.

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. Постановка задачи

На интервале [7, 11] задана унимодальная функция одной переменной f(x) = cos(x)*th(x) . Используя метод Фибоначчи, найти интервал нахождения минимума f(x) при заданном числе точек N. 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 её минимум

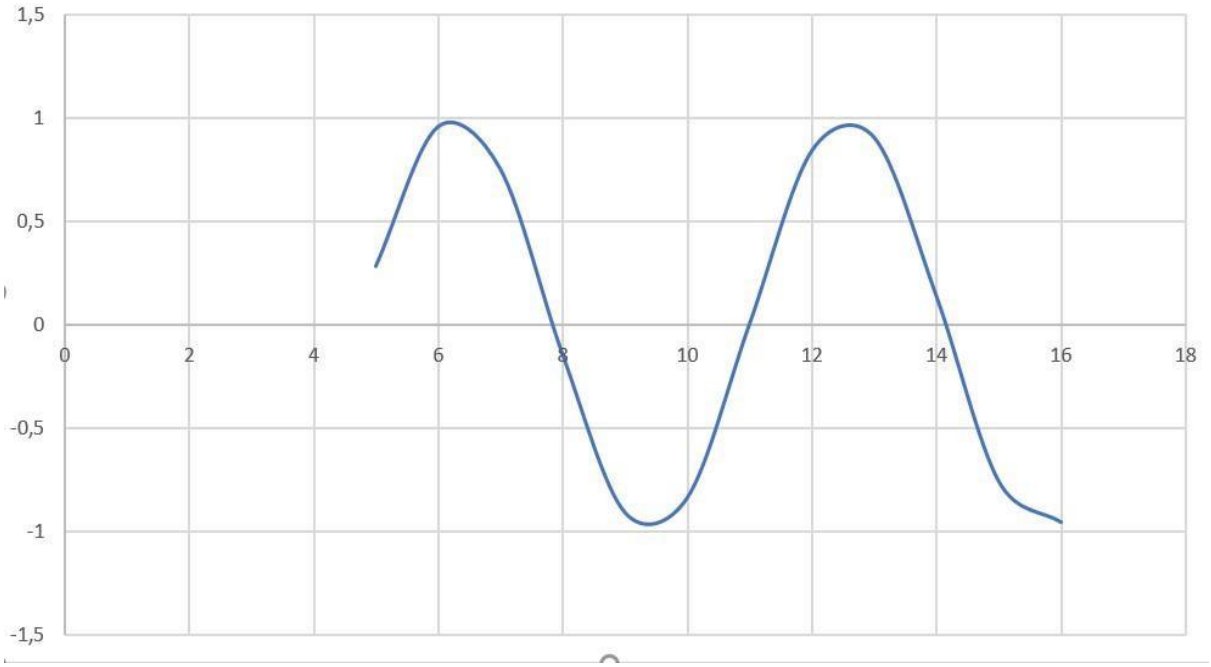

График функции f(x) = cos(x)*th(x) на отрезке [7; 11]

Минимум точки x:= 9.43

Теперь рассчитаем с помощью методов Фибоначчи и оптимального пассивного поиска.

Результаты программы приведены ниже.

Метод оптимального пассивного поиска

1	9	2
2	9.6666666667	1.33
3	9	1
4	9.4	0.8
5	9.6666666667	0.667
6	9.2857142857	0.571
7	9.5	0.5
8	9.2222222222	0.444
9	9.4	0.4
10	9.5454545455	0.364
11	9.3333333333	0.333
12	9.4615384615	0.308
13	9.2857142857	0.286
14	9.4	0.267
15	9.5	0.25
16	9.3529411765	0.235
17	9.4444444444	0.222
18	9.5263157895	0.211
19	9.4	0.2
20	9.4761904762	0.19
21	9.3636363636	0.182
22	9.4347826087	0.174
23	9.5	0.167
24	9.4	0.16
25	9.4615384615	0.154
26	9.3703703704	0.148
27	9.4285714286	0.143
28	9.4827586207	0.138
29	9.4	0.133
30	9.4516129032	0.129
31	9.375	0.125
32	9.4242424242	0.121
33	9.4705882353	0.118
34	9.4	0.114
35	9.4444444444	0.111
36	9.3783783784	0.108
37	9.4210526316	0.105
38	9.4615384615	0.103
39	9.4	0.1
40	9.4390243902	0.0976
41	9.380952381	0.0952
42	9.4186046512	0.093
43	9.4545454545	0.0909
44	9.4	0.0889
45	9.4347826087	0.087
46	9.3829787234	0.0851
47	9.4166666667	0.0833
48	9.4489795918	0.0816
49	9.4	0.08
50	9.431372549	0.0784
51	9.4615384615	0.0769
52	9.4150943396	0.0755
53	9.4444444444	0.0741
54	9.4	0.0727
55	9.4285714286	0.0714
56	9.4561403509	0.0702
57	9.4137931034	0.069
58	9.4406779661	0.0678
59	9.4	0.0667
60	9.4262295082	0.0656

61	9.4516129032	0.0645
62	9.4126984127	0.0635
63	9.4375	0.0625
64	9.4	0.0615
65	9.4242424242	0.0606
66	9.447761194	0.0597
67	9.4117647059	0.0588
68	9.4347826087	0.058
69	9.4	0.0571
70	9.4225352113	0.0563
71	9.4444444444	0.0556
72	9.4109589041	0.0548
73	9.4324324324	0.0541
74	9.4	0.0533
75	9.4210526316	0.0526
76	9.4415584416	0.0519
77	9.4102564103	0.0513
78	9.4303797468	0.0506
79	9.4	0.05

9.4 +- 0.05

Метод Фибоначчи

1	10.05572809
2	9.11145618
3	9.695048315
4	9.33436854
5	9.5572809
6	9.419513485
7	9.38699101
8	9.43961348
9	9.407091005
10	9.427191
11	9.431935965
12	9.42425845
13	9.422446035
14	9.4253785849
15	9.4235661699
16	9.4246863048
17	9.4249507301
18	9.4245228753
19	9.4247873006
20	9.4248497344
21	9.4247487386
22	9.4248111723
23	9.4247726103
24	9.4247634289
25	9.4247781192
26	9.4247817917
27	9.4247762828
28	9.4247799552
29	9.4247781188
30	9.4247762832

9.43961348

4. Вывод

В данной лабораторной работе был найден минимум унимодальной функции с помощью метода оптимального пассивного поиска и метода Фибоначчи.

Исходный код программы.

```
#include <iostream>
#include <cmath>
#include <iomanip>
#include <vector>
#include <algorithm>

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

#define epc 0.1

double F(const double& x) {
	return (cos(x) * tanh(x));
}

std::vector<std::pair<double, double>> optpas(const double& a, const double& b) {
```

```
	std::vector<std::pair<double, double>> val;
	size_t N = 1;
	double del = (b - a) / (N + 1);
	double miny;
	while (del > epc / 2) {
		std::vector<double> vecty;
		del = (b - a) / (N + 1);
		for (size_t k = 1; k <= N; ++k) {
			vecty.push_back(F((b - a) / (N + 1) * k + a));
		}
		size_t ymik = std::min_element(vecty.begin(), vecty.end()) - vecty.begin() + 1;
		miny = (b - a) / (N + 1) * ymik + a;
		val.push_back({ miny, del });
		N++;
	}
	return val;
}

unsigned int fibbon(const size_t& n) {
	if (n < 1)
		return 0;
	unsigned int n1 = 0, n2 = 1, nn = 0;
	for (size_t i = 1; i < n; ++i) {
		nn = n1 + n2;
		n1 = n2;
		n2 = nn;
	}
	return nn;
}
```

```
std::vector<double> fibbon(size_t N, double& a, double& b) {
	std::vector<double> val;
	double  x1 = a + (b - a) * fibbon(N) / fibbon(N + 2);
	double  x2 = a + b - x1;
	double  y1 = F(x1);
	double  y2 = F(x2);
	while (N--) {
		if (y1 > y2) {
			a = x1;
			x1 = x2;
			x2 = b - (x1 - a);
			y1 = y2;
			y2 = F(x2);
			val.push_back(x2);
		}
		else {
			b = x2;
			x2 = x1;
			x1 = a + (b - x2);
			y2 = y1;
			y1 = F(x1);
			val.push_back(x1);
		}
	}
	return val;
}

void PrintValues(const std::vector<double>& val) {
	cout << std::string(40, '-') << endl;
	cout << std::setw(18) << std::left << "| znach N " << "|";
	cout << std::setw(19) << std::left << " znach x |" << endl;
```

```
	cout << std::string(40, '-') << endl;


	for (size_t i = 0; i < val.size(); i++) {
		cout << "|";
		cout << std::setw(10) << std::right << i + 1;
		cout << std::setw(10) << "|";
		cout << std::setw(15) << std::right << std::setprecision(11) << val[i];
		cout << std::setw(2) << "|" << endl;
	}
	cout << std::string(40, '-') << endl;
	cout << val[7] << endl;
}

void PrintValues(const std::vector<std::pair<double, double>>& val) {
	cout << std::string(60, '-') << endl;
	cout << std::setw(20) << std::left << "| koll N " << "|";
	cout << std::setw(20) << std::left << " fib met of x |";
	cout << std::setw(10) << std::left << " pogr |" << endl;
	cout << std::string(60, '-') << endl;

	int i = 0;
	double an1, an2;
	for (auto const& val : val)
	{
		cout << "|";
		cout << std::setw(10) << std::right << i + 1;
		cout << std::setw(10) << "|";
		cout << std::setw(15) << std::setprecision(11) << val.first << std::setw(6) << "|";
		cout << std::setw(10) << std::setprecision(3) << val.second;
		cout << std::setw(5) << "|" << endl;
		i++;
```

```
            an1 = val.first;
            an2 = val.second;
        }
        cout << std::string(60, '-') << endl;
        cout << an1 << " +- " << an2 << endl;
}
int main() {
        //setlocale(LC_ALL, "Russian");

        size_t N = 30;
        double a = 7.0, b = 11.0;

        cout << "Ôóíêöèÿ cos(x)* tanh(x) äëÿ èíòåðâàëà [7, 11]" << endl << endl;
        cout << "Ìåòîä ïîèñêîâîãî ïàññèâíîãî ïîèñêà" << endl;
        cout << "Äëÿ ïîãðåøíîñòè 0,1" << endl;
        PrintValues(optpas(a, b));
        cout << endl;
        cout << "Ìåòîä Ôèáîíà÷÷è" << endl;
        cout << "Ïðè N=30 äëÿ òî÷íîñòè 0,1 äîñòàòî÷íî 8 èòåðàöèé" << endl;
        PrintValues(fibbon(N, a, b));
        return 0;
}
```

Контрольный вопрос

В чем состоит сущность метода оптимального пассивного поиска?

Оптимально пассивным поиском называют минимальный метод поиска, в котором информация о значениях функции, вычисленных в предшествующих точках, не может быть использована.

Сущность метода - деление интервала на равные отрезки и исключении неудовлетворяющих условию отрезков.