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 № 3
по дисциплине «Теория систем и системный анализ»

Тема: «Исследование алгоритма имитации отжига**»**

Вариант 12

Выполнила: Николаева Е.Д.,
студентка группы ИУ8-31

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

г. Москва, 2020 г.

Цель работы

Изучение метода имитации отжига для поиска экстремума на примере унимодальной и мультимодальной функций одного переменного.

Условие задачи

1. На интервале [7; 11] задана унимодальная функция одного переменного f(x) = cos(x)*th(x). Используя метод имитации отжига осуществить поиск минимума f(x).

2. При аналогичных исходных условиях осуществить поиск минимума f(x), модулированной сигналом sin(5x), т.е. мультимодальной функции f(x)*sin(5x).

Графики заданных функций

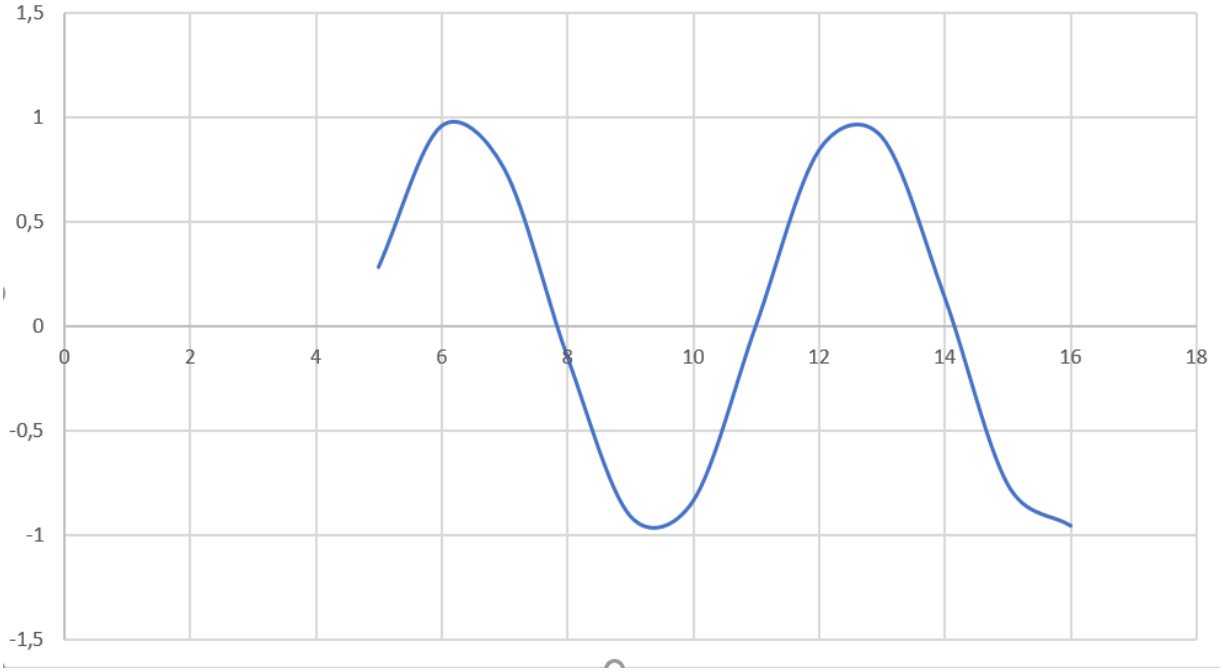

Рисунок 1 – График функции f(x) = cos(x)*th(x) на отрезке [7,11]

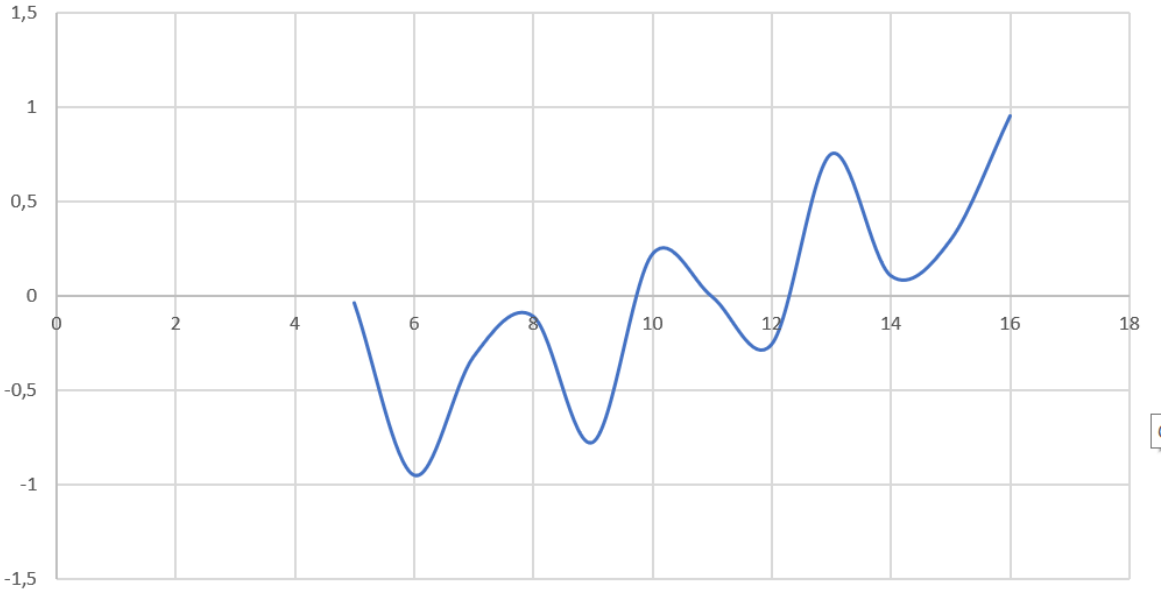

Рисунок 2 – График функции f(x) = cos(x)*th(x)*sin(5x) на отрезке [7; 11]

Имитация отжига для заданных функций

Таблица, которую выводит программа для метода имитации отжига для y = cos(x)*th(x)

N	T	x	f(x)
1	10000	9.25434	-0.91113
2	9500	9.25434	-0.91113
3	9025	9.34004	-0.91113
4	8573.75	9.34004	-0.91113
5	8145.06	9.34004	-0.91113
6	7737.81	9.98642	-0.91113
7	7350.92	9.98642	-0.91113
8	6983.37	9.84201	-0.91113
9	6634.2	9.05414	-0.91113
10	6302.49	9.05414	-0.91113
11	5987.37	9.05414	-0.91113
12	5688	9.05414	-0.91113
13	5403.6	9.05414	-0.91113
14	5133.42	9.05414	-0.91113
15	4876.75	9.05414	-0.91113
16	4632.91	9.12665	-0.91113
17	4401.27	9.28474	-0.91113
18	4181.2	9.40706	-0.91113
19	3972.14	9.42866	-0.91113
20	3773.54	9.42866	-0.91113
21	3584.86	9.42866	-0.91113
22	3405.62	9.42866	-0.91113
23	3235.34	9.42866	-0.91113
24	3073.57	9.07953	-0.91113
25	2919.89	9.07953	-0.91113
26	2773.9	9.9067	-0.91113
27	2635.2	9.9067	-0.91113
28	2503.44	9.15741	-0.91113
29	2378.27	9.15741	-0.91113
30	2259.36	9.15741	-0.91113
31	2146.39	9.15741	-0.91113

32	2039.07	9.15741	-0.91113
33	1937.11	9.15741	-0.91113
34	1840.26	9.15741	-0.91113
35	1748.25	9.15741	-0.91113
36	1660.83	9.15741	-0.91113
37	1577.79	9.15741	-0.91113
38	1498.9	9.15741	-0.91113
39	1423.96	9.15741	-0.91113
40	1352.76	9.15741	-0.91113
41	1285.12	9.76473	-0.91113
42	1220.87	9.76473	-0.91113
43	1159.82	9.76473	-0.91113
44	1101.83	9.97494	-0.91113
45	1046.74	9.97494	-0.91113
46	994.403	9.97494	-0.91113
47	944.682	9.97494	-0.91113
48	897.448	9.97494	-0.91113
49	852.576	9.43587	-0.91113
50	809.947	9.28962	-0.91113
51	769.45	9.28962	-0.91113
52	730.977	9.28962	-0.91113
53	694.428	9.28962	-0.91113
54	659.707	9.28962	-0.91113
55	626.722	9.28962	-0.91113
56	595.386	9.62923	-0.91113
57	565.616	9.62923	-0.91113
58	537.335	9.79904	-0.91113
59	510.469	9.01923	-0.91113
60	484.945	9.01923	-0.91113
61	460.698	9.01923	-0.91113
62	437.663	9.77157	-0.91113
63	415.78	9.77157	-0.91113
64	394.991	9.77157	-0.91113
65	375.241	9.73275	-0.91113
66	356.479	9.73275	-0.91113
67	338.655	9.73275	-0.91113
68	321.723	9.73275	-0.91113
69	305.636	9.73275	-0.91113
70	290.355	9.73275	-0.91113
71	275.837	9.92807	-0.91113
72	262.045	9.92807	-0.91113
73	248.943	9.27497	-0.91113
74	236.496	9.72897	-0.91113
75	224.671	9.72897	-0.91113
76	213.437	9.72897	-0.91113
77	202.765	9.72897	-0.91113
78	192.627	9.72897	-0.91113
79	182.996	9.65194	-0.91113
80	173.846	9.65194	-0.91113
81	165.154	9.65194	-0.91113
82	156.896	9.19819	-0.91113
83	149.051	9.19819	-0.91113
84	141.599	9.19819	-0.91113
85	134.519	9.19819	-0.91113
86	127.793	9.19819	-0.91113
87	121.403	9.19819	-0.91113
88	115.333	9.26911	-0.91113
89	109.566	9.26911	-0.91113
90	104.088	9.26911	-0.91113

91	98.8836	9.0036	-0.91113
92	93.9395	9.0036	-0.91113
93	89.2425	9.0036	-0.91113
94	84.7804	9.0036	-0.91113
95	80.5413	9.0036	-0.91113
96	76.5143	9.50703	-0.91113
97	72.6886	9.63045	-0.91113
98	69.0541	9.63045	-0.91113
99	65.6014	9.63045	-0.91113
100	62.3214	9.9725	-0.91113
101	59.2053	9.9725	-0.91113
102	56.245	9.9725	-0.91113
103	53.4328	9.9725	-0.91113
104	50.7611	9.9725	-0.91113
105	48.2231	9.40095	-0.91113
106	45.8119	9.98886	-0.91113
107	43.5213	9.98886	-0.91113
108	41.3453	9.98886	-0.91113
109	39.278	9.98886	-0.91113
110	37.3141	9.98886	-0.91113
111	35.4484	9.21284	-0.91113
112	33.676	9.21284	-0.91113
113	31.9922	9.21284	-0.91113
114	30.3926	9.74459	-0.91113
115	28.8729	9.17377	-0.91113
116	27.4293	9.17377	-0.91113
117	26.0578	9.17377	-0.91113
118	24.7549	9.17377	-0.91113
119	23.5172	9.17377	-0.91113
120	22.3413	9.13031	-0.91113
121	21.2243	9.51473	-0.91113
122	20.1631	9.51473	-0.91113
123	19.1549	9.51473	-0.91113
124	18.1972	9.51473	-0.91113
125	17.2873	9.51473	-0.91113
126	16.4229	9.51473	-0.91113
127	15.6018	9.51473	-0.91113
128	14.8217	9.50069	-0.91113
129	14.0806	9.50069	-0.91113
130	13.3766	9.50069	-0.91113
131	12.7078	9.50069	-0.91113
132	12.0724	9.50069	-0.91113
133	11.4687	9.50069	-0.91113
134	10.8953	9.51228	-0.91113
135	10.3505	9.41658	-0.91113
136	9.83302	9.41658	-0.91113
137	9.34136	9.39131	-0.91113
138	8.8743	9.5389	-0.91113
139	8.43058	9.5389	-0.91113
140	8.00905	9.4991	-0.91113
141	7.6086	9.88363	-0.91113
142	7.22817	9.26301	-0.91113
143	6.86676	9.26301	-0.91113
144	6.52342	9.95163	-0.91113
145	6.19725	9.22053	-0.91113
146	5.88739	9.22053	-0.91113
147	5.59302	9.22053	-0.91113
148	5.31337	9.79403	-0.91113
149	5.0477	9.33845	-0.91113

150	4.79532	9.33845	-0.91113
151	4.55555	9.33845	-0.91113
152	4.32777	9.44819	-0.91113
153	4.11138	9.48152	-0.91113
154	3.90581	9.76449	-0.91113
155	3.71052	9.76449	-0.91113
156	3.525	9.30415	-0.91113
157	3.34875	9.30415	-0.91113
158	3.18131	9.45882	-0.91113
159	3.02224	9.91073	-0.91113
160	2.87113	9.91073	-0.91113
161	2.72758	9.91073	-0.91113
162	2.5912	9.2768	-0.91113
163	2.46164	9.2768	-0.91113
164	2.33856	9.2768	-0.91113
165	2.22163	9.2768	-0.91113
166	2.11055	9.2768	-0.91113
167	2.00502	9.2768	-0.91113
168	1.90477	9.56392	-0.91113
169	1.80953	9.56392	-0.91113
170	1.71905	9.56392	-0.91113
171	1.6331	9.56392	-0.91113
172	1.55145	9.00482	-0.91113
173	1.47387	9.00482	-0.91113
174	1.40018	9.95199	-0.91113
175	1.33017	9.95199	-0.91113
176	1.26366	9.95199	-0.91113
177	1.20048	9.90011	-0.91113
178	1.14045	9.90011	-0.91113
179	1.08343	9.90011	-0.91113
180	1.02926	9.90011	-0.91113
181	0.977798	9.43855	-0.91113
182	0.928908	9.43855	-0.91113
183	0.882462	9.43855	-0.91113
184	0.838339	9.43855	-0.91113
185	0.796422	9.0202	-0.91113
186	0.756601	9.0202	-0.91113
187	0.718771	9.0202	-0.91113
188	0.682833	9.6512	-0.91113
189	0.648691	9.6512	-0.91113
190	0.616256	9.6512	-0.91113
191	0.585444	9.6512	-0.91113
192	0.556171	9.6512	-0.91113
193	0.528363	9.1286	-0.91113
194	0.501945	9.42512	-0.91113
195	0.476847	9.42512	-0.91113
196	0.453005	9.83151	-0.91113
197	0.430355	9.83151	-0.91113
198	0.408837	9.83151	-0.91113
199	0.388395	9.98935	-0.91113
200	0.368975	9.98935	-0.91113
201	0.350527	9.98935	-0.91113
202	0.333	9.98935	-0.91113
203	0.31635	9.52339	-0.91113
204	0.300533	9.52339	-0.91113
205	0.285506	9.52339	-0.91113
206	0.271231	9.52339	-0.91113
207	0.257669	9.52339	-0.91113
208	0.244786	9.52339	-0.91113

209	0.232547	9.52339	-0.91113
210	0.220919	9.52339	-0.91113
211	0.209873	9.88656	-0.91113
212	0.19938	9.88656	-0.91113
213	0.189411	9.88656	-0.91113
214	0.17994	9.46858	-0.91113
215	0.170943	9.46858	-0.91113
216	0.162396	9.46858	-0.91113
217	0.154276	9.46858	-0.91113
218	0.146562	9.46858	-0.91113
219	0.139234	9.46858	-0.91113
220	0.132272	9.10614	-0.91113
221	0.125659	9.44441	-0.91113
222	0.119376	9.44441	-0.91113
223	0.113407	9.44441	-0.91113
224	0.107737	9.44441	-0.91113
225	0.10235	9.44441	-0.91113
226	0.0972324	9.44441	-0.91113
227	0.0923708	9.2934	-0.91113
228	0.0877523	9.2934	-0.91113
229	0.0833647	9.2934	-0.91113
230	0.0791964	9.2934	-0.91113
231	0.0752366	9.2934	-0.91113
232	0.0714748	9.2934	-0.91113
233	0.067901	9.2934	-0.91113
234	0.064506	9.46089	-0.91113
235	0.0612807	9.46089	-0.91113
236	0.0582167	9.46089	-0.91113
237	0.0553058	9.46089	-0.91113
238	0.0525405	9.16962	-0.91113
239	0.0499135	9.16962	-0.91113
240	0.0474178	9.16962	-0.91113
241	0.0450469	9.16962	-0.91113
242	0.0427946	9.16962	-0.91113
243	0.0406549	9.16962	-0.91113
244	0.0386221	9.16962	-0.91113
245	0.036691	9.16962	-0.91113
246	0.0348565	9.16962	-0.91113
247	0.0331136	9.16962	-0.91113
248	0.031458	9.16962	-0.91113
249	0.0298851	9.16962	-0.91113
250	0.0283908	9.16962	-0.91113
251	0.0269713	9.16962	-0.91113
252	0.0256227	9.16962	-0.91113
253	0.0243416	9.16962	-0.91113
254	0.0231245	9.16962	-0.91113
255	0.0219683	9.16962	-0.91113
256	0.0208699	9.16962	-0.91113
257	0.0198264	9.16962	-0.91113
258	0.018835	9.16962	-0.91113
259	0.0178933	9.16962	-0.91113
260	0.0169986	9.16962	-0.91113
261	0.0161487	9.16962	-0.91113
262	0.0153413	9.16962	-0.91113
263	0.0145742	9.63973	-0.91113
264	0.0138455	9.63973	-0.91113
265	0.0131532	9.63973	-0.91113
266	0.0124956	9.76595	-0.91113
267	0.0118708	9.76595	-0.91113

268	0.0112772	9.74252	-0.91113
269	0.0107134	9.74252	-0.91113
270	0.0101777	9.91159	-0.91113

Result: Xmin = 9.91159 Fmin = -0.91113

Таблица, которую выводит программа для метода имитации отжига для y = cos(x)*th(x)*sin(5x)

N	T	x	f(x)
1	10000	7.12427	-0.322807
2	9500	7.12427	-0.322807
3	9025	9.82626	-0.775284
4	8573.75	9.82626	-0.775284
5	8145.06	9.82626	-0.775284
6	7737.81	9.82626	-0.775284
7	7350.92	9.82626	-0.775284
8	6983.37	9.82626	-0.775284
9	6634.2	9.82626	-0.775284
10	6302.49	9.82626	-0.775284
11	5987.37	9.82626	-0.775284
12	5688	9.82626	-0.775284
13	5403.6	9.82626	-0.775284
14	5133.42	9.82626	-0.775284
15	4876.75	9.82626	-0.775284
16	4632.91	9.82626	-0.775284
17	4401.27	9.82626	-0.775284
18	4181.2	9.82626	-0.775284
19	3972.14	9.82626	-0.775284
20	3773.54	9.82626	-0.775284
21	3584.86	9.82626	-0.775284
22	3405.62	9.82626	-0.775284
23	3235.34	9.82626	-0.775284
24	3073.57	9.82626	-0.775284
25	2919.89	9.82626	-0.775284
26	2773.9	9.58223	-0.775284
27	2635.2	9.62703	-0.775284
28	2503.44	9.62703	-0.775284
29	2378.27	9.62703	-0.775284
30	2259.36	9.62703	-0.775284
31	2146.39	9.27815	-0.775284
32	2039.07	9.27815	-0.775284
33	1937.11	9.1623	-0.775284
34	1840.26	9.31428	-0.775284
35	1748.25	9.14582	-0.775284
36	1660.83	9.14582	-0.775284
37	1577.79	9.14582	-0.775284
38	1498.9	9.14582	-0.775284
39	1423.96	9.14582	-0.775284
40	1352.76	9.36299	-0.775284
41	1285.12	9.36299	-0.775284
42	1220.87	9.36299	-0.775284
43	1159.82	9.36299	-0.775284
44	1101.83	9.78939	-0.775284
45	1046.74	9.78939	-0.775284
46	994.403	9.78939	-0.775284

47	944.682	9.36885	-0.775284
48	897.448	9.36885	-0.775284
49	852.576	9.36885	-0.775284
50	809.947	9.36885	-0.775284
51	769.45	9.36885	-0.775284
52	730.977	9.36885	-0.775284
53	694.428	9.83407	-0.775284
54	659.707	9.83407	-0.775284
55	626.722	9.83407	-0.775284
56	595.386	9.83407	-0.775284
57	565.616	9.91379	-0.775284
58	537.335	9.91379	-0.775284
59	510.469	9.91379	-0.775284
60	484.945	9.91379	-0.775284
61	460.698	9.2823	-0.775284
62	437.663	9.2823	-0.775284
63	415.78	9.98141	-0.775284
64	394.991	9.94552	-0.775284
65	375.241	9.95773	-0.775284
66	356.479	9.95773	-0.775284
67	338.655	9.95773	-0.775284
68	321.723	9.95773	-0.775284
69	305.636	9.07062	-0.775284
70	290.355	9.07062	-0.775284
71	275.837	9.07062	-0.775284
72	262.045	9.61934	-0.775284
73	248.943	9.61934	-0.775284
74	236.496	9.61934	-0.775284
75	224.671	9.61934	-0.775284
76	213.437	9.61934	-0.775284
77	202.765	9.78915	-0.775284
78	192.627	9.78915	-0.775284
79	182.996	9.78915	-0.775284
80	173.846	9.89975	-0.775284
81	165.154	9.89975	-0.775284
82	156.896	9.46284	-0.775284
83	149.051	9.61959	-0.775284
84	141.599	9.17597	-0.775284
85	134.519	9.17597	-0.775284
86	127.793	9.17597	-0.775284
87	121.403	9.17597	-0.775284
88	115.333	9.91147	-0.775284
89	109.566	9.12384	-0.775284
90	104.088	9.71969	-0.775284
91	98.8836	9.7712	-0.775284
92	93.9395	9.7712	-0.775284
93	89.2425	9.7712	-0.775284
94	84.7804	9.7712	-0.775284
95	80.5413	9.7712	-0.775284
96	76.5143	9.7712	-0.775284
97	72.6886	9.7712	-0.775284
98	69.0541	9.7712	-0.775284
99	65.6014	9.42378	-0.775284
100	62.3214	9.42378	-0.775284
101	59.2053	9.42378	-0.775284
102	56.245	9.42378	-0.775284
103	53.4328	9.42378	-0.775284
104	50.7611	9.25141	-0.775284
105	48.2231	9.25141	-0.775284

106	45.8119	9.25141	-0.775284
107	43.5213	9.25141	-0.775284
108	41.3453	9.25141	-0.775284
109	39.278	9.25141	-0.775284
110	37.3141	9.25141	-0.775284
111	35.4484	9.25141	-0.775284
112	33.676	9.44636	-0.775284
113	31.9922	9.44636	-0.775284
114	30.3926	9.33039	-0.775284
115	28.8729	9.38008	-0.775284
116	27.4293	9.38008	-0.775284
117	26.0578	9.38008	-0.775284
118	24.7549	9.35774	-0.775284
119	23.5172	9.2768	-0.775284
120	22.3413	9.2768	-0.775284
121	21.2243	9.2768	-0.775284
122	20.1631	9.2768	-0.775284
123	19.1549	9.2768	-0.775284
124	18.1972	9.2768	-0.775284
125	17.2873	9.2768	-0.775284
126	16.4229	9.5157	-0.775284
127	15.6018	9.5157	-0.775284
128	14.8217	9.14228	-0.775284
129	14.0806	9.14228	-0.775284
130	13.3766	9.14228	-0.775284
131	12.7078	9.14228	-0.775284
132	12.0724	9.14228	-0.775284
133	11.4687	9.14228	-0.775284
134	10.8953	9.14228	-0.775284
135	10.3505	9.14228	-0.775284
136	9.83302	9.9288	-0.775284
137	9.34136	9.9288	-0.775284
138	8.8743	9.85556	-0.775284
139	8.43058	9.85556	-0.775284
140	8.00905	9.85556	-0.775284
141	7.6086	9.85556	-0.775284
142	7.22817	9.29328	-0.775284
143	6.86676	9.29328	-0.775284
144	6.52342	9.29328	-0.775284
145	6.19725	9.29328	-0.775284
146	5.88739	9.59053	-0.775284
147	5.59302	9.76656	-0.775284
148	5.31337	9.76656	-0.775284
149	5.0477	9.76656	-0.775284
150	4.79532	9.76656	-0.775284
151	4.55555	9.76656	-0.775284
152	4.32777	9.76656	-0.775284
153	4.11138	9.76656	-0.775284
154	3.90581	9.76656	-0.775284
155	3.71052	9.76656	-0.775284
156	3.525	9.76656	-0.775284
157	3.34875	9.76656	-0.775284
158	3.18131	9.57552	-0.775284
159	3.02224	9.57552	-0.775284
160	2.87113	9.57552	-0.775284
161	2.72758	9.18073	-0.775284
162	2.5912	9.79708	-0.775284
163	2.46164	9.79708	-0.775284
164	2.33856	9.79708	-0.775284

165	2.22163	9.2801	-0.775284
166	2.11055	9.2801	-0.775284
167	2.00502	9.2801	-0.775284
168	1.90477	9.2801	-0.775284
169	1.80953	9.2801	-0.775284
170	1.71905	9.83761	-0.775284
171	1.6331	9.83761	-0.775284
172	1.55145	9.83761	-0.775284
173	1.47387	9.83761	-0.775284
174	1.40018	9.9874	-0.775284
175	1.33017	9.9874	-0.775284
176	1.26366	9.9874	-0.775284
177	1.20048	9.9874	-0.775284
178	1.14045	9.9874	-0.775284
179	1.08343	9.9874	-0.775284
180	1.02926	9.9874	-0.775284
181	0.977798	9.9874	-0.775284
182	0.928908	9.9874	-0.775284
183	0.882462	9.9874	-0.775284
184	0.838339	9.73556	-0.775284
185	0.796422	9.73556	-0.775284
186	0.756601	9.73556	-0.775284
187	0.718771	9.73556	-0.775284
188	0.682833	9.73556	-0.775284
189	0.648691	9.73556	-0.775284
190	0.616256	9.54988	-0.775284
191	0.585444	9.54988	-0.775284
192	0.556171	9.54988	-0.775284
193	0.528363	9.61605	-0.775284
194	0.501945	9.61605	-0.775284
195	0.476847	9.61605	-0.775284
196	0.453005	9.53706	-0.775284
197	0.430355	9.20161	-0.775284
198	0.408837	9.20161	-0.775284
199	0.388395	9.20161	-0.775284
200	0.368975	9.20161	-0.775284
201	0.350527	9.20161	-0.775284
202	0.333	9.20161	-0.775284
203	0.31635	9.20161	-0.775284
204	0.300533	9.20161	-0.775284
205	0.285506	9.20161	-0.775284
206	0.271231	9.56661	-0.775284
207	0.257669	9.48482	-0.775284
208	0.244786	9.48482	-0.775284
209	0.232547	9.48482	-0.775284
210	0.220919	9.48482	-0.775284
211	0.209873	9.84945	-0.775284
212	0.19938	9.84945	-0.775284
213	0.189411	9.84945	-0.775284
214	0.17994	9.67928	-0.775284
215	0.170943	9.67928	-0.775284
216	0.162396	9.67928	-0.775284
217	0.154276	9.98215	-0.775284
218	0.146562	9.98215	-0.775284
219	0.139234	9.5179	-0.775284
220	0.132272	9.5179	-0.775284
221	0.125659	9.5179	-0.775284
222	0.119376	9.5179	-0.775284
223	0.113407	9.5179	-0.775284

224	0.107737	9.5179	-0.775284
225	0.10235	9.5179	-0.775284
226	0.0972324	9.5179	-0.775284
227	0.0923708	9.5179	-0.775284
228	0.0877523	9.5179	-0.775284
229	0.0833647	9.17792	-0.775284
230	0.0791964	9.17792	-0.775284
231	0.0752366	9.17792	-0.775284
232	0.0714748	9.40364	-0.775284
233	0.067901	9.40364	-0.775284
234	0.064506	9.64095	-0.775284
235	0.0612807	9.64095	-0.775284
236	0.0582167	9.61434	-0.775284
237	0.0553058	9.61434	-0.775284
238	0.0525405	9.61434	-0.775284
239	0.0499135	9.61434	-0.775284
240	0.0474178	9.03058	-0.775284
241	0.0450469	9.03058	-0.775284
242	0.0427946	9.03058	-0.775284
243	0.0406549	9.03058	-0.775284
244	0.0386221	9.03058	-0.775284
245	0.036691	9.03058	-0.775284
246	0.0348565	9.60237	-0.775284
247	0.0331136	9.05792	-0.775284
248	0.031458	9.93588	-0.775284
249	0.0298851	9.9487	-0.775284
250	0.0283908	9.9487	-0.775284
251	0.0269713	9.9487	-0.775284
252	0.0256227	9.9487	-0.775284
253	0.0243416	9.9487	-0.775284
254	0.0231245	9.9487	-0.775284
255	0.0219683	9.9487	-0.775284
256	0.0208699	9.9487	-0.775284
257	0.0198264	9.9487	-0.775284
258	0.018835	9.9487	-0.775284
259	0.0178933	9.9487	-0.775284
260	0.0169986	9.32038	-0.775284
261	0.0161487	9.76119	-0.775284
262	0.0153413	9.92624	-0.775284
263	0.0145742	9.92624	-0.775284
264	0.0138455	9.92624	-0.775284
265	0.0131532	9.92624	-0.775284
266	0.0124956	9.92624	-0.775284
267	0.0118708	9.92624	-0.775284
268	0.0112772	9.92624	-0.775284
269	0.0107134	9.92624	-0.775284
270	0.0101777	9.92624	-0.775284

```
Result: Xmin = -0.731468, Fmin = -0.46435
```

Выводы

В результате проделанной работы я исследовала метод имитации отжига. Убедилась в том, что алгоритм имитации отжига является эффективным алгоритмом случайного поиска глобального минимума, на примере данной унимодальной и мультимодальной функции одного переменного. Метод эффективно работает для обеих функций.

Приложение. Исходный код программы

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

double func (int x)
{
    double f=cos(x)*tanh(x);
    return f;
}

double funcm (int x)
{
    double fm=cos(x)*tanh(x)*sin(5*x);
    return fm;
}

void print(const int it, const double T,
               const double val, const double f) {
    std::cout << "| " << std::setw(4) << it
         << "| " << std::setw(10) << T
         << "| " << std::setw(12) << val
         << "| " << std::setw(14) << f << "|"<<std::endl;
}


int main()
{
    std::cout << "Simulated Annealing" << std::endl;
    std::cout<<"Min of function f(x) "<<std::endl;
    std::cout << std::left << std::string(50, '-') << '\n'
         << "| " << std::setw(4) << "N"
         << "| " << std::setw(10) << "T"
         << "| " << std::setw(12) << "x"
         << "| " << std::setw(14) << "f(x)" << "|"<<std::endl
         << std::string(50, '-') << std::endl;
    double tmin = 0.01;
    double tmax = 10000.0, xi = 0.0, res = 0.0,a=7,b=11;
    int N = 1;
    double x0 = 0, x = (double)(a + rand() * 1./RAND_MAX * (b - a));
    res = func(x);
    while (tmax > tmin)
    {
        xi =(double)(a + rand() * 1./RAND_MAX * (b - a));
        double df = func(xi) - func(x);
        if (df <= 0)
        {
            x0 = xi;
            x = xi;
            res = func(x0);
        }
        else
        {
            double probability = (double)(a + rand() * 1./RAND_MAX * (b - a));
            double P = exp(-df / tmax);
            if (probability < P)
            {
                x0 = xi;
                x = xi;
                res = func(x0);
            }
        }
```

```
        print(N,tmax ,x0, func(x0) );
        tmax = tmax * 0.95;
        N++;

    }
    std::cout << std::endl << "Result: Xmin = " << x0 << "   Fmin = " << res <<
std::endl;



    std::cout<<"\nMin of function f(x)*sin(5x) "<<std::endl;
    std::cout << std::left << std::string(50, '-') << '\n'
              << "| " << std::setw(4) << "N"
              << "| " << std::setw(10) << "T"
              << "| " << std::setw(12) << "x"
              << "| " << std::setw(14) << "f(x)" << "|"<<std::endl
              << std::string(50, '-') << std::endl;

     tmin = 0.01;
     tmax = 10000.0;
     xi = 0.0;
     res = 0.0;
     a=7;
     b=11;
   N = 1;
     x0 = 0;
     x = (double)(a + rand() * 1./RAND_MAX * (b - a));
    res = funcm(x);
    while (tmax > tmin)
    {
        xi = (double)(a + rand() * 1./RAND_MAX * (b - a));
        double df = funcm(xi) - funcm(x);
        if (df <= 0)
        {
            x0 = xi;
            x = xi;
            res = funcm(x0);
        }
        else
        {
            double probability = (double)(a + rand() * 1./RAND_MAX * (b - a));
            double P = exp(-df / tmax);
            if (probability < P)
            {
                x0 = xi;
                x = xi;
                res = funcm(x0);
            }
        }

        print(N,tmax ,x0, funcm(x0) );
        tmax = tmax * 0.95;
        N++;

    }
    std::cout << std::endl << "Result: Xmin = " << x0 << "   Fmin = " << res <<
std::endl;


    return 0;
}
```

Ответ на контрольный вопрос

В чем состоит сущность метода имитации отжига? Какова область применимости данного метода?

Метод имитации отжига основан на том, что локальное (субоптимальное) решение, найденное в процессе решения задачи оптимизации, также можно рассматривать как дефектное решение. Улучшить это решение (приблизиться к глобальному оптимуму) можно путём его случайных флюктуаций, амплитуда которых уменьшается с ростом номера итераций. Принципиальным в алгоритме SA является то, что, в отличие от большинства других стохастических алгоритмов поисковой оптимизации, он допускает шаги, приводящие к увеличению значений фитнес-функции. Метод имитации отжига применяется для решения разных оптимизационных задач – финансовых, компьютерной графики, комбинаторных, и т.д., используется в нейронных сетях.