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)

КАФЕДРА «Информационная безопасность» (ИУ8)

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

Вариант 12

Выполнил: Мишков А.О.
студент группы ИУ8-32

Проверил:

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

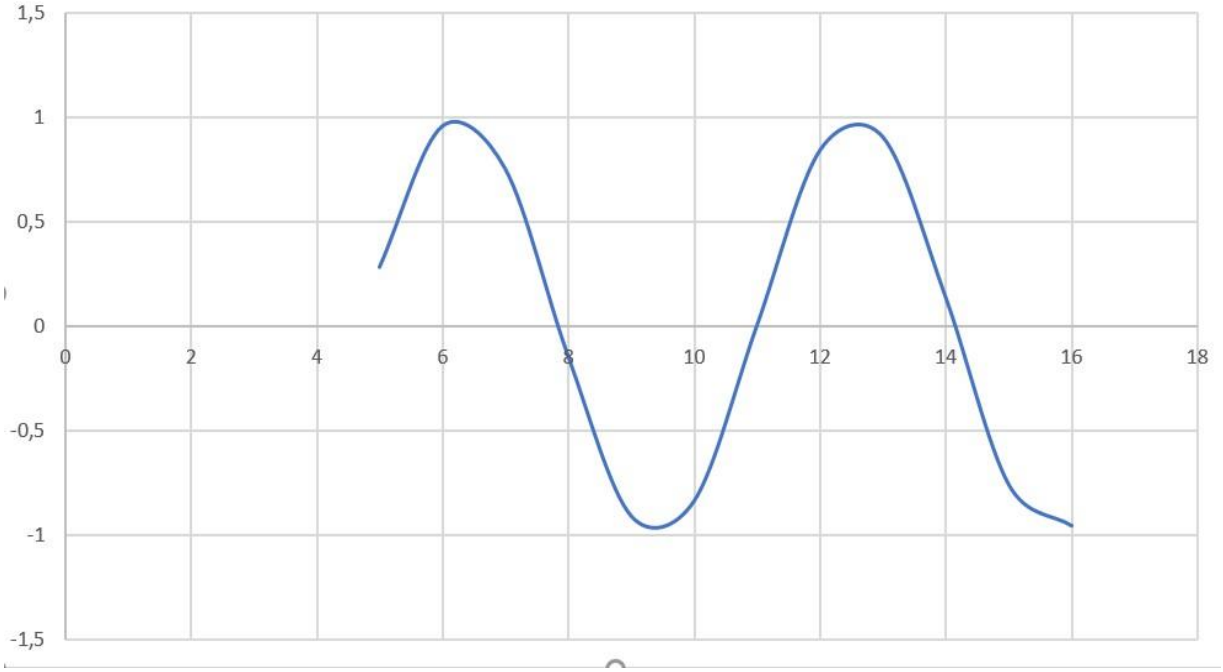


Рисунок 1 – График функции f(x) = cos(x)*th(x) на отрезке [7,11]

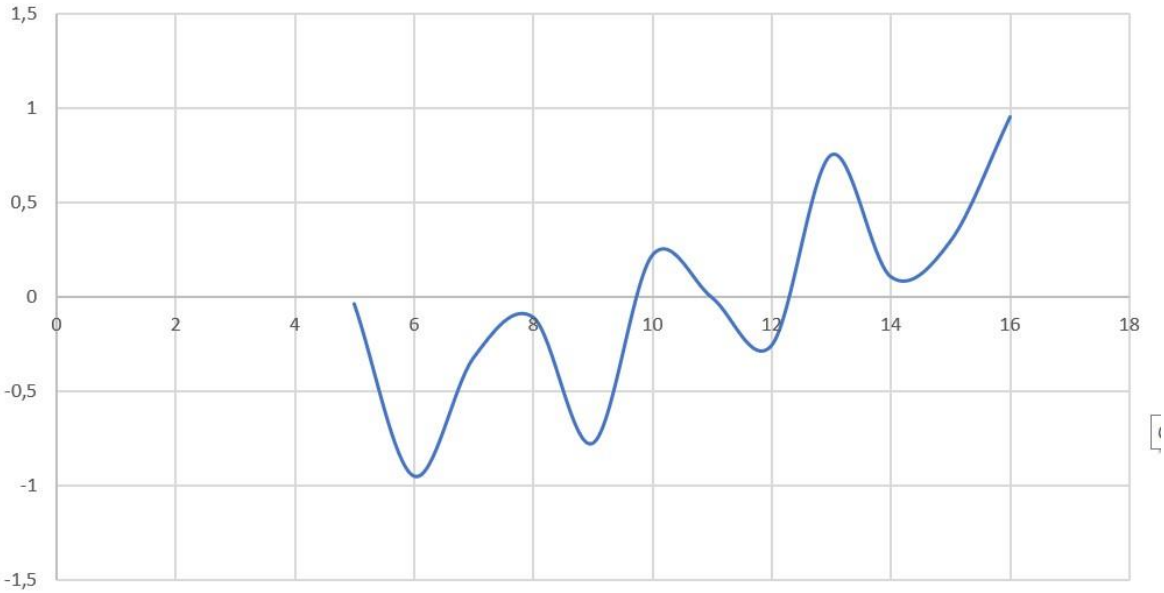


Рисунок 2 – График функции f(x) = cos(x)*th(x)*sin(5x) на отрезке [7; 11]

Имитация отжига для заданных функций

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

Ecstremum f(x)

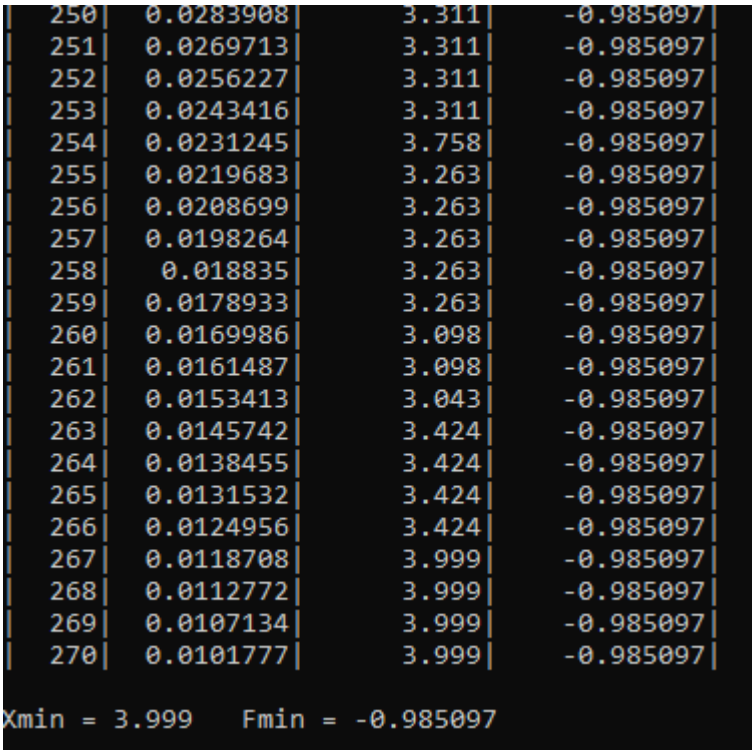
N	T	x	f(x)
1	10000	2.46	-0.401177
2	9500	2.832	-0.401177
3	9025	2.99	-0.401177
4	8573.75	3.162	-0.985097
5	8145.06	3.162	-0.985097
6	7737.81	3.347	-0.985097
7	7350.92	3.452	-0.985097
8	6983.37	3.455	-0.985097
9	6634.2	3.455	-0.985097
10	6302.49	3.455	-0.985097
11	5987.37	3.958	-0.985097
12	5688	3.958	-0.985097
13	5403.6	3.438	-0.985097
14	5133.42	3.826	-0.985097
15	4876.75	3.826	-0.985097
16	4632.91	3.599	-0.985097
17	4401.27	3.599	-0.985097
18	4181.2	3.599	-0.985097
19	3972.14	3.915	-0.985097
20	3773.54	3.915	-0.985097
21	3584.86	3.888	-0.985097
22	3405.62	3.888	-0.985097
23	3235.34	3.766	-0.985097
24	3073.57	3.034	-0.985097
25	2919.89	3.369	-0.985097
26	2773.9	3.905	-0.985097
27	2635.2	3.905	-0.985097
28	2503.44	3.529	-0.985097
29	2378.27	3.891	-0.985097
30	2259.36	3.891	-0.985097
31	2146.39	3.891	-0.985097
32	2039.07	3.891	-0.985097
33	1937.11	3.891	-0.985097
34	1840.26	3.891	-0.985097
35	1748.25	3.891	-0.985097
36	1660.83	3.891	-0.985097
37	1577.79	3.891	-0.985097
38	1498.9	3.891	-0.985097
39	1423.96	3.891	-0.985097
40	1352.76	3.812	-0.985097
41	1285.12	3.812	-0.985097
42	1220.87	3.342	-0.985097
43	1159.82	3.342	-0.985097
44	1101.83	3.037	-0.985097
45	1046.74	3.037	-0.985097
46	994.403	3.037	-0.985097
47	944.682	3.037	-0.985097
48	897.448	3.227	-0.985097
49	852.576	3.361	-0.985097
50	809.947	3.361	-0.985097
51	769.45	3.361	-0.985097
52	730.977	3.361	-0.985097
53	694.428	3.361	-0.985097
54	659.707	3.947	-0.985097
55	626.722	3.947	-0.985097
56	595.386	3.965	-0.985097
57	565.616	3.874	-0.985097
58	537.335	3.874	-0.985097
59	510.469	3.438	-0.985097
60	484.945	3.927	-0.985097

Microsoft Visual Studio Debug Console

61	460.698	3.617	-0.985097
62	437.663	3.617	-0.985097
63	415.78	3.03	-0.985097
64	394.991	3.03	-0.985097
65	375.241	3.03	-0.985097
66	356.479	3.832	-0.985097
67	338.655	3.832	-0.985097
68	321.723	3.647	-0.985097
69	305.636	3.647	-0.985097
70	290.355	3.647	-0.985097
71	275.837	3.161	-0.985097
72	262.045	3.878	-0.985097
73	248.943	3.878	-0.985097
74	236.496	3.414	-0.985097
75	224.671	3.065	-0.985097
76	213.437	3.065	-0.985097
77	202.765	3.267	-0.985097
78	192.627	3.267	-0.985097
79	182.996	3.006	-0.985097
80	173.846	3.006	-0.985097
81	165.154	3.158	-0.985097
82	156.896	3.158	-0.985097
83	149.051	3.758	-0.985097
84	141.599	3.758	-0.985097
85	134.519	3.03	-0.985097
86	127.793	3.548	-0.985097
87	121.403	3.84	-0.985097
88	115.333	3.84	-0.985097
89	109.566	3.216	-0.985097
90	104.088	3.216	-0.985097
91	98.8836	3.216	-0.985097
92	93.9395	3.216	-0.985097
93	89.2425	3.216	-0.985097
94	84.7804	3.216	-0.985097
95	80.5413	3.94	-0.985097
96	76.5143	3.94	-0.985097
97	72.6886	3.755	-0.985097
98	69.0541	3.212	-0.985097
99	65.6014	3.212	-0.985097
100	62.3214	3.937	-0.985097
101	59.2053	3.996	-0.985097
102	56.245	3.996	-0.985097
103	53.4328	3.157	-0.985097
104	50.7611	3.157	-0.985097
105	48.2231	3.157	-0.985097
106	45.8119	3.904	-0.985097
107	43.5213	3.904	-0.985097
108	41.3453	3.904	-0.985097
109	39.278	3.904	-0.985097
110	37.3141	3.04	-0.985097
111	35.4484	3.04	-0.985097
112	33.676	3.04	-0.985097
113	31.9922	3.371	-0.985097
114	30.3926	3.371	-0.985097
115	28.8729	3.012	-0.985097
116	27.4293	3.838	-0.985097
117	26.0578	3.838	-0.985097
118	24.7549	3.475	-0.985097
119	23.5172	3.475	-0.985097
120	22.3413	3.475	-0.985097
121	21.2243	3.475	-0.985097
122	20.1631	3.398	-0.985097
123	19.1549	3.398	-0.985097

124	18.1972	3.398	-0.985097
125	17.2873	3.888	-0.985097
126	16.4229	3.639	-0.985097
127	15.6018	3.975	-0.985097
128	14.8217	3.975	-0.985097
129	14.0806	3.305	-0.985097
130	13.3766	3.115	-0.985097
131	12.7078	3.115	-0.985097
132	12.0724	3.304	-0.985097
133	11.4687	3.304	-0.985097
134	10.8953	3.945	-0.985097
135	10.3505	3.945	-0.985097
136	9.83302	3.013	-0.985097
137	9.34136	3.044	-0.985097
138	8.8743	3.044	-0.985097
139	8.43058	3.044	-0.985097
140	8.00905	3.044	-0.985097
141	7.6086	3.984	-0.985097
142	7.22817	3.984	-0.985097
143	6.86676	3.984	-0.985097
144	6.52342	3.338	-0.985097
145	6.19725	3.338	-0.985097
146	5.88739	3.303	-0.985097
147	5.59302	3.5	-0.985097
148	5.31337	3.946	-0.985097
149	5.0477	3.946	-0.985097
150	4.79532	3.116	-0.985097
151	4.55555	3.116	-0.985097
152	4.32777	3.793	-0.985097
153	4.11138	3.793	-0.985097
154	3.90581	3.793	-0.985097
155	3.71052	3.647	-0.985097
156	3.525	3.647	-0.985097
157	3.34875	3.647	-0.985097
158	3.18131	3.647	-0.985097
159	3.02224	3.647	-0.985097
160	2.87113	3.647	-0.985097
161	2.72758	3.808	-0.985097
162	2.5912	3.808	-0.985097
163	2.46164	3.007	-0.985097
164	2.33856	3.007	-0.985097
165	2.22163	3.263	-0.985097
166	2.11055	3.263	-0.985097
167	2.00502	3.263	-0.985097
168	1.90477	3.885	-0.985097
169	1.80953	3.885	-0.985097
170	1.71905	3.858	-0.985097
171	1.6331	3.858	-0.985097
172	1.55145	3.858	-0.985097
173	1.47387	3.486	-0.985097
174	1.40018	3.486	-0.985097
175	1.33017	3.486	-0.985097
176	1.26366	3.486	-0.985097
177	1.20048	3.486	-0.985097
178	1.14045	3.486	-0.985097
179	1.08343	3.486	-0.985097
180	1.02926	3.486	-0.985097
181	0.977798	3.486	-0.985097
182	0.928908	3.486	-0.985097
183	0.882462	3.486	-0.985097
184	0.838339	3.486	-0.985097
185	0.796422	3.173	-0.985097
186	0.756601	3.173	-0.985097

187	0.718771	3.173	-0.985097
188	0.682833	3.173	-0.985097
189	0.648691	3.173	-0.985097
190	0.616256	3.654	-0.985097
191	0.585444	3.654	-0.985097
192	0.556171	3.969	-0.985097
193	0.528363	3.969	-0.985097
194	0.501945	3.969	-0.985097
195	0.476847	3.969	-0.985097
196	0.453005	3.043	-0.985097
197	0.430355	3.939	-0.985097
198	0.408837	3.509	-0.985097
199	0.388395	3.509	-0.985097
200	0.368975	3.21	-0.985097
201	0.350527	3.21	-0.985097
202	0.333	3.919	-0.985097
203	0.31635	3.919	-0.985097
204	0.300533	3.181	-0.985097
205	0.285506	3.181	-0.985097
206	0.271231	3.931	-0.985097
207	0.257669	3.859	-0.985097
208	0.244786	3.859	-0.985097
209	0.232547	3.859	-0.985097
210	0.220919	3.859	-0.985097
211	0.209873	3.859	-0.985097
212	0.19938	3.859	-0.985097
213	0.189411	3.859	-0.985097
214	0.17994	3.197	-0.985097
215	0.170943	3.197	-0.985097
216	0.162396	3.197	-0.985097
217	0.154276	3.312	-0.985097
218	0.146562	3.346	-0.985097
219	0.139234	3.346	-0.985097
220	0.132272	3.346	-0.985097
221	0.125659	3.332	-0.985097
222	0.119376	3.332	-0.985097
223	0.113407	3.332	-0.985097
224	0.107737	3.551	-0.985097
225	0.10235	3.551	-0.985097
226	0.0972324	3.551	-0.985097
227	0.0923708	3.551	-0.985097
228	0.0877523	3.627	-0.985097
229	0.0833647	3.661	-0.985097
230	0.0791964	3.661	-0.985097
231	0.0752366	3.661	-0.985097
232	0.0714748	3.54	-0.985097
233	0.067901	3.151	-0.985097
234	0.064506	3.151	-0.985097
235	0.0612807	3.747	-0.985097
236	0.0582167	3.747	-0.985097
237	0.0553058	3.499	-0.985097
238	0.0525405	3.744	-0.985097
239	0.0499135	3.744	-0.985097
240	0.0474178	3.744	-0.985097
241	0.0450469	3.744	-0.985097
242	0.0427946	3.013	-0.985097
243	0.0406549	3.013	-0.985097
244	0.0386221	3.013	-0.985097
245	0.036691	3.013	-0.985097
246	0.0348565	3.013	-0.985097
247	0.0331136	3.311	-0.985097
248	0.031458	3.311	-0.985097
249	0.0298851	3.311	-0.985097



F(x):Xmin = 9.91159 Fmin = -0.91113
F(x)*sin(5x): Xmin = -0.731468, Fmin = -0.46435

Ecstremum f(x)*sin(5x)

N	T	x	f(x)
1	10000	2.059	0.218249
2	9500	1.897	-0.394589
3	9025	3.18	-0.640596
4	8573.75	3.18	-0.640596
5	8145.06	3.18	-0.640596
6	7737.81	3.159	-0.640596
7	7350.92	3.159	-0.640596
8	6983.37	3.704	-0.640596
9	6634.2	3.704	-0.640596
10	6302.49	3.666	-0.640596
11	5987.37	3.783	-0.640596
12	5688	3.783	-0.640596
13	5403.6	3.783	-0.640596
14	5133.42	3.116	-0.640596
15	4876.75	3.116	-0.640596
16	4632.91	3.066	-0.640596
17	4401.27	3.186	-0.640596
18	4181.2	3.186	-0.640596
19	3972.14	3.865	-0.640596
20	3773.54	3.958	-0.640596
21	3584.86	3.677	-0.640596
22	3405.62	3.677	-0.640596
23	3235.34	3.677	-0.640596
24	3073.57	3.677	-0.640596
25	2919.89	3.677	-0.640596
26	2773.9	3.677	-0.640596
27	2635.2	3.426	-0.640596
28	2503.44	3.007	-0.640596
29	2378.27	3.756	-0.640596
30	2259.36	3.658	-0.640596
31	2146.39	3.658	-0.640596
32	2039.07	3.658	-0.640596
33	1937.11	3.658	-0.640596
34	1840.26	3.658	-0.640596
35	1748.25	3.58	-0.640596
36	1660.83	3.554	-0.640596
37	1577.79	3.554	-0.640596
38	1498.9	3.469	-0.640596
39	1423.96	3.469	-0.640596
40	1352.76	3.469	-0.640596
41	1285.12	3.469	-0.640596
42	1220.87	3.4	-0.640596
43	1159.82	3.957	-0.640596
44	1101.83	3.957	-0.640596
45	1046.74	3.957	-0.640596
46	994.403	3.957	-0.640596
47	944.682	3.957	-0.640596
48	897.448	3.957	-0.640596
49	852.576	3.957	-0.640596
50	809.947	3	-0.640596
51	769.45	3	-0.640596
52	730.977	3	-0.640596
53	694.428	3.012	-0.640596
54	659.707	3.779	-0.640596
55	626.722	3.919	-0.640596
56	595.386	3.919	-0.640596
57	565.616	3.939	-0.640596
58	537.335	3.939	-0.640596
59	510.469	3.939	-0.640596

123	19.1549	3.024	-0.640596
124	18.1972	3.024	-0.640596
125	17.2873	3.46	-0.640596
126	16.4229	3.46	-0.640596
127	15.6018	3.46	-0.640596
128	14.8217	3.135	-0.640596
129	14.0806	3.848	-0.640596
130	13.3766	3.846	-0.640596
131	12.7078	3.846	-0.640596
132	12.0724	3.846	-0.640596
133	11.4687	3.846	-0.640596
134	10.8953	3.252	-0.640596
135	10.3505	3.252	-0.640596
136	9.83302	3.252	-0.640596
137	9.34136	3.31	-0.640596
138	8.8743	3.31	-0.640596
139	8.43058	3.31	-0.640596
140	8.00905	3.31	-0.640596
141	7.6086	3.31	-0.640596
142	7.22817	3.31	-0.640596
143	6.86676	3.31	-0.640596
144	6.52342	3.31	-0.640596
145	6.19725	3.49	-0.640596
146	5.88739	3.49	-0.640596
147	5.59302	3.49	-0.640596
148	5.31337	3.307	-0.640596
149	5.0477	3.307	-0.640596
150	4.79532	3.536	-0.640596
151	4.55555	3.536	-0.640596
152	4.32777	3.536	-0.640596
153	4.11138	3.536	-0.640596
154	3.90581	3.677	-0.640596
155	3.71052	3.677	-0.640596
156	3.525	3.677	-0.640596
157	3.34875	3.281	-0.640596
158	3.18131	3.281	-0.640596
159	3.02224	3.545	-0.640596
160	2.87113	3.545	-0.640596
161	2.72758	3.545	-0.640596
162	2.5912	3.545	-0.640596
163	2.46164	3.545	-0.640596
164	2.33856	3.545	-0.640596
165	2.22163	3.545	-0.640596
166	2.11055	3.545	-0.640596
167	2.00502	3.545	-0.640596
168	1.90477	3.545	-0.640596
169	1.80953	3.545	-0.640596
170	1.71905	3.545	-0.640596
171	1.6331	3.545	-0.640596
172	1.55145	3.545	-0.640596
173	1.47387	3.545	-0.640596
174	1.40018	3.545	-0.640596
175	1.33017	3.545	-0.640596
176	1.26366	3.545	-0.640596
177	1.20048	3.007	-0.640596
178	1.14045	3.663	-0.640596
179	1.08343	3.979	-0.640596
180	1.02926	3.979	-0.640596
181	0.977798	3.979	-0.640596
182	0.928908	3.979	-0.640596
183	0.882462	3.202	-0.640596
184	0.838339	3.306	-0.640596
185	0.796422	3.229	-0.640596

186	0.756601	3.229	-0.640596
187	0.718771	3.974	-0.640596
188	0.682833	3.039	-0.640596
189	0.648691	3.039	-0.640596
190	0.616256	3.039	-0.640596
191	0.585444	3.039	-0.640596
192	0.556171	3.039	-0.640596
193	0.528363	3.039	-0.640596
194	0.501945	3.037	-0.640596
195	0.476847	3.037	-0.640596
196	0.453005	3.037	-0.640596
197	0.430355	3.037	-0.640596
198	0.408837	3.037	-0.640596
199	0.388395	3.037	-0.640596
200	0.368975	3.037	-0.640596
201	0.350527	3.037	-0.640596
202	0.333	3.037	-0.640596
203	0.31635	3.037	-0.640596
204	0.300533	3.463	-0.640596
205	0.285506	3.642	-0.640596
206	0.271231	3.642	-0.640596
207	0.257669	3.752	-0.640596
208	0.244786	3.89	-0.640596
209	0.232547	3.597	-0.640596
210	0.220919	3.418	-0.640596
211	0.209873	3.418	-0.640596
212	0.19938	3.459	-0.640596
213	0.189411	3.01	-0.640596
214	0.17994	3.01	-0.640596
215	0.170943	3.035	-0.640596
216	0.162396	3.035	-0.640596
217	0.154276	3.035	-0.640596
218	0.146562	3.901	-0.640596
219	0.139234	3.901	-0.640596
220	0.132272	3.68	-0.640596
221	0.125659	3.088	-0.640596
222	0.119376	3.088	-0.640596
223	0.113407	3.548	-0.640596
224	0.107737	3.548	-0.640596
225	0.10235	3.943	-0.640596
226	0.0972324	3.943	-0.640596
227	0.0923708	3.943	-0.640596
228	0.0877523	3.943	-0.640596
229	0.0833647	3.943	-0.640596
230	0.0791964	3.557	-0.640596
231	0.0752366	3.535	-0.640596
232	0.0714748	3.535	-0.640596
233	0.067901	3.714	-0.640596
234	0.064506	3.714	-0.640596
235	0.0612807	3.85	-0.640596
236	0.0582167	3.85	-0.640596
237	0.0553058	3.85	-0.640596
238	0.0525405	3.85	-0.640596
239	0.0499135	3.757	-0.640596
240	0.0474178	3.943	-0.640596
241	0.0450469	3.95	-0.640596
242	0.0427946	3.95	-0.640596
243	0.0406549	3.95	-0.640596
244	0.0386221	3.95	-0.640596
245	0.036691	3.95	-0.640596
246	0.0348565	3.95	-0.640596
247	0.0331136	3.95	-0.640596
248	0.031458	3.95	-0.640596

249	0.0298851	3.95	-0.640596
250	0.0283908	3.44	-0.640596
251	0.0269713	3.44	-0.640596
252	0.0256227	3.601	-0.640596
253	0.0243416	3.601	-0.640596
254	0.0231245	3.892	-0.640596
255	0.0219683	3.203	-0.640596
256	0.0208699	3.203	-0.640596
257	0.0198264	3.368	-0.640596
258	0.018835	3.368	-0.640596
259	0.0178933	3.368	-0.640596
260	0.0169986	3.13	-0.640596
261	0.0161487	3.13	-0.640596
262	0.0153413	3.13	-0.640596
263	0.0145742	3.13	-0.640596
264	0.0138455	3.13	-0.640596
265	0.0131532	3.13	-0.640596
266	0.0124956	3.13	-0.640596
267	0.0118708	3.62	-0.640596
268	0.0112772	3.511	-0.640596
269	0.0107134	3.511	-0.640596
270	0.0101777	3.683	-0.640596

Xmin = 3.683 Fmin = -0.640596

Выводы

В результате проделанной работы я исследовал метод имитации отжига. Убедился 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 fu(int x)
{
    double f = cos(x) * tanh(x);
    return f;
}

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

void print(const int i, const double t,
    const double co, const double f) {
    std::cout << "| " << std::setw(4) << i
        << "| " << std::setw(10) << t
        << "| " << std::setw(10) << co
        << "| " << std::setw(12) << f << "|" << std::endl;
}


int main()
{
    std::cout << "Ecstremum f(x) " << std::endl;
    std::cout  <<  "| " << std::setw(4) << "N"
        << "| " << std::setw(10) << "T"
        << "| " << std::setw(10) << "x"
        << "| " << std::setw(12) << "f(x)" << "|" << std::endl
        << std::string(50, '-') << std::endl;
    double tmin = 0.01;
    double tmax = 10000.0, xsch = 0.0, r = 0.0, a = 7, b = 11;
    int N = 1;
    double xn = 0, x = (double)(rand() % 2501) / 1000 + 1.5;
    r = fu(x);
    while (tmax > tmin)
    {
        xsch = (double)(rand() % 2501) / 1000 + 1.5;
        double d = fu(xsch) - fu(x);
        if (d <= 0)
        {
            xn = xsch;
            x = xsch;
            r = fu(xn);
        }
        else
        {
            double ver = (double)(a + rand() * 1. / RAND_MAX * (b - a));
            double P = exp(-d / tmax);
```

```
            if (ver < P)
            {
                xn = xsch;
                x = xsch;
                r = fu(xn);
            }
        }

        print(N, tmax, xn, fu(xn));
        tmax = tmax * 0.95;
        N++;

    }
    std::cout << std::endl << "Xmin = " << xn << "   Fmin = " << r <<
std::endl;



    std::cout << "\nEcstremum f(x)*sin(5x) " << std::endl;
    std::cout << "| " << std::setw(4) << "N"
        << "| " << std::setw(10) << "T"
        << "| " << std::setw(10) << "x"
        << "| " << std::setw(12) << "f(x)" << "|" << std::endl
        << std::string(50, '-') << std::endl;

    tmin = 0.01; tmax = 10000.0; xsch = 0.0; r = 0.0; a = 7; b = 11; N =
1; xn = 0;
    x = (double)(rand() % 2501) / 1000 + 1.5;
    r = fumin(x);
    while (tmax > tmin)
    {
        xsch = (double)(rand() % 2501) / 1000 + 1.5;
        double d = fumin(xsch) - fumin(x);
        if (d <= 0)
        {
            xn = xsch;
            x = xsch;
            r = fumin(xn);
        }
        else
        {
            double ver = (double)(a + rand() * 1. / RAND_MAX * (b - a));
            double P = exp(-d / tmax);
            if (ver < P)
            {
                xn = xsch;
                x = xsch;
                r = fumin(xn);
            }
        }

        print(N, tmax, xn, fumin(xn));
        tmax = tmax * 0.95;
        N++;
```

```
    }
    std::cout << std::endl << "Xmin = " << xn << "   Fmin = " << r <<
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.