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 1

Выполнил: Александроов А.А.,
студент группы ИУ8-31
Проверила: Коннова Н.С., доцент
каф. ИУ8

г. Москва, 2020

Цель работы

Изучение алгоритма имитации отжига экстремума на примере унимодальной и мультимодальной функций одного переменного.

Условие задачи

Постановка задачи

1. На интервале [-5, 2] задана унимодальная функция одного переменного f(x) = -0.5x*cos(0.5x)-0.5. Используя метод имитации отжига осуществить поиск минимума f (x).

2. При аналогичных исходных условиях осуществить поиск минимума f (x), модулированной сигналом sin5x , т.е. мультимодальной функции (-0.5x*cos(0.5x)-0.5) * sin(5x).

Графики заданных функций

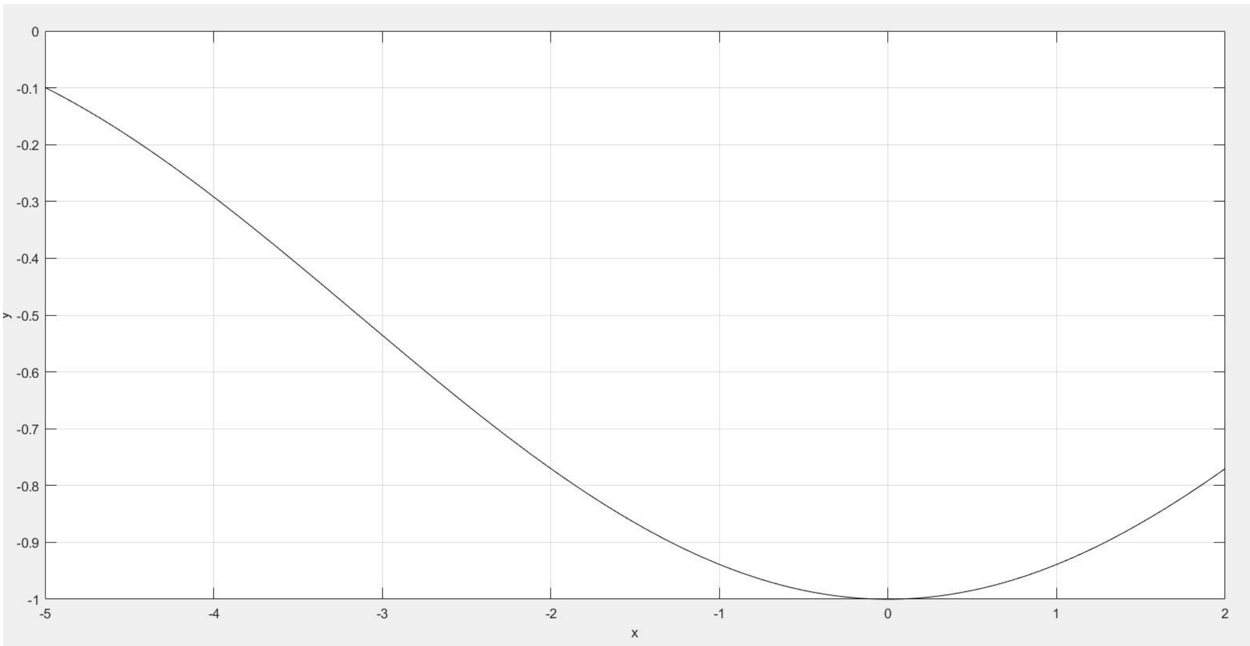

Рисунок 1 - График функции $f(x) = -0.5 * cos(0.5x) - 0.5$ на [-5;2]

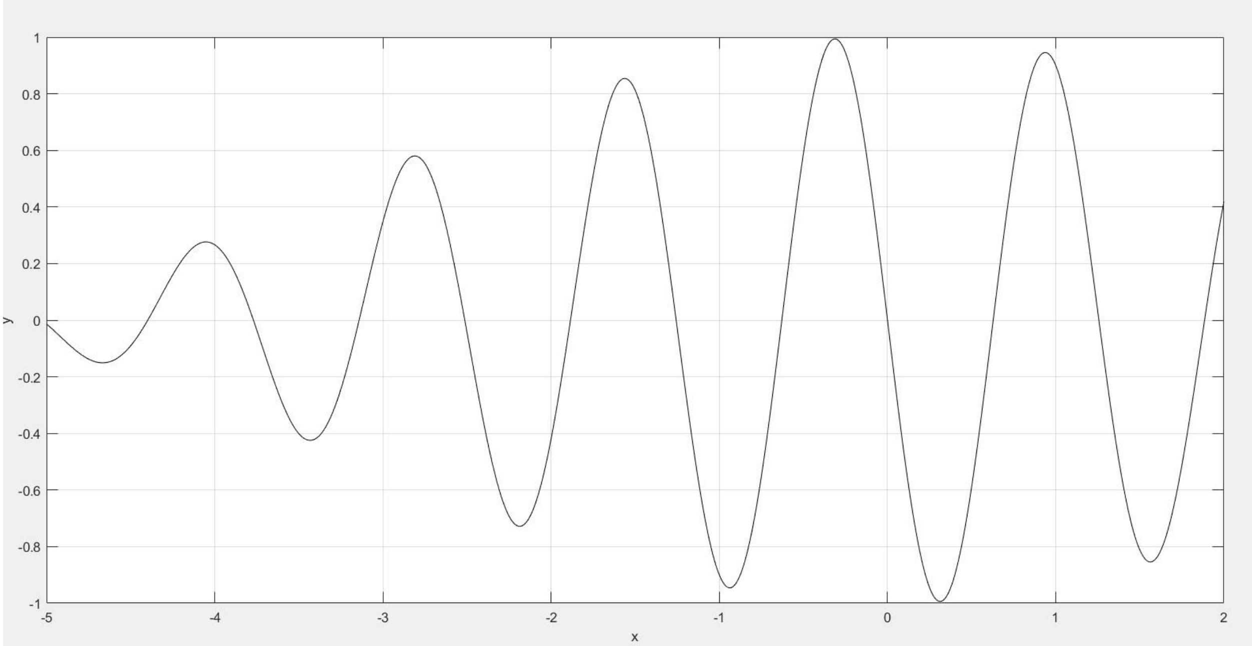

Рисунок 2 - График функции $f(x) = (-0.5 * \cos(0.5x) - 0.5) * \sin(5x)$ на [-5;2]

Поиск минимума унимодальной функции

Unomodal function

N	T	x	f(x)
1	10000	-2.4563	-0.667991
2	10000	1.25458	-0.904811
3	9500	-0.288279	-0.994815
4	9025	-0.245257	-0.996245
5	8573.75	-1.74437	-0.821578
6	8145.06	1.4165	-0.879751
7	7737.81	-4.33051	-0.21997
8	7350.92	-4.80754	-0.130033
9	6983.37	-4.27747	-0.231052
10	6634.2	-3.97954	-0.29659
11	6302.49	-3.67718	-0.367698

12	5987.37	-4.2148	-0.244389
13	5688	1.9249	-0.785755
14	5403.6	-1.7191	-0.826389
15	5133.42	0.0909591	-0.999483
16	4876.75	-4.98309	-0.101973
17	4632.91	1.12723	-0.922665
18	4401.27	-0.240709	-0.996383
19	4181.2	-0.863596	-0.954107
20	3972.14	-2.20105	-0.726563
21	3773.54	-0.428928	-0.988545
22	3584.86	-4.26494	-0.233698
23	3405.62	-2.61529	-0.630063
24	3235.34	-1.72124	-0.825984
25	3073.57	-4.42757	-0.200204
26	2919.89	-3.96649	-0.299572
27	2773.9	-1.17194	-0.916588
28	2635.2	-0.340591	-0.992767
29	2503.44	-4.55	-0.176287
30	2378.27	-2.14839	-0.738221
31	2259.36	-4.35753	-0.214398
32	2146.39	-1.42586	-0.878224
33	2039.07	-3.33707	-0.451209
34	1937.11	-0.640663	-0.974566
35	1840.26	-2.19357	-0.72823

36	1748.25	-4.35976	-0.213942
37	1660.83	-4.83474	-0.125493
38	1577.79	-4.97538	-0.103142
39	1498.9	0.0714764	-0.999681
40	1423.96	-0.215172	-0.997109
41	1352.76	-2.38895	-0.68375
42	1285.12	-3.54144	-0.400702
43	1220.87	-0.259389	-0.995801
44	1159.82	-0.262678	-0.995694
45	1101.83	-0.162957	-0.998341
46	1046.74	0.770828	-0.963321
47	994.403	-2.50355	-0.656819
48	944.682	0.826932	-0.957867
49	897.448	0.348065	-0.992447
50	852.576	1.8633	-0.798256
51	809.947	-3.07779	-0.515947
52	769.45	-0.0489918	-0.99985
53	730.977	1.28154	-0.900817
54	694.428	0.0993582	-0.999383
55	659.707	1.92178	-0.786395
56	626.722	0.264618	-0.99563
57	595.386	-4.14315	-0.259946
58	565.616	-1.81083	-0.80868
59	537.335	0.92833	-0.947098

60	510.469	-1.96049	-0.778409
61	484.945	-3.46062	-0.42058
62	460.698	0.371226	-0.991412
63	437.663	0.46827	-0.986358
64	415.78	-1.89145	-0.792577
65	394.991	-4.02694	-0.285822
66	375.241	-0.701031	-0.969598
67	356.479	-1.53049	-0.860607
68	338.655	-1.8741	-0.796084
69	321.723	-4.03649	-0.283666
70	305.636	-1.26022	-0.903982
71	290.355	-2.51027	-0.655224
72	275.837	-1.97867	-0.774622
73	262.045	1.8224	-0.8064
74	248.943	-1.09617	-0.926761
75	236.496	-2.05689	-0.758075
76	224.671	-2.83156	-0.577199
77	213.437	-1.69287	-0.831329
78	202.765	0.357614	-0.992028
79	192.627	-2.27171	-0.710679
80	182.996	-2.72972	-0.602242
81	173.846	-0.257956	-0.995847
82	165.154	-4.35849	-0.214202
83	156.896	-4.12194	-0.264611

84	149.051	-1.24994	-0.90549
85	141.599	0.705527	-0.969211
86	134.519	-3.14888	-0.498179
87	127.793	-1.9955	-0.771096
88	121.403	0.320638	-0.993588
89	115.333	1.72964	-0.824389
90	109.566	-4.86178	-0.121048
91	104.088	1.98386	-0.773539
92	98.8836	-1.122	-0.923362
93	93.9395	-1.81712	-0.807441
94	89.2425	1.89933	-0.790979
95	84.7804	-2.30785	-0.702451
96	80.5413	-3.94227	-0.305135
97	76.5143	1.64239	-0.840673
98	72.6886	-0.635139	-0.974999
99	69.0541	-1.78179	-0.814358
100	65.6014	-1.2885	-0.899775
101	62.3214	0.488489	-0.98516
102	59.2053	1.63835	-0.841412
103	56.245	1.13	-0.922295
104	53.4328	-2.62571	-0.627544
105	50.7611	-2.11995	-0.744448
106	48.2231	-3.92019	-0.310231
107	45.8119	-3.49052	-0.413211

108	43.5213	-0.915692	-0.948503
109	41.3453	-2.49663	-0.65846
110	39.278	-1.7853	-0.813676
111	37.3141	-2.35842	-0.690828
112	35.4484	-1.80144	-0.810523
113	33.676	1.51958	-0.86249
114	31.9922	0.115149	-0.999172
115	30.3926	-3.01312	-0.532097
116	28.8729	-3.87957	-0.319663
117	27.4293	-2.73866	-0.600053
118	26.0578	-2.82184	-0.579597
119	24.7549	-3.09627	-0.511331
120	23.5172	0.757835	-0.964533
121	22.3413	-0.725584	-0.967455
122	21.2243	-1.74549	-0.821364
123	20.1631	-1.08724	-0.927922
124	19.1549	-0.291943	-0.994683
125	18.1972	-1.62368	-0.844083
126	17.2873	-4.96034	-0.105441
127	16.4229	-4.63314	-0.160731
128	15.6018	-0.777609	-0.962681
129	14.8217	-4.23414	-0.240246
130	14.0806	1.72576	-0.825127
131	13.3766	-2.99502	-0.536611

132	12.7078	-2.63266	-0.625865
133	12.0724	-1.06275	-0.931056
134	11.4687	-1.13556	-0.921548
135	10.8953	-2.9476	-0.548423
136	10.3505	-3.69417	-0.363607
137	9.83302	-1.46965	-0.870975
138	9.34136	-1.43283	-0.877083
139	8.8743	0.350985	-0.99232
140	8.43058	1.06998	-0.930137
141	8.00905	-0.105999	-0.999298
142	7.6086	-0.172185	-0.998148
143	7.22817	-3.85148	-0.32623
144	6.86676	1.65809	-0.837791
145	6.52342	0.856572	-0.954839
146	6.19725	0.450461	-0.987371
147	5.88739	0.945138	-0.945201
148	5.59302	-0.293165	-0.994638
149	5.31337	1.312	-0.896219
150	5.0477	0.472695	-0.9861
151	4.79532	-1.23683	-0.9074
152	4.55555	-4.52232	-0.181591
153	4.32777	0.211724	-0.997201
154	4.11138	-0.585067	-0.978758
155	3.90581	-0.585067	-0.978758

156	3.71052	-4.85519	-0.122125
157	3.525	-2.35984	-0.6905
158	3.34875	-1.32484	-0.894253
159	3.18131	-0.0283477	-0.99995
160	3.02224	-4.00885	-0.289917
161	2.87113	-1.55744	-0.855906
162	2.72758	0.88515	-0.951826
163	2.5912	0.88515	-0.951826
164	2.46164	-4.96633	-0.104522
165	2.33856	0.845969	-0.955934
166	2.22163	0.890237	-0.95128
167	2.11055	-0.208892	-0.997275
168	2.00502	-0.213528	-0.997153
169	1.90477	-0.896891	-0.950561
170	1.80953	-0.722911	-0.967692
171	1.71905	-4.13372	-0.262017
172	1.6331	-0.245232	-0.996246
173	1.55145	0.295334	-0.994559
174	1.47387	0.295334	-0.994559
175	1.40018	-2.94015	-0.550277
176	1.33017	1.73363	-0.823629
177	1.26366	-1.53567	-0.859709
178	1.20048	-0.722799	-0.967701
179	1.14045	-1.59121	-0.849927

180	1.08343	0.26835	-0.995506
181	1.02926	1.15394	-0.91906
182	0.977798	-0.8465	-0.955879
183	0.928908	1.7269	-0.82491
184	0.882462	-0.812835	-0.959272
185	0.838339	-2.62601	-0.627474
186	0.796422	-1.9226	-0.786228
187	0.756601	-1.52845	-0.86096
188	0.718771	-4.2965	-0.227055
189	0.682833	-2.72405	-0.603628
190	0.648691	-0.193387	-0.997664
191	0.616256	-0.193387	-0.997664
192	0.585444	0.672896	-0.971967
193	0.556171	-0.692195	-0.970352
194	0.528363	-0.692195	-0.970352
195	0.501945	-1.44295	-0.875417
196	0.476847	-1.44295	-0.875417
197	0.453005	1.00385	-0.938329
198	0.430355	0.391743	-0.990439
199	0.408837	1.69948	-0.83009
200	0.388395	-1.33991	-0.891925
201	0.368975	0.853412	-0.955167
202	0.350527	0.31838	-0.993678
203	0.333	0.31838	-0.993678

204	0.31635	0.31838	-0.993678
205	0.300533	0.31838	-0.993678
206	0.285506	0.31838	-0.993678
207	0.271231	1.00071	-0.938707
208	0.257669	1.7018	-0.829653
209	0.244786	-1.28899	-0.899702
210	0.232547	1.25484	-0.904773
211	0.220919	-2.51979	-0.652959
212	0.209873	-1.43736	-0.876338
213	0.19938	1.44844	-0.874507
214	0.189411	-1.95985	-0.778542
215	0.17994	0.248645	-0.996141
216	0.170943	0.248645	-0.996141
217	0.162396	-0.747505	-0.965482
218	0.154276	0.995655	-0.939311
219	0.146562	1.20653	-0.911744
220	0.139234	-1.98929	-0.7724
221	0.132272	-1.57062	-0.853584
222	0.125659	0.395517	-0.990255
223	0.119376	0.0208824	-0.999973
224	0.113407	-0.616323	-0.976446
225	0.107737	1.09904	-0.926387
226	0.10235	1.09904	-0.926387

RESULT: Xmin = 1.09904 Fmin = -0.926387

Поиск минимума для мультимодальной функции

Multimodal function

N	T	x	f(x)
1	10000	-3.53002	-0.376004
2	10000	1.45932	-0.740609
3	9500	-2.72207	0.522207
4	9025	-4.26725	0.142041
5	8573.75	-2.02464	-0.49188
6	8145.06	-1.3769	0.501302
7	7737.81	-4.64832	-0.149916
8	7350.92	-1.04765	-0.806895
9	6983.37	0.128267	-0.59765
10	6634.2	0.0983495	-0.471881
11	6302.49	-1.01297	-0.879605
12	5987.37	-3.50759	-0.395352
13	5688	-3.41638	-0.423185
14	5403.6	-1.47508	0.772302
15	5133.42	-2.36688	-0.46039
16	4876.75	-3.36538	-0.39959
17	4632.91	-2.14551	-0.71247
18	4401.27	0.226286	-0.902128
19	4181.2	-0.601447	0.130946
20	3972.14	0.577928	-0.244129
21	3773.54	1.17682	0.355886

22	3584.86	1.90184	0.0666676
23	3405.62	-4.35319	0.0480716
24	3235.34	-1.77241	0.435418
25	3073.57	-3.71981	-0.088615
26	2919.89	-1.03966	-0.825836
27	2773.9	0.155126	-0.699114
28	2635.2	-2.12059	-0.68767
29	2503.44	-3.14436	-0.00691496
30	2378.27	-1.69414	0.677982
31	2259.36	-2.23976	-0.703119
32	2146.39	-4.70712	-0.147329
33	2039.07	-0.393629	0.91319
34	1937.11	-0.0286739	0.142872
35	1840.26	0.289171	-0.987028
36	1748.25	-0.49038	0.626786
37	1660.83	-3.74145	-0.0499692
38	1577.79	-4.63589	-0.148646
39	1498.9	-0.167766	0.742553
40	1423.96	-3.77886	0.0153667
41	1352.76	-1.58984	0.846321
42	1285.12	-0.0105964	0.0529571
43	1220.87	-1.68799	0.693393
44	1159.82	-3.78301	0.0224065
45	1101.83	0.491776	-0.621413

46	1046.74	1.76093	-0.4756
47	994.403	-1.3531	0.412742
48	944.682	-3.94326	0.23241
49	897.448	-2.84629	0.57101
50	852.576	-2.93558	0.472736
51	809.947	0.459567	-0.737326
52	769.45	1.47385	-0.770009
53	730.977	-0.175347	0.767178
54	694.428	-4.07975	0.273904
55	659.707	-2.56898	0.176281
56	626.722	-4.37105	0.028676
57	595.386	0.891575	0.920496
58	565.616	0.138574	-0.637982
59	537.335	0.497523	-0.598984
60	510.469	0.502688	-0.578419
61	484.945	-4.36414	0.0361333
62	460.698	1.73992	-0.545493
63	437.663	0.469787	-0.702444
64	415.78	-0.286773	0.985557
65	394.991	-1.04948	-0.802402
66	375.241	0.930218	0.945108
67	356.479	-1.28854	0.14294
68	338.655	-0.892457	-0.92145
69	321.723	0.190602	-0.813313

70	305.636	0.164281	-0.730869
71	290.355	-4.10446	0.26708
72	275.837	0.803879	0.738679
73	262.045	0.0533495	-0.263549
74	248.943	-0.443671	0.787788
75	236.496	-4.51563	-0.10129
76	224.671	0.185283	-0.797757
77	213.437	-4.81879	-0.110435
78	202.765	-1.67614	0.721342
79	192.627	-1.81668	0.270341
80	182.996	-3.17346	-0.0780559
81	173.846	-2.99151	0.366534
82	165.154	-3.44963	-0.423096
83	156.896	-3.25159	-0.246972
84	149.051	1.26359	-0.0314247
85	141.599	1.23949	0.0776898
86	134.519	1.34403	-0.377203
87	127.793	-2.04906	-0.555758
88	121.403	-2.69864	0.487704
89	115.333	-0.673382	-0.217142
90	109.566	-4.53436	-0.11282
91	104.088	0.222157	-0.893286
92	98.8836	-4.30567	0.100516
93	93.9395	0.591542	-0.178878

94	89.2425	0.747683	0.542607
95	84.7804	1.88118	-0.0150139
96	80.5413	1.23206	0.111326
97	76.5143	-4.93761	-0.0468686
98	72.6886	-4.56337	-0.12771
99	69.0541	-2.2517	-0.690621
100	65.6014	0.741119	0.516432
101	62.3214	-0.425159	0.840335
102	59.2053	-4.56088	-0.126575
103	56.245	1.12521	0.56378
104	53.4328	-2.27192	-0.664061
105	50.7611	0.388804	-0.922387
106	48.2231	-4.70165	-0.148137
107	45.8119	-1.41738	0.633276
108	43.5213	-0.0814963	0.396134
109	41.3453	0.883972	0.911511
110	39.278	-2.75488	0.557282
111	37.3141	1.34961	-0.399168
112	35.4484	-2.94403	0.458583
113	33.676	-4.95606	-0.036625
114	31.9922	0.7388	0.50704
115	30.3926	0.7388	0.50704
116	28.8729	0.939997	0.945712
117	27.4293	1.40239	-0.587406

118	26.0578	1.00238	0.896719
119	24.7549	-2.84931	0.569388
120	23.5172	-0.256497	0.954789
121	22.3413	1.72553	-0.590282
122	21.2243	0.493113	-0.61624
123	20.1631	-2.48063	-0.107602
124	19.1549	-4.00887	0.269658
125	18.1972	-0.760456	-0.591745
126	17.2873	-3.71051	-0.105256
127	16.4229	-4.17784	0.225141
128	15.6018	0.133512	-0.618382
129	14.8217	1.70613	-0.64621
130	14.0806	0.1089	-0.517605
131	13.3766	1.05574	0.786451
132	12.7078	-4.15409	0.241929
133	12.0724	1.09968	0.654588
134	11.4687	1.90473	0.077983
135	10.8953	-0.108694	0.516728
136	10.3505	0.844731	0.844133
137	9.83302	0.284024	-0.983693
138	9.34136	-0.152885	0.691113
139	8.8743	-4.56529	-0.128561
140	8.43058	-2.40938	-0.337058
141	8.00905	-1.54446	0.850748

142	7.6086	-3.91627	0.207904
143	7.22817	0.974908	0.92941
144	6.86676	-2.92514	0.489204
145	6.52342	-0.777647	-0.65383
146	6.19725	-1.74947	0.514348
147	5.88739	-0.562639	0.316188
148	5.59302	-0.927309	-0.944489
149	5.31337	-0.703778	-0.357121
150	5.0477	-1.22118	-0.160439
151	4.79532	-2.05049	-0.559231
152	4.55555	-2.16544	-0.724076
153	4.32777	-2.43001	-0.272647
154	4.11138	-3.06575	0.1921
155	3.90581	-1.29985	0.192521
156	3.71052	-4.6447	-0.149615
157	3.525	-2.45512	-0.191598
158	3.34875	-2.25946	-0.681219
159	3.18131	0.639321	0.0535931
160	3.02224	-0.88417	-0.911762
161	2.87113	1.07404	0.735649
162	2.72758	-4.42863	-0.030281
163	2.5912	1.23428	0.101253
164	2.46164	1.54628	-0.851423
165	2.33856	0.245269	-0.937727

166	2.22163	0.245269	-0.937727
167	2.11055	-3.5042	-0.397878
168	2.00502	-3.79401	0.0408355
169	1.90477	1.80178	-0.327426
170	1.80953	-1.22313	-0.151626
171	1.71905	-1.22313	-0.151626
172	1.6331	0.181878	-0.787498
173	1.55145	0.181878	-0.787498
174	1.47387	0.181878	-0.787498
175	1.40018	0.181878	-0.787498
176	1.33017	1.60586	-0.834312
177	1.26366	-2.23317	-0.708973
178	1.20048	0.245182	-0.937582
179	1.14045	0.245182	-0.937582
180	1.08343	0.245182	-0.937582
181	1.02926	0.245182	-0.937582
182	0.977798	0.245182	-0.937582
183	0.928908	-3.20123	-0.142516
184	0.882462	-1.31568	0.260568
185	0.838339	-1.31568	0.260568
186	0.796422	-3.13694	0.0116597
187	0.756601	-0.343668	0.981851
188	0.718771	-0.789625	-0.694123
189	0.682833	1.86323	-0.0865372

190	0.648691	1.86323	-0.0865372
191	0.616256	-0.0846481	0.410533
192	0.585444	-1.10931	-0.621442
193	0.556171	0.137298	-0.633079
194	0.528363	0.137298	-0.633079
195	0.501945	0.137298	-0.633079
196	0.476847	0.137298	-0.633079
197	0.453005	0.137298	-0.633079
198	0.430355	0.137298	-0.633079
199	0.408837	0.137298	-0.633079
200	0.388395	0.215407	-0.878004
201	0.368975	0.215407	-0.878004
202	0.350527	0.215407	-0.878004
203	0.333	0.215407	-0.878004
204	0.31635	0.215407	-0.878004
205	0.300533	0.215407	-0.878004
206	0.285506	-0.864696	-0.882751
207	0.271231	-2.14921	-0.715181
208	0.257669	-3.22457	-0.193182
209	0.244786	-3.22457	-0.193182
210	0.232547	-3.22457	-0.193182
211	0.220919	-3.22457	-0.193182
212	0.209873	0.264164	-0.964698
213	0.19938	0.264164	-0.964698

214	0.189411	0.264164	-0.964698
215	0.17994	0.264164	-0.964698
216	0.170943	0.264164	-0.964698
217	0.162396	0.264164	-0.964698
218	0.154276	0.264164	-0.964698
219	0.146562	0.264164	-0.964698
220	0.139234	0.264164	-0.964698
221	0.132272	0.264164	-0.964698
222	0.125659	0.264164	-0.964698
223	0.119376	0.264164	-0.964698
224	0.113407	0.264164	-0.964698
225	0.107737	0.264164	-0.964698
226	0.10235	0.264164	-0.964698

```
RESULT: Xmin = 0.264164         Fmin = -0.964698
```

Ссылка на репозиторий с программой – https://github.com/Vumba798/tsisa_lab03

Исходный код программы предоставлен в приложениях 1-4.

Выводы

В результате проделанной работы можно сказать, что метод имитации отжига может использоваться для унимодальных и мультимодальных функций одинаково эффективно. При этом, эффективность этого метода гораздо выше, чем у метода случайного поиска.

Приложение 1. Исходный код файла demo.cpp

```
#include <annealing.hpp>
#include <iostream>

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

int main(){
    double tMin = 0;
    double tMax = 0;
    while(true) {
        cout << "Please, input minimal temperature: ";
        cin >> tMin;
        cout << "Plesae, input maximal temperature: ";
        cin >> tMax;
        if (tMax > tMin) {
            break;
        }
        cout << "tMin must be less than tMax!" << endl;
    }

    cout << "\n\n\t\tUnomodal function" << endl;
    print_measures(annealing(tMin, tMax, unomodal_function));
    cout << "\n\n\n\t\tMultimodal function" << endl;
    print_measures(annealing(tMin, tMax, multimodal_function));


    return 0;
}
```

Приложение 2. Исходный код файла anneanling.hpp

```
#ifndef ANNEALING
#define ANNEALING
#include <vector>

struct Measure {
  size_t number = 0;
  double temperature = 0;
  double x = 0;
  double value = 0;

  inline Measure(const size_t& n,
    const double& temp,
    const double& point,
    const double& val) :
    number(n),
```

```
        temperature(temp),
        x(point),
        value(val) {}
};

const double unomodal_function(const double& x) noexcept;
const double multimodal_function(const double& x) noexcept;
const std::vector<Measure> annealing(const double& minTemp,
        const double& maxTemp, const double (*func)(const double& x));
void print_measures(const std::vector<Measure> &measures);

#endif //ANNEALING
```

Приложение 3. Исходный код файла annealing.cpp

```
#include <annealing.hpp>
#include <cmath>
#include <iostream>
#include <iomanip>
#include <random>
#include <time.h>

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

const double unomodal_function(const double& x) noexcept {
   return -0.5*cos(0.5*x) - 0.5;


}
const double multimodal_function(const double& x) noexcept {
   return std::sin(5*x) * (-0.5 * std::cos(0.5*x) - 0.5);
}
```

```
const std::vector<Measure> annealing(const double& minTemp,
        const double& maxTemp, const double(*func)(const double& x)) {
    std::srand(std::time(NULL));
    double temp = maxTemp;
    double xPrev = (static_cast<double>(std::rand()) /
        static_cast<double>(RAND_MAX/7) - 5); // Previous point on the interval
    double valuePrev = func(xPrev);
    double x = 0;
    double value = 0;
    double delta = 0;
    size_t N = 1;
    std::vector<Measure> measures = {{1, temp, xPrev, valuePrev}};

    while (temp > minTemp) {
        x = static_cast<double>(std::rand()) / static_cast<double>(RAND_MAX/7) - 5;
        value = func(x);
        delta = value - valuePrev;
        if (delta <= 0) {
            xPrev = x;
            valuePrev = value;
        }else if (static_cast<double>(std::rand())/static_cast<double>(RAND_MAX)
                <= std::exp(-delta / temp)) {
            xPrev = x;
            valuePrev = value;
        }
        ++N;
        measures.emplace_back(N, temp, xPrev, valuePrev);
        temp *= 0.95;
    }
    return measures;
}
```

```
void print_measures(const std::vector<Measure> &measures) {
    cout << "|----------------------------------------------------------|\n";
    cout << "| " << std::setw(8) << "N";
    cout << " | " << std::setw(12) << "T";
    cout << " | " << std::setw(12) << "x";
    cout << " | " << std::setw(15) << "f(x)" << " |" << endl;;
    for (size_t i = 0; i < measures.size(); ++i) {
        cout << "|-" << std::setfill('-') << std::setw(8) << "-";
        cout << "-+-" << std::setw(12) << "-";
        cout << "-+-" << std::setw(12) << "-";
        cout << "-+-" << std::setw(15) << "-" << "-|" << endl;;

        cout << "| " << std::setfill(' ');
        cout << std::setw(8) << measures[i].number << " | ";
        cout << std::setw(12) << measures[i].temperature << " | ";
        cout << std::setw(12) << measures[i].x << " | ";
        cout << std::setw(15) << measures[i].value << " |" << endl;
    }
    cout << "|----------------------------------------------------------|\n";
    cout << "\tRESULT: Xmin = " << measures.back().x;
    cout << "\t\tFmin = " << measures.back().value << endl;
}
```

Приложение 4. Исходный код файла CMakeLists.txt

```
cmake_minimum_required(VERSION 3.4)
set(CMAKE_CXX_STANDARD 11)
set(CMAKE_CXX_STANDARD_REQUIRED ON)

project(lab03)

add_library(annealing STATIC
  ${CMAKE_CURRENT_SOURCE_DIR}/sources/annealing.cpp
)
add_executable(demo
  ${CMAKE_CURRENT_SOURCE_DIR}/sources/demo.cpp
)

target_include_directories(annealing
    PUBLIC ${CMAKE_CURRENT_SOURCE_DIR}/include
)

target_include_directories(demo
    PUBLIC ${CMAKE_CURRENT_SOURCE_DIR}/include
)

target_link_libraries(demo PUBLIC annealing).
```

Контрольные вопросы

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

Сущность метода имитации отжига заключается в том, что локальное или же «субоптимальное» решение, полученное во время поиска минимального решения, может рассматриваться в качестве дефектного. Происходит это в результате случайных флуктуаций, амплитуда которых уменьшается с ростом номера итерации.

Этот метод позволяет избежать попадания в локальные минимумы с относительно высокой эффективностью, а значит, найти истинное минимальное значение быстрыми темпами.

Изначально этот метод был использован в исправлении дефектов в кристаллической решётке металла, однако с середины 20-го века этот метод был использован для решения различных задач на оптимизацию, в частности он используется в финансовых, комбинаторных проблемах, может применятся для решения задач в области компьютерной графики и обучения нейронных сетей.