GODDARD
GRANT

IN-61-CR
253025
34P.

FINAL REPORT

TO

THE NATIONAL AERONAUTICS AND SPACE AGENCY

FOR

POWER SPECTRAL ESTIMATION ALGORTHMS

(CONTRACT No. NAG-5-499)

MANJIT S. BHATIA

PRINCIPAL INVESTIGATOR

BOWIE STATE UNIVERSITY
BOWIE, MARYLAND

DECEMBER, 1989

(NASA-CR-186189) POWER SPECTRAL ESTIMATION ALGORITHMS Final Report (Bowie State Coll.) 34 p CSCL 09B

N90-14812

Unclas
G3/61 0253025

ORIGINAL PAGE IS
OF POOR QUALITY

POWER SPECTRAL ESTIMATION ALGORITHMS

In this project, we have developed algorithms to estimate the power spectrum using Maximum Entropy Methods. These algorithms were coded in FORTRAN 77 and were implemented on the VAX 780.

We recall that the important considerations in this analysis are:

1) RESOLUTION: How close in frequency can two spectral components be spaced and still be identified?

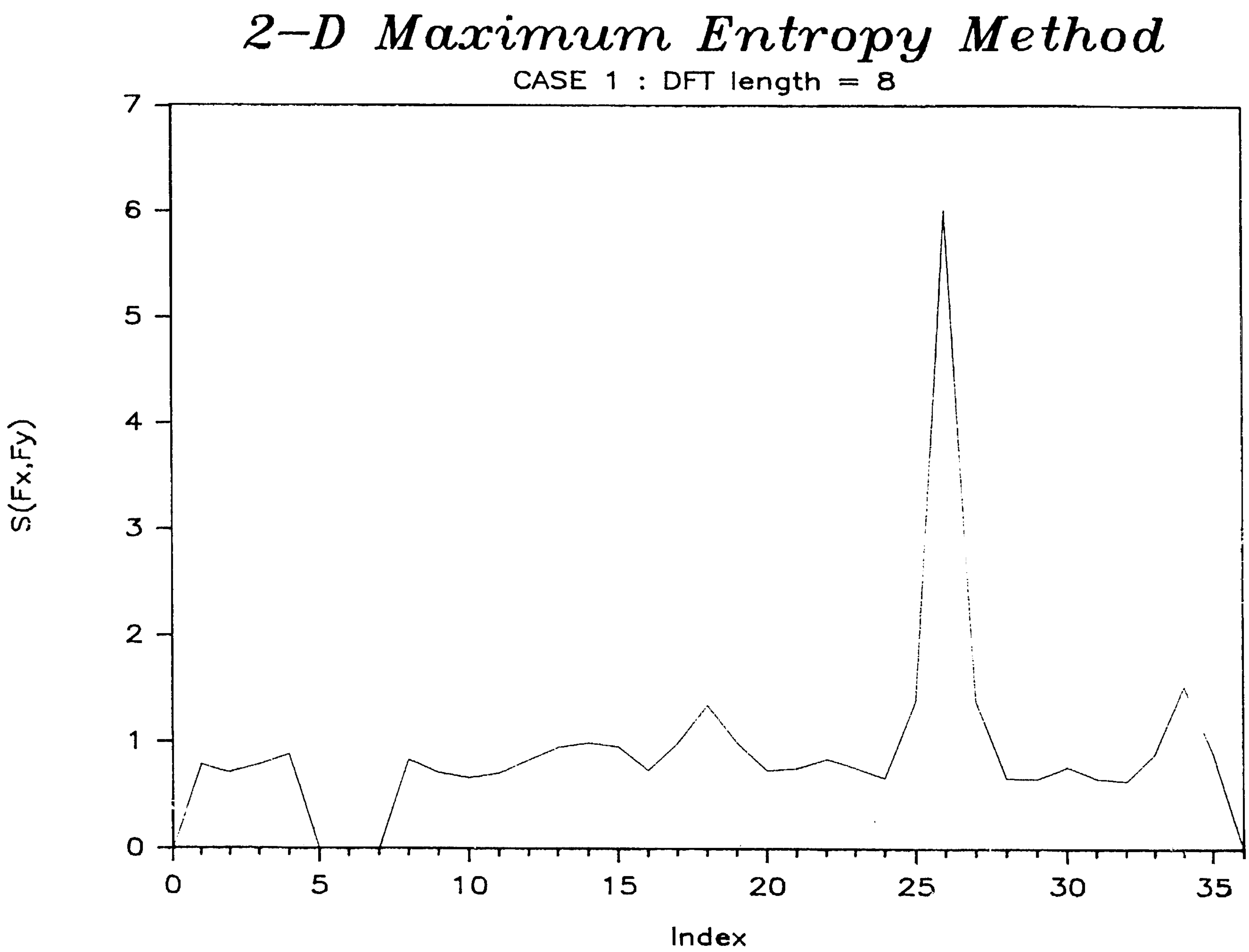
2) DYNAMIC RANGE: How small can a spectral peak be, relative to the largest, and still be observed in the spectra?

3) VARIANCE: How accurate is the estimate of the spectra to the actual spectra?

Our work with the application of the algorithms based on Maximum Entropy Methods to a variety of data shows that these criteria are met quite well. Additional work in this direction would help confirm our findings.

All of the software developed has been turned over to the technical monitor. A copy of a typical program is included in this report.

Some of the actual data and graphs based on this data are also included in this report.



2-D Maximum Entropy Method
CASE 1 : DFT length = 8
S(Fx,Fy)
0
1
2
3
4
5
6
7
0
5
10
15
20
25
30
35
Index

CASE 1:

NUMBER OF SINUSOIDS = 1
NOISE POWER = 5.0

SINUSOID = 1
POWER = 1.0
XFREQ = 0.375
YFREQ = 0.25

ORIGINAL PAGE IS
OF POOR QUALITY

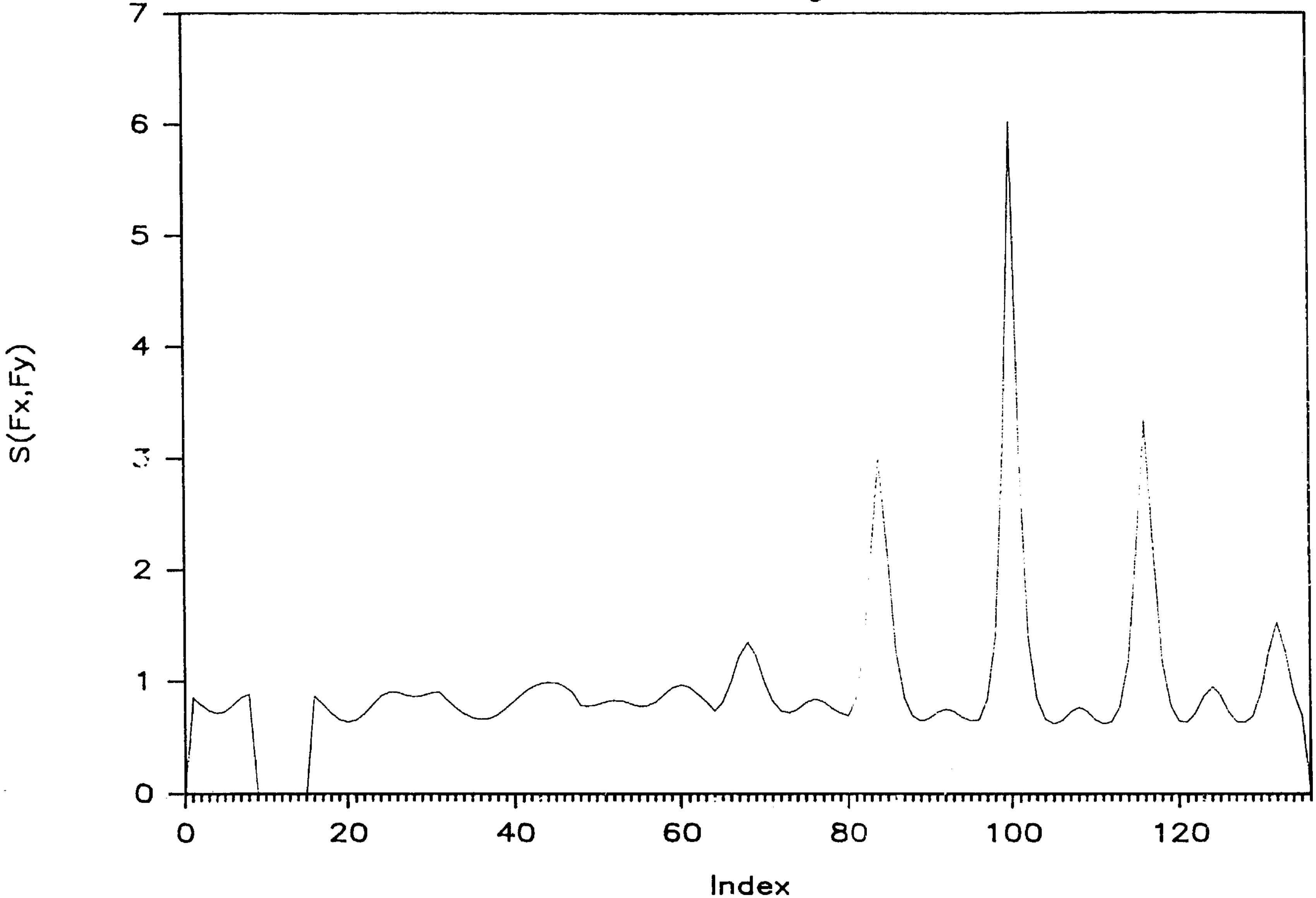
ACF MATRIX = 5 x 5

DFT LENGTH = 8

INDEX	Fx	Fy	S(Fx,Fy)
0	0.00000	0.00000	0.00000
1	0.00000	0.12500	0.79854
2	0.00000	0.25000	0.72018
3	0.00000	0.37500	0.79854
4	0.00000	0.50000	0.89603
5	0.00000	0.62500	0.00000
6	0.00000	0.75000	0.00000
7	0.00000	0.87500	0.00000
8	0.12500	0.00000	0.84068
9	0.12500	0.12500	0.717[illegible]
10	0.12500	0.25000	0.66839
11	0.12500	0.37500	0.71721
12	0.12500	0.50000	0.84068
13	0.12500	0.62500	0.95897
14	0.12500	0.75000	1.00100
15	0.12500	0.87500	0.95897
16	0.25000	0.00000	0.73856
17	0.25000	0.12500	1.00136
18	0.25000	0.25000	1.36219
19	0.25000	0.37500	1.00136
20	0.25000	0.50000	0.73856
21	0.25000	0.62500	0.76410
22	0.25000	0.75000	0.85278
23	0.25000	0.87500	0.76410
24	0.37500	0.00000	0.66658
25	0.37500	0.12500	1.39413
26	0.37500	0.25000	6.03194
27	0.37500	0.37500	1.39414
28	0.37500	0.50000	0.66658
29	0.37500	0.62500	0.66343
30	0.37500	0.75000	0.77938
31	0.37500	0.87500	0.66343
32	0.50000	0.00000	0.64292
33	0.50000	0.12500	0.90740
34	0.50000	0.25000	1.54154
35	0.50000	0.37500	0.90740
36	0.50000	0.50000	0.00000

2-D Maximum Entropy Method

CASE 2 : DFT length = 16



CASE 2:

NUMBER OF SINUSOIDS = 1
NOISE POWER = 5.0

SINUSOID = 1
POWER = 1.0
XFREQ = 0.375
YFREQ = 0.25

ORIGINAL PAGE IS
OF POOR QUALITY

ACF MATRIX = 5 x 5

DFT LENGTH = 16

INDEX	Fx	Fy	S(Fx,Fy)
0	0.00000	0.00000	0.00000
1	0.00000	0.06250	0.86509
2	0.00000	0.12500	0.79854
3	0.00000	0.18750	0.74149
4	0.00000	0.25000	0.72018
5	0.00000	0.31250	0.74149
6	0.00000	0.37500	0.79854
7	0.00000	0.43750	0.86510
8	0.00000	0.50000	0.89603
9	0.00000	0.56250	0.00000
10	0.00000	0.62500	0.00000
11	0.00000	0.68750	0.00000
12	0.00000	0.75000	0.00000
13	0.00000	0.81250	0.00000
14	0.00000	0.87500	0.00000
15	0.00000	0.93750	0.00000
16	0.06250	0.00000	0.88072
17	0.06250	0.06250	0.80381
18	0.06250	0.12500	0.72171
19	0.06250	0.18750	0.66426
20	0.06250	0.25000	0.64420
21	0.06250	0.31250	0.66427
22	0.06250	0.37500	0.72171
23	0.06250	0.43750	0.80381
24	0.06250	0.50000	0.88072
25	0.06250	0.56250	0.91754
26	0.06250	0.62500	0.90855
27	0.06250	0.68750	0.88251
28	0.06250	0.75000	0.87006
29	0.06250	0.81250	0.88251
30	0.06250	0.87500	0.90855
31	0.06250	0.93750	0.91754
32	0.12500	0.00000	0.84068
33	0.12500	0.06250	0.77331
34	0.12500	0.12500	0.71720
35	0.12500	0.18750	0.68087
36	0.12500	0.25000	0.66339
37	0.12500	0.31250	[illegible]

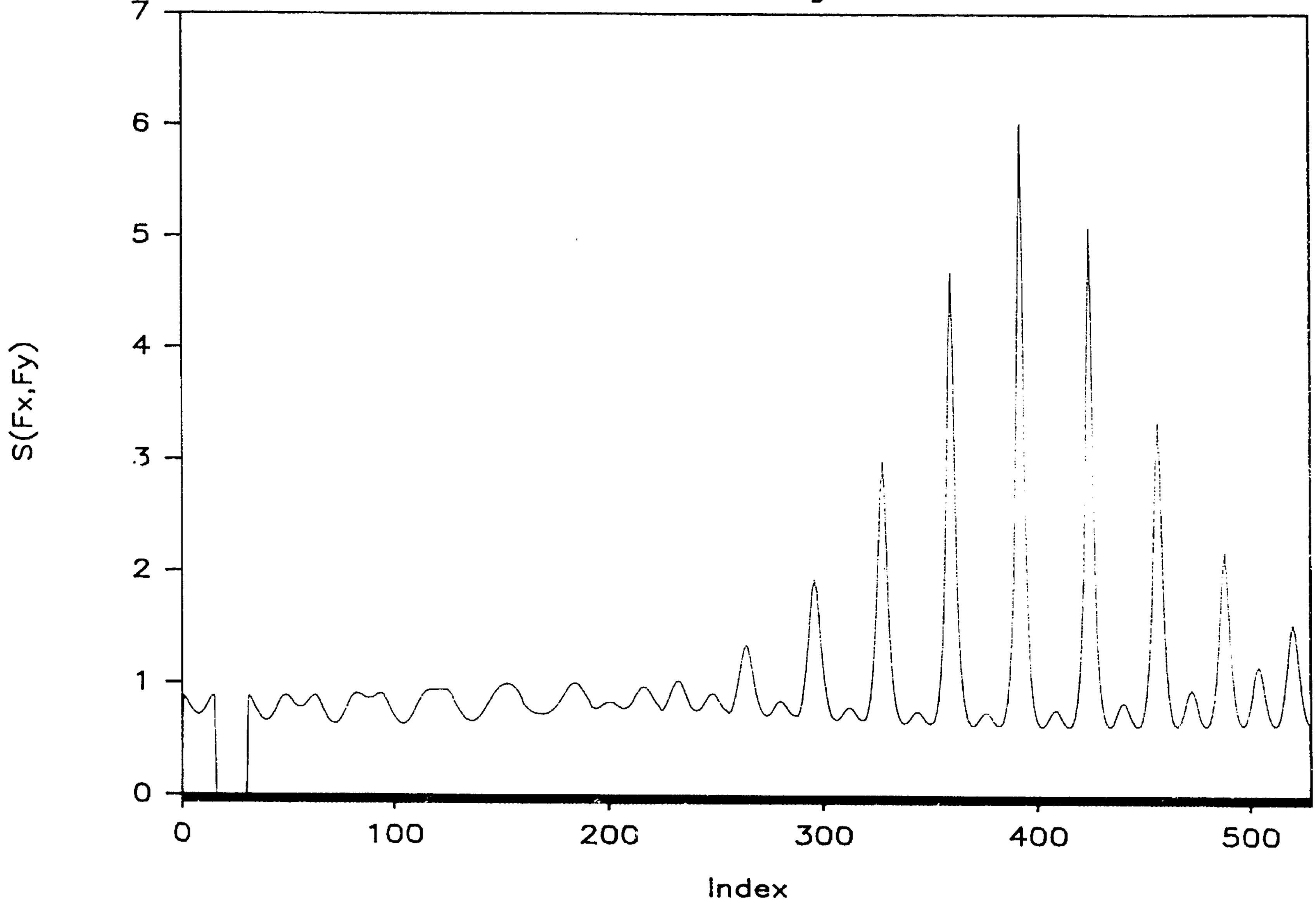
38	0.12500	0.37500	[illegible]
39	0.12500	0.43750	[illegible]
40	0.12500	0.50000	[illegible]
41	0.12500	0.56250	0.90570
42	0.12500	0.62500	[illegible]
43	0.12500	0.68750	[illegible]
44	0.12500	0.75000	[illegible]
45	0.12500	0.81250	0.95066
46	0.12500	0.87500	[illegible]
47	0.12500	0.93750	0.90570
48	0.18750	0.00000	0.78911
49	0.18750	0.06250	0.78425
50	0.18750	0.12500	0.80358
51	0.18750	0.18750	0.82999
52	0.18750	0.25000	0.84224
53	0.18750	0.31250	[illegible]
54	0.18750	0.37500	0.80358
55	0.18750	0.43750	[illegible]
56	0.18750	0.50000	0.78911
57	0.18750	0.56250	0.82527
58	0.18750	0.62500	0.88707
59	0.18750	0.68750	0.95074
60	0.18750	0.75000	0.97879
61	0.18750	0.81250	0.95074
62	0.18750	0.87500	0.88707
63	0.18750	0.93750	0.82527
64	0.25000	0.00000	0.73856
65	0.25000	0.06250	0.82548
66	0.25000	0.12500	1.00136
67	0.25000	0.18750	1.23537
68	0.25000	0.25000	1.36219
69	0.25000	0.31250	1.23537
70	0.25000	0.37500	1.00136
71	0.25000	0.43750	0.82548
72	0.25000	0.50000	0.73856
73	0.25000	0.56250	0.72504
74	0.25000	0.62500	0.76410
75	0.25000	0.68750	0.82331
76	0.25000	0.75000	[illegible]
77	0.25000	0.81250	0.82331
78	0.25000	0.87500	0.76410
79	0.25000	0.93750	0.72504
80	0.31250	0.00000	0.69668
81	0.31250	0.06250	0.86301
82	0.31250	0.12500	1.27659
83	0.31250	0.18750	2.17168
84	0.31250	0.25000	2.99733
85	0.31250	0.31250	2.17168
86	0.31250	0.37500	1.27659
87	0.31250	0.43750	0.86301
88	0.31250	0.50000	0.69669
89	0.31250	0.56250	[illegible]
90	0.31250	0.62500	0.67869
91	0.31250	0.68750	0.73416
92	0.31250	0.75000	[illegible]

ORIGINAL PAGE IS OF POOR QUALITY

93	0.31250	0.81[illegible]	[illegible]
94	0.31250	0.87500	0.6[illegible]
95	0.31250	0.93750	0.6[illegible]
96	0.37500	0.00000	0.66658
97	0.37500	0.06250	0.85266
98	0.37500	0.12500	1.39414
99	0.37500	0.18750	3.10639
100	0.37500	0.25000	6.03196
101	0.37500	0.31250	3.10640
102	0.37500	0.37500	1.39414
103	0.37500	0.43750	0.85266
104	0.37500	0.50000	0.66658
105	0.37500	0.56250	0.62486
106	0.37500	0.62500	0.66343
107	0.37500	0.68750	0.73847
108	0.37500	0.75000	0.77938
109	0.37500	0.81250	0.73847
110	0.37500	0.87500	0.66343
111	0.37500	0.93750	0.62486
112	0.43750	0.00000	0.64878
113	0.43750	0.06250	0.78615
114	0.43750	0.12500	1.18596
115	0.43750	0.18750	2.20205
116	0.43750	0.25000	3.35561
117	0.43750	0.31250	2.20206
118	0.43750	0.37500	1.18596
119	0.43750	0.43750	0.78615
120	0.43750	0.50000	0.64878
121	0.43750	0.56250	0.64322
122	0.43750	0.62500	0.73235
123	0.43750	0.68750	0.87990
124	0.43750	0.75000	0.96469
125	0.43750	0.81250	0.8[illegible]
126	0.43750	0.87500	0.73235
127	0.43750	0.93750	0.64322
128	0.50000	0.00000	0.64292
129	0.50000	0.06250	0.70293
130	0.50000	0.12500	0.95740
131	0.50000	0.18750	1.27962
132	0.50000	0.25000	1.54134
133	0.50000	0.31250	1.27962
134	0.50000	0.37500	0.95740
135	0.50000	0.43750	0.70293
136	0.50000	0.50000	0.00000

2–D Maximum Entropy Method

CASE 3 : DFT length = 32



CASE 3:

NUMBER OF SINUSOIDS = 1
NOISE POWER = 5.0

SINUSOID = 1
POWER = 1.0
XFREQ = 0.375
YFREQ = 0.25

ORIGINAL PAGE IS
OF POOR QUALITY

ACF MATRIX = 5 x 5

DFT LENGTH = 32

INDEX	Fx	Fy	S(Fx,Fy)
0	0.00000	0.00000	0.00000
1	0.00000	0.03130	0.88778
2	0.00000	0.06250	0.86509
3	0.00000	0.09380	0.83323
4	0.00000	0.12500	0.79854
5	0.00000	0.15630	0.76662
6	0.00000	0.18750	0.74149
7	0.00000	0.21880	0.72560
8	0.00000	0.25000	0.72018
9	0.00000	0.28130	0.72560
10	0.00000	0.31250	0.74149
11	0.00000	0.34380	0.76662
12	0.00000	0.37500	0.79854
13	0.00000	0.40630	0.83323
14	0.00000	0.43750	0.86510
15	0.00000	0.46880	0.88778
16	0.00000	0.50000	0.89603
17	0.00000	0.53130	0.00000
18	0.00000	0.56250	0.00000
19	0.00000	0.59380	0.00000
20	0.00000	0.62500	0.00000
21	0.00000	0.65630	0.00000
22	0.00000	0.68750	0.00000
23	0.00000	0.71880	0.00000
24	0.00000	0.75000	0.00000
25	0.00000	0.78130	0.00000
26	0.00000	0.81250	0.00000
27	0.00000	0.84380	0.00000
28	0.00000	0.87500	0.00000
29	0.00000	0.90630	0.00000
30	0.00000	0.93750	0.00000
31	0.00000	0.96880	0.00000
32	0.03130	0.00000	0.89210
33	0.03130	0.03130	0.86763
34	0.03130	0.06250	0.83231
35	0.03130	0.09380	0.79200
36	0.03130	0.12500	0.75249
37	0.03130	0.15630	0.71830

38	0.03130	0.18750	0.672[illegible]
39	0.03130	0.21880	0.67627
40	0.03130	0.25000	0.67085
41	0.03130	0.28130	0.67619
42	0.03130	0.31250	0.69236
43	0.03130	0.34380	0.71830
44	0.03130	0.37500	0.75247
45	0.03130	0.40630	0.79200
46	0.03130	0.43750	0.83231
47	0.03130	0.46880	0.86764
48	0.03130	0.50000	0.89210
49	0.03130	0.53130	0.90173
50	0.03130	0.56250	0.89606
51	0.03130	0.59380	0.87832
52	0.03130	0.62500	0.85599
53	0.03130	0.65630	0.82887
54	0.03130	0.68750	0.80776
55	0.03130	0.71880	0.79391
56	0.03130	0.75000	0.78910
57	0.03130	0.78130	0.79391
58	0.03130	0.81250	0.80776
59	0.03130	0.84380	0.82887
60	0.03130	0.87500	0.85598
61	0.03130	0.90630	0.87832
62	0.03130	0.93750	0.89606
63	0.03130	0.96880	0.90172
64	0.06250	0.00000	0.86072
65	0.06250	0.03130	0.84535
66	0.06250	0.06250	0.80381
67	0.06250	0.09380	0.76112
68	0.06250	0.12500	0.72171
69	0.06250	0.15630	0.68877
70	0.06250	0.18750	0.66426
71	0.06250	0.21880	0.64925
72	0.06250	0.25000	0.64420
73	0.06250	0.28130	0.64925
74	0.06250	0.31250	0.66427
75	0.06250	0.34380	0.68877
76	0.06250	0.37500	0.72171
77	0.06250	0.40630	0.76112
78	0.06250	0.43750	0.80381
79	0.06250	0.46880	0.84535
80	0.06250	0.50000	0.86072
81	0.06250	0.53130	0.90555
82	0.06250	0.56250	0.91734
83	0.06250	0.59380	0.91745
84	0.06250	0.62500	0.90855
85	0.06250	0.65630	0.89540
86	0.06250	0.68750	0.88251
87	0.06250	0.71880	0.87335
88	0.06250	0.75000	0.87006
89	0.06250	0.78130	0.87335
90	0.06250	0.81250	0.88251
91	0.06250	0.84380	0.89540
92	0.06250	0.87500	0.90855

ORIGINAL PAGE IS
OF POOR QUALITY

148	0.12500	0.62500	[illegible]
149	0.12500	[illegible]	[illegible]
150	0.12500	[illegible]	[illegible]
151	0.12500	0.71880	[illegible]
152	0.12500	0.75000	1.00[illegible]
153	0.12500	0.78130	0.99843
154	0.12500	0.81250	0.99066
155	0.12500	0.84380	0.97754
156	0.12500	0.87500	0.95897
157	0.12500	0.90630	0.93516
158	0.12500	0.93750	0.90670
159	0.12500	0.96880	0.87471
160	0.15630	0.00000	0.81546
161	0.15630	0.03130	0.79259
162	0.15630	0.06250	0.77381
163	0.15630	0.09380	0.75896
164	0.15630	0.12500	0.74771
165	0.15630	0.15630	0.73960
166	0.15630	0.18750	0.73421
167	0.15630	0.21880	0.73114
168	0.15630	0.25000	0.73014
169	0.15630	0.28130	0.73114
170	0.15630	0.31250	0.73421
171	0.15630	0.34380	0.73961
172	0.15630	0.37500	0.74771
173	0.15630	0.40630	0.75896
174	0.15630	0.43750	0.77381
175	0.15630	0.46880	0.79259
176	0.15630	0.50000	0.81546
177	0.15630	0.53130	0.84223
178	0.15630	0.56250	0.87226
179	0.15630	0.59380	0.90429
180	0.15630	0.62500	0.93641
181	0.15630	0.65630	0.96603
182	0.15630	0.68750	0.99021
183	0.15630	0.71880	1.00612
184	0.15630	0.75000	1.01168
185	0.15630	0.78130	1.00612
186	0.15630	0.81250	0.99021
187	0.15630	0.84380	0.96603
188	0.15630	0.87500	0.93641
189	0.15630	0.90630	0.90429
190	0.15630	0.93750	0.87226
191	0.15630	0.96880	0.84223
192	0.18750	0.00000	0.78911
193	0.18750	0.03130	0.78296
194	0.18750	0.06250	0.78425
195	0.18750	0.09380	0.79175
196	0.18750	0.12500	0.80353
197	0.18750	0.15630	0.81727
198	0.18750	0.18750	0.82999
199	0.18750	0.21880	0.83899
200	0.18750	0.25000	0.84224
201	0.18750	0.28130	0.83899
202	0.18750	0.31250	0.82999

ORIGINAL PAGE IS OF POOR QUALITY

203	0.18750	[illegible]	[illegible]
204	0.18750	[illegible]	[illegible]
205	0.18750	0.40630	[illegible]
206	0.18750	0.43750	[illegible]
207	0.18750	0.46880	[illegible]
208	0.18750	0.50000	[illegible]
209	0.18750	0.53130	0.80328
210	0.18750	0.56250	0.82527
211	0.18750	0.59380	0.85397
212	0.18750	0.62500	[illegible]
213	0.18750	0.65630	[illegible]
214	0.18750	0.68750	[illegible]
215	0.18750	0.71880	[illegible]
216	0.18750	0.75000	[illegible]
217	0.18750	0.78130	[illegible]
218	0.18750	0.81250	[illegible]
219	0.18750	0.84380	0.92093
220	0.18750	0.87500	0.88707
221	0.18750	0.90630	0.85397
222	0.18750	0.93750	0.82527
223	0.18750	0.96880	0.80328
224	0.21880	0.00000	0.76310
225	0.21880	0.03130	0.77648
226	0.21880	0.06250	0.80264
227	0.21880	0.09380	0.84061
228	0.21880	0.12500	0.88780
229	0.21880	0.15630	0.93913
230	0.21880	0.18750	0.98678
231	0.21880	0.21880	1.02109
232	0.21880	0.25000	1.03364
233	0.21880	0.28130	1.02109
234	0.21880	0.31250	0.98678
235	0.21880	0.34380	0.93913
236	0.21880	0.37500	0.88780
237	0.21880	0.40630	0.84061
238	0.21880	0.43750	0.80264
239	0.21880	0.46880	0.77648
240	0.21880	0.50000	0.76310
241	0.21880	0.53130	0.76248
242	0.21880	0.56250	0.77383
243	0.21880	0.59380	0.79559
244	0.21880	0.62500	0.82503
245	0.21880	0.65630	0.85794
246	0.21880	0.68750	0.88860
247	0.21880	0.71880	0.91058
248	0.21880	0.75000	0.91859
249	0.21880	0.78130	0.91058
250	0.21880	0.81250	0.88860
251	0.21880	0.84380	0.85794
252	0.21880	0.87500	0.82503
253	0.21880	0.90630	0.79559
254	0.21880	0.93750	0.77383
255	0.21880	0.96880	0.76248
256	0.25000	0.00000	[illegible]
257	0.25000	0.03130	0.77162

258	0.25000	0.06250	[illegible]
259	0.25000	0.09380	[illegible]
260	0.25000	0.12500	[illegible]
261	0.25000	0.15630	[illegible]
262	0.25000	0.18750	[illegible]
263	0.25000	0.21880	1.32701
264	0.25000	0.25000	[illegible]
265	0.25000	0.28130	1.32701
266	0.25000	0.31250	1.23037
267	0.25000	0.34380	1.11767
268	0.25000	0.37500	1.00136
269	0.25000	0.40630	0.90215
270	0.25000	0.43750	0.82548
271	0.25000	0.46880	0.77162
272	0.25000	0.50000	0.73856
273	0.25000	0.53130	0.72384
274	0.25000	0.56250	0.72504
275	0.25000	0.59380	0.73956
276	0.25000	0.62500	0.76410
277	0.25000	0.65630	0.79407
278	0.25000	0.68750	0.82331
279	0.25000	0.71880	0.84485
280	0.25000	0.75000	0.85279
281	0.25000	0.78130	0.84485
282	0.25000	0.81250	0.82331
283	0.25000	0.84380	0.79407
284	0.25000	0.87500	0.76410
285	0.25000	0.90630	0.73956
286	0.25000	0.93750	0.72504
287	0.25000	0.96880	0.72384
288	0.28130	0.00000	0.71625
289	0.28130	0.03130	0.76654
290	0.28130	0.06250	0.84767
291	0.28130	0.09380	0.96834
292	0.28130	0.12500	1.13811
293	0.28130	0.15630	1.36121
294	0.28130	0.18750	1.61977
295	0.28130	0.21880	1.84911
296	0.28130	0.25000	1.94479
297	0.28130	0.28130	1.84912
298	0.28130	0.31250	1.61978
299	0.28130	0.34380	1.36121
300	0.28130	0.37500	1.13811
301	0.28130	0.40630	0.96834
302	0.28130	0.43750	0.84767
303	0.28130	0.46880	0.76654
304	0.28130	0.50000	0.71625
305	0.28130	0.53130	0.69026
306	0.28130	0.56250	0.68374
307	0.28130	0.59380	[illegible]
308	0.28130	0.62500	0.71368
309	0.28130	0.65630	0.74124
310	0.28130	0.68750	0.76917
311	0.28130	0.71880	0.79014
312	0.28130	0.75000	[illegible]

ORIGINAL PAGE IS
OF POOR QUALITY

313	0.28130	0.78130	[illegible]
314	0.28130	0.81250	[illegible]
315	0.28130	0.84380	[illegible]
316	0.28130	0.87500	0.71368
317	0.28130	0.90630	0.69285
318	0.28130	0.93750	[illegible]
319	0.28130	0.96880	0.69026
320	0.31250	0.00000	0.69643
321	0.31250	0.03130	0.75941
322	0.31250	0.06250	0.86301
323	0.31250	0.09380	1.02568
324	0.31250	0.12500	1.27658
325	0.31250	0.15630	1.65243
326	0.31250	0.18750	2.17165
327	0.31250	0.21880	2.73105
328	0.31250	0.25000	2.99927
329	0.31250	0.28130	2.73105
330	0.31250	0.31250	2.17166
331	0.31250	0.34380	1.65243
332	0.31250	0.37500	1.27659
333	0.31250	0.40630	1.02568
334	0.31250	0.43750	0.86301
335	0.31250	0.46880	0.75941
336	0.31250	0.50000	0.69668
337	0.31250	0.53130	0.66348
338	0.31250	0.56250	0.65266
339	0.31250	0.59380	0.65921
340	0.31250	0.62500	0.67870
341	0.31250	0.65630	0.70591
342	0.31250	0.68750	0.73416
343	0.31250	0.71880	0.75568
344	0.31250	0.75000	0.76373
345	0.31250	0.78130	0.75568
346	0.31250	0.81250	0.73416
347	0.31250	0.84380	0.70591
348	0.31250	0.87500	0.67870
349	0.31250	0.90630	0.65921
350	0.31250	0.93750	0.65266
351	0.31250	0.96880	0.66348
352	0.34380	0.00000	0.68009
353	0.34380	0.03130	0.74886
354	0.34380	0.06250	0.86574
355	0.34380	0.09380	1.05803
356	0.34380	0.12500	1.37622
357	0.34380	0.15630	1.91137
358	0.34380	0.18750	2.79374
359	0.34380	0.21880	3.99649
360	0.34380	0.25000	4.69084
361	0.34380	0.28130	3.99649
362	0.34380	0.31250	2.79374
363	0.34380	0.34380	1.91137
364	0.34380	0.37500	1.37622
365	0.34380	0.40630	1.05803
366	0.34380	0.43750	0.86574
367	0.34380	0.46880	0.74887

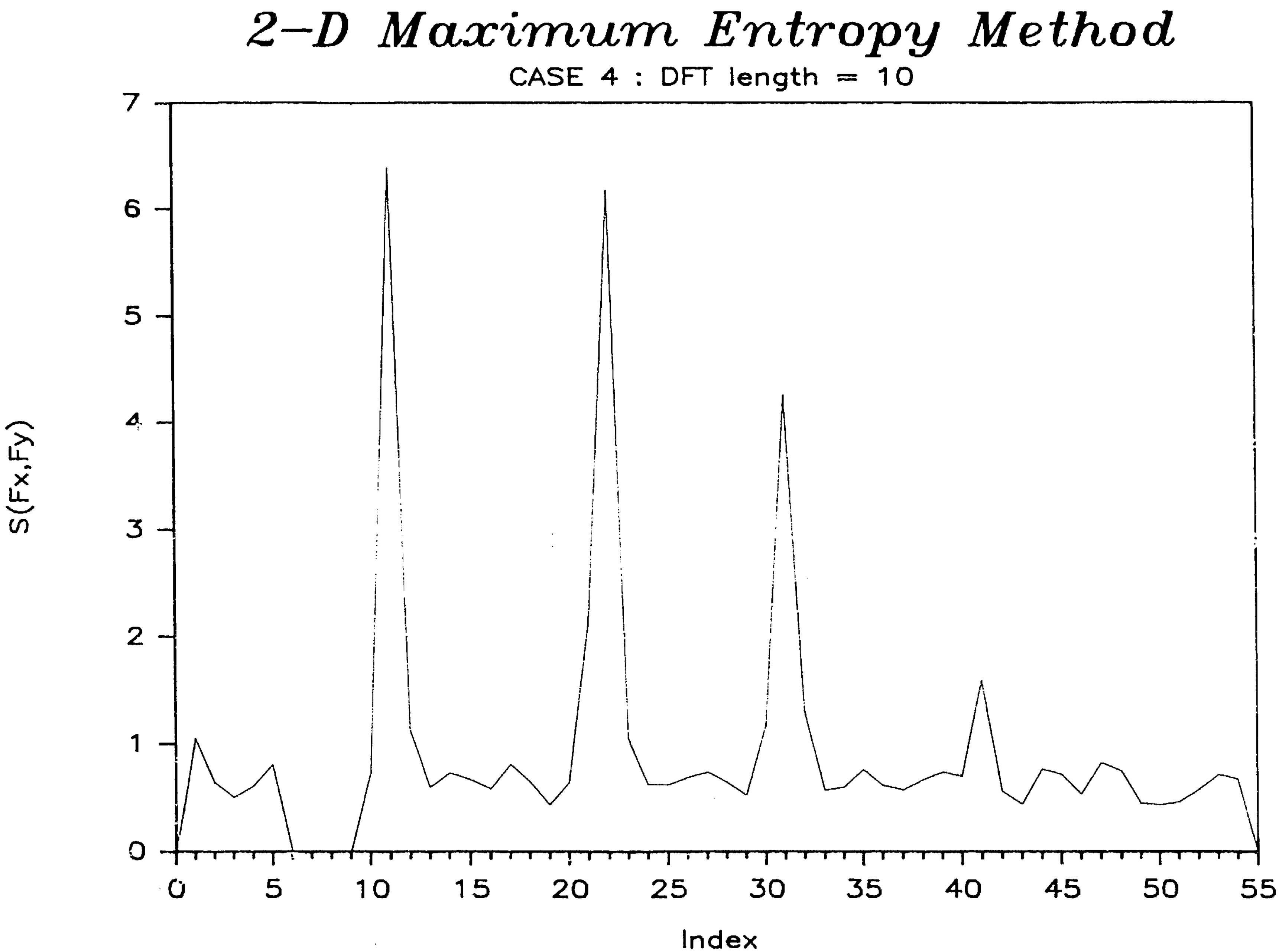
368	0.34380	0.50000	[illegible]
369	0.34380	0.53130	[illegible]
370	0.34380	0.56250	[illegible]
371	0.34380	0.59380	0.64019
372	0.34380	0.62500	0.66148
373	0.34380	0.65630	0.69136
374	0.34380	0.68750	0.72265
375	0.34380	0.71880	0.74669
376	0.34380	0.75000	0.75579
377	0.34380	0.78130	0.74669
378	0.34380	0.81250	0.72266
379	0.34380	0.84380	0.69136
380	0.34380	0.87500	0.66140
381	0.34380	0.90630	0.64019
382	0.34380	0.93750	0.63291
383	0.34380	0.96880	0.64434
384	0.37500	0.00000	0.66658
385	0.37500	0.03130	0.73439
386	0.37500	0.06250	0.85266
387	0.37500	0.09380	1.05221
388	0.37500	0.12500	1.39413
389	0.37500	0.15630	2.00246
390	0.37500	0.18750	3.10631
391	0.37500	0.21880	4.85310
392	0.37500	0.25000	6.03159
393	0.37500	0.28130	4.85311
394	0.37500	0.31250	3.10632
395	0.37500	0.34380	2.00246
396	0.37500	0.37500	1.39413
397	0.37500	0.40630	1.05221
398	0.37500	0.43750	0.85266
399	0.37500	0.46880	0.73439
400	0.37500	0.50000	0.66658
401	0.37500	0.53130	0.63305
402	0.37500	0.56250	0.62486
403	0.37500	0.59380	0.63649
404	0.37500	0.62500	0.66343
405	0.37500	0.65630	0.70010
406	0.37500	0.68750	0.73847
407	0.37500	0.71880	0.76813
408	0.37500	0.75000	0.77938
409	0.37500	0.78130	0.76813
410	0.37500	0.81250	0.73847
411	0.37500	0.84380	0.70010
412	0.37500	0.87500	0.66343
413	0.37500	0.90630	0.63649
414	0.37500	0.93750	0.62486
415	0.37500	0.96880	0.63305
416	0.40630	0.00000	0.65616
417	0.40630	0.03130	0.71644
418	0.40630	0.06250	0.82461
419	0.40630	0.09380	1.00803
420	0.40630	0.12500	1.31984
421	0.40630	0.15630	1.86311
422	0.40630	0.18750	2.80[illegible]

ORIGINAL PAGE IS
OF POOR QUALITY

423	0.40630	[illegible]	[illegible]
424	0.40630	0.25000	[illegible]
425	0.40630	0.28130	[illegible]
426	0.40630	0.31250	[illegible]
427	0.40630	0.34380	[illegible]
428	0.40630	0.37500	1.31985
429	0.40630	0.40630	[illegible]
430	0.40630	0.43750	0.82461
431	0.40630	0.46880	0.71644
432	0.40630	0.50000	0.65614
433	0.40630	0.53130	0.62940
434	0.40630	0.56250	0.62841
435	0.40630	0.59380	0.64863
436	0.40630	0.62500	0.68622
437	0.40630	0.65630	0.73563
438	0.40630	0.68750	0.78743
439	0.40630	0.71880	0.82795
440	0.40630	0.75000	0.84346
441	0.40630	0.78130	0.82795
442	0.40630	0.81250	0.78743
443	0.40630	0.84380	0.73563
444	0.40630	0.87500	0.68622
445	0.40630	0.90630	0.64863
446	0.40630	0.93750	0.62841
447	0.40630	0.96880	0.62940
448	0.43750	0.00000	0.64878
449	0.43750	0.03130	0.69636
450	0.43750	0.06250	0.78615
451	0.43750	0.09380	0.93773
452	0.43750	0.12500	1.18596
453	0.43750	0.15630	1.58721
454	0.43750	0.18750	2.20201
455	0.43750	0.21880	2.95524
456	0.43750	0.25000	3.35547
457	0.43750	0.28130	2.95524
458	0.43750	0.31250	2.20201
459	0.43750	0.34380	1.58721
460	0.43750	0.37500	1.18596
461	0.43750	0.40630	0.93773
462	0.43750	0.43750	0.78615
463	0.43750	0.46880	0.69636
464	0.43750	0.50000	0.64878
465	0.43750	0.53130	0.63286
466	0.43750	0.56250	0.64322
467	0.43750	0.59380	0.67717
468	0.43750	0.62500	0.73236
469	0.43750	0.65630	0.80377
470	0.43750	0.68750	0.87990
471	0.43750	0.71880	0.94094
472	0.43750	0.75000	0.96469
473	0.43750	0.78130	0.94094
474	0.43750	0.81250	0.87990
475	0.43750	0.84380	0.80377
476	0.43750	0.87500	0.73235
477	0.43750	0.90630	0.67717

ORIGINAL PAGE IS OF POOR QUALITY

478	0.43750	0.93750	[illegible]
479	0.43750	0.96880	[illegible]
480	0.46880	0.00000	[illegible]
481	0.46880	0.03130	[illegible]
482	0.46880	0.06250	0.74360
483	0.46880	0.09380	[illegible]
484	0.46880	0.12500	1.03827
485	0.46880	0.15630	1.30431
486	0.46880	0.18750	1.65917
487	0.46880	0.21880	2.02389
488	0.46880	0.25000	2.19245
489	0.46880	0.28130	2.02389
490	0.46880	0.31250	1.65917
491	0.46880	0.34380	1.30431
492	0.46880	0.37500	1.03827
493	0.46880	0.40630	0.85840
494	0.46880	0.43750	0.74360
495	0.46880	0.46880	0.67605
496	0.46880	0.50000	0.64439
497	0.46880	0.53130	0.64261
498	0.46880	0.56250	0.66856
499	0.46880	0.59380	0.72249
500	0.46880	0.62500	0.80512
501	0.46880	0.65630	0.91353
502	0.46880	0.68750	1.03399
503	0.46880	0.71880	1.13512
504	0.46880	0.75000	[illegible]
505	0.46880	0.78130	1.13512
506	0.46880	0.81250	1.03398
507	0.46880	0.84380	0.91353
508	0.46880	0.87500	[illegible]
509	0.46880	0.90630	[illegible]
510	0.46880	0.93750	0.66856
511	0.46880	0.96880	0.64261
512	0.50000	0.00000	0.64292
513	0.50000	0.03130	0.65751
514	0.50000	0.06250	0.70293
515	0.50000	0.09380	0.78398
516	0.50000	0.12500	0.90740
517	0.50000	0.15630	1.07693
518	0.50000	0.18750	1.27961
519	0.50000	0.21880	1.46366
520	0.50000	0.25000	1.54152
521	0.50000	0.28130	1.46366
522	0.50000	0.31250	1.27961
523	0.50000	0.34380	1.07693
524	0.50000	0.37500	0.90740
525	0.50000	0.40630	0.78398
526	0.50000	0.43750	0.70293
527	0.50000	0.46880	0.65751
528	0.50000	0.50000	0.00000


2-D Maximum Entropy Method
CASE 4 : DFT length = 10
S(Fx,Fy)
Index
0
1
2
3
4
5
6
7
0
5
10
15
20
25
30
35
40
45
50
55

CASE 4:

NUMBER OF SINUSOIDS = 3
NOISE POWER = 6.0

SINUSOID = 1
POWER = 1.0
XFREQ = 0.1
YFREQ = 0.1

SINUSOID = 2
POWER = 1.0
XFREQ = 0.3
YFREQ = 0.1

SINUSOID = 3
POWER = 1.0
XFREQ = 0.2
YFREQ = 0.2

ACF MATRIX = 7 x 7

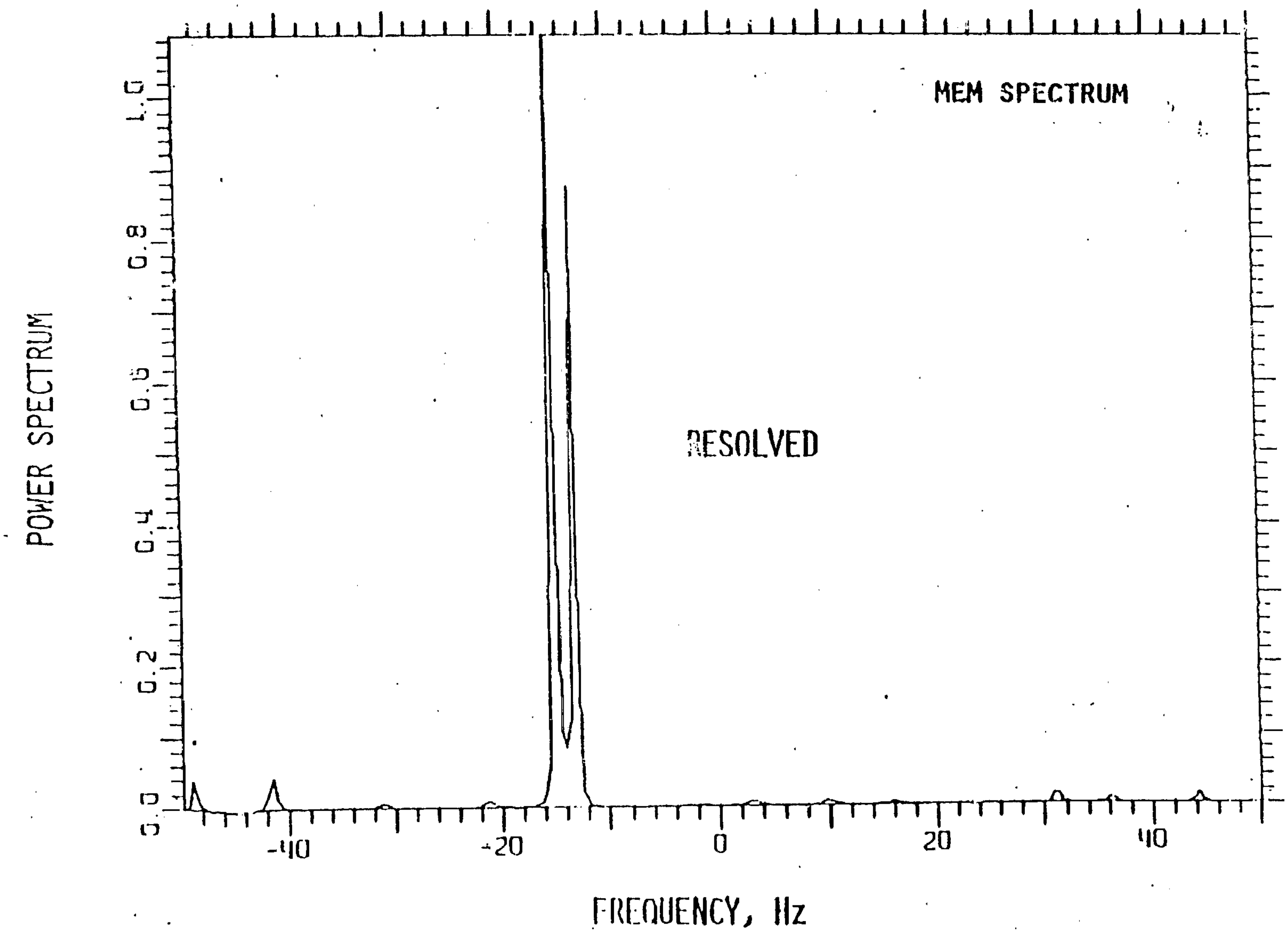
DFT LENGTH = 10

ORIGINAL PAGE IS OF POOR QUALITY

INDEX	Fx	Fy	S(Fx,Fy)
0	0.00000	0.00000	0.00000
1	0.00000	0.10000	1.05745
2	0.00000	0.20000	0.64671
3	0.00000	0.30000	0.50208
4	0.00000	0.40000	0.61538
5	0.00000	0.50000	0.81946
6	0.00000	0.60000	0.00000
7	0.00000	0.70000	0.00000
8	0.00000	0.80000	0.00000
9	0.00000	0.90000	0.00000
10	0.10000	0.00000	0.73649
11	0.10000	0.10000	6.40696
12	0.10000	0.20000	1.13217
13	0.10000	0.30000	0.59657
14	0.10000	0.40000	0.73546
15	0.10000	0.50000	0.66967
16	0.10000	0.60000	0.58012
17	0.10000	0.70000	0.81931
18	0.10000	0.80000	0.65055
19	0.10000	0.90000	0.43186
20	0.20000	0.00000	0.64967
21	0.20000	0.10000	2.13862
22	0.20000	0.20000	6.19231
23	0.20000	0.30000	1.04869
24	0.20000	0.40000	0.62238
25	0.20000	0.50000	0.61939
26	0.20000	0.60000	0.69615
27	0.20000	0.70000	0.74584

28	0.20000	0.80000	[illegible]
29	0.20000	0.90000	0.52[illegible]
30	0.30000	0.00000	1.1[illegible]
31	0.30000	0.10000	4.27[illegible]
32	0.30000	0.20000	1.30108
33	0.30000	0.30000	0.57365
34	0.30000	0.40000	0.60545
35	0.30000	0.50000	0.77027
36	0.30000	0.60000	0.61815
37	0.30000	0.70000	0.57268
38	0.30000	0.80000	0.67831
39	0.30000	0.90000	0.74797
40	0.40000	0.00000	0.69804
41	0.40000	0.10000	1.60545
42	0.40000	0.20000	0.56048
43	0.40000	0.30000	0.43885
44	0.40000	0.40000	0.77655
45	0.40000	0.50000	0.72143
46	0.40000	0.60000	0.53407
47	0.40000	0.70000	0.83245
48	0.40000	0.80000	0.75012
49	0.40000	0.90000	0.45024
50	0.50000	0.00000	0.43705
51	0.50000	0.10000	0.46899
52	0.50000	0.20000	0.58416
53	0.50000	0.30000	0.72468
54	0.50000	0.40000	0.67774
55	0.50000	0.50000	0.00000

RESULT WITH MEM



```
      PROGRAM MEM2D

c*****************************************************************************
c         THE 2-D MAXIMUM ENTROPY METHOD PSD ESTIMATOR:
c
c         To obtain the 2-D mem solution for real
c         data (real symmetric acf), using direct dft
c         computation. The MEM PSD does not required that
c         the known lags of the 2-D acf have a uniform support
c         grid; arbitrary lags can be used as the constraints.
c
c         xlam   = lambda coefficient array.
c         xold   = old lambda array required for
c                  beta computation.
c         r      = known acf value.
c         px     = computed acf values or correction acf.
c         gap    = logical array speciyingwhat gaps
c                  if any, exist in the acf.
c         n      = dft length used in the iterations.
c         n2     = n/2
c         n21    = n2 + 1
c         n4     = n * n
c         n1     = max +ve (x,y) index for the smallest
c                  square region containing 'R'.
c         mn     = max acf array size for this program(25).
c         mn2    = center point of the acf array.
c         mn3    = mn2 + 1
c         mn4    = 2 * mn2
c         m1     = min (x,y) index for known region.
c         m2     = max (x,y) index for known region.
c         sclf   = scale factor.
c         ztst   = error level.
c
c*****************************************************************************

      real*4  x1, x2, z, den
      real*4  xlam(25,25), xold(25,25), r(25,25), px(25,25)
      common n, n2, n21, n4, n1, mn, mn2, mn3, mn4, m1, m2

1     format(10x,' nitr  = ',i3,5x,' error = ',e12.4,/)
2     format(10x,' alpha = ',e14.6,/)
3     format(10x,' beta  = ',e14.6,/)

      mn = 25
      mn2 = (mn - 1) / 2 + 1

      call acf2d(r)

      rmn2 = r(mn2,mn2)

      do i = m1,m2
      do j = m1,m2
         r(i,j) = r(i,j) / rmn2
      enddo
```

ORIGINAL PAGE IS OF POOR QUALITY

```
      enddo

      zold  = 1.0E30
      zold1 = 1.0E30

      write(*, 90)
90    format(/, 10x, 'enter error level (ztst) ',/ )

      read *, ztst

      write(*,95)ztst
95    format(/, 5x, ' error level (ztst) =  ', f10.8, / )

      sclf  = 0.5
      alpha = 0.0
      beta  = 0.0

      do i = 1,mn
      do j = 1,mn
         xold(i,j) = 0.0
         xlam(i,j) = 0.0
         px(i,j)   = 0.0
      enddo
      enddo

      xlam(mn2,mn2) = 1.0
      xold(mn2,mn2) = xlam(mn2,mn2)

      den = 0.0
      do i = m1,m2
      do j = m1,m2
         den = den + r(i,j) * r(i,j)
      enddo
      enddo

      den = den - 1.0

      nitr = 0

1000  call ft(px,xlam,x1,x2)

      nitr = nitr + 1
      z = 0.0
      do i = m1,m2
      do j = m1,m2
         px(i,j) = r(i,j) - px(i,j)
         z = z + px(i,j) * px(i,j)
      enddo
      enddo

      z = z / den

      write (*,1) nitr, z

      if ( z .lt. ztst ) go to 9000
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
      if ( alpha .gt. 0.9999 ) go to 9000

      if ( z .ge. zold .or. beta .ne. 0.0) go to 970

      go to 971

970   alpha = ( 1.0 + alpha ) / 2.0
      sclf  = sclf / 2.0
      zold1 = zold
971   zold  = z

      call ft2 ( px,x3 )

      xx = 1.0 + sclf * x2 / x3
      alpha = amax1 ( alpha, xx )

      write ( *,2 ) alpha

      call ft3 ( xold, px, alpha, xlam )

      call ft2( xlam, x4 )

      if ( x4 .gt. 0.0 ) go to 972

      bmin  = - x4 / ( x1 - x4 )
      beta  = ( 1.0 + ( 1.0 - sclf ) * ( 1.0 / bmin - 1.0 )) * bmin
      alpha = ( 1.0 + alpha ) / 2.0

      go to 973

972   beta = 0.0

973   write ( *, 3 ) beta

      do i = m1, m2
      do j = m1, m2
      xlam( i, j ) = beta * XOLD( i, j ) + ( 1.0 - beta ) * xlam( i,j )
      xold(i,j) = xlam( i, j)
      enddo
      enddo
      go to 1000
9000  continue

      write(*, 9900)
9900  format( 1x, ' convergence achieved ! test in progress,
     *  patience.... ')

      n   = 4  * n
      n2  = n  / 2
      n21 = n2 + 1
      n4  = n  * n

      call ft( px, xlam, x1, x2 )
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
      z = 0.0
      do i = m1, m2
      do j = m1, m2
         z = z + ( r( i, j ) - px( i, j ) ) ** 2
      enddo
      enddo
      z = z / den

      if ( z .le. ztst ) go to 974

      write ( *, 9002 ) z
9002  format(/,  ' error level too high : ', e12.4, / )

      go to 975

974   write ( *, 9001 ) z
9001  format(/, ' good solution. error is : ', e12.4,// )

975   continue

cc       do i = m1, m2
cc       do j = m1, m2

cc       write( *, 9300 ) i, j, xlam( i, j )
9300  format( 1x ,'lambda( ', I2,1X,I3,' ) =  ', E14.6, /)

cc       write( 7, 9400 ) xlam( i, j )
9400  format( 1x, f10.7 )

cc       enddo
cc       enddo

      call print( xlam, m2-m1+1 )

      stop
      end

C*****************************************************************
      subroutine acf2d( r )
C*****************************************************************

      integer n, n1, mn, mn2, m1, m2
      real*4  r(25, 25), p(20), xfreq(20), yfreq(20), noise, t(25,25)
      common  n, n2, n21, n4, n1, mn, mn2, mn3, mn4, m1, m2

      pi = 4.0 * atan(1.0)

      write(*, 110)
110   format(/, 10x, 'enter number of sinusoids',/)

      read*, nsin

      write( *, 112) nsin
112   format(/, 5x, ' number of sinusoids = ', i2,/)
```

```
      do i = 1, nsin

      write(*, 130)
130   format(/, 10x, 'enter power, xfreq, yfreq ( abs(freq) =< 0.5 )',/)

      read *,p(i),  xfreq(i), yfreq(i)

      write( *,132 ) i, p(i), xfreq(i), yfreq(i)
132   format(/, 5x,' sinusoid = ', i3, 3x, ' power = ', f8.2, 5x,
     * ' xfreq = ', f7.5, 5x, ' yfreq = ', f7.5, / )

      enddo

      write(*, 150)
150   format(/, 10x, 'enter noise power',/)

      read*, noise

      write( *,152 )noise
152   format(/, 5x, ' noise power = ', f7.3,/)

      write(*, 170)
170   format(/, 10x, 'enter dimension of smallest square containing
     * acf (must be odd)',/)

      read*, n12

      write( *, 172 )n12, n12
172   format(/, 5x, ' acf matrix ', i2, ' BY   ', i2,/)

      n1  = ( n12 - 1 ) / 2
      m1  = mn2 - n1
      m2  = mn2 + n1
      mn3 = mn2 + 1
      mn4 = 2   * mn2

      write(*, 197)
197   format(/, 10x, 'dft length (must be even)',/)

      read*, n

      write( *, 199 )n
199   format(/, 5x, ' dft length = ', i5, //)

      n2  = n  / 2
      n21 = n2 + 1
      n4  = n  * n

      do 30 ns = 1, nsin
            wx = 2. * pi * xfreq( ns )
            wy = 2. * pi * yfreq( ns )
      do 30 i = m1, m2
            ia = i - mn2
      do 30 j = m1, m2
            ja = j - mn2
```

```
            if ( ns .eq. 1 )    r( i, j ) = 0.0
            r(i,j) = r(i,j)+p(ns)*cos(wx*ia+wy*ja)

 30     continue

        r( mn2, mn2 ) = r( mn2, mn2 ) + noise
        DO I = M1, M2

           WRITE(*,911)( R(I,J), J=M1,M2)
 911    FORMAT( 9(F10.4))
        ENDDO
        return
        end
c*********************************************************************
        subroutine ft2(px,xmin)
c*********************************************************************
        real*4 px(25,25)
        complex z0,z1,z2,z3,z4,z5,z6,z7
        common n,n2,n21,n4,n1,mn,mn2,mn3,mn4,m1,m2

        pin = 8.0 *atan(1.0)/float(n)

        x00 = 0.0
        x22 = 0.0
        do 100 i = m1,m2
        do 100 j = m1,m2
        x00 = x00 + px(i,j)
 100    x22 = x22+px(i,j)*((-1.0)**(i+j-mn4))
        xmin = amin1(x00,x22)

        x0 = px(mn2,mn2)
        z0 = cmplx(cos(pin),sin(pin))
        z1 = (1.0,0.0)
        z6 = cmplx(cos(pin*n1),-sin(pin*n1))
        z7 = (1.0,0.0)

        do 600 k = 1,n21
           z2 = z1
           x1 = 0.0
        do 200 m = mn3,m2
           x1 = x1+px(m,mn2)*(real(z2))
 200       z2 = z2*z1

        z3 = (1.0,0.0)
        do 500 l = 1,n
           if (k*l .eq. 1) go to 500
           if (k. eq. 1 .and. l .gt. n21) go to 601
           if (k. eq. n21 .and. l .gt. n2) go to 600
           x2 = 0.0
           z4 = z7
        do 400 m = m1,m2
           z5 = z3
        do 300 nn = mn3,m2
           x2 = x2+px(m,nn)*real(z4*z5)
 300       z5 = z5*z3
```

ORIGINAL PAGE IS
OF POOR QUALITY

ORIGINAL PAGE IS
OF POOR QUALITY

```
      if(m.eq.m2) go to 400
      z4 = z4*z1
400   continue

      xk1 = x0+2.0*(x1+x2)
      xmin = amin1(xmin,xk1)
500   z3 = z3*z0
601   z1 = z1*z0
      z7 = z6*conjg(z4)
600   continue

      return
      end
c**************************************************************
      subroutine ft3( xold, px, alpha, xlam )
c**************************************************************
      real*4 px( 25, 25 ), xlam( 25, 25 ), xold( 25, 25 )
      complex z0, z1,z2,z3,z4,z5,z6,z7
      common n, n2,n21,n4,n1,mn,mn2,mn3,mn4,m1,m2

      xn2 = n4
      pin = 8.0*atan( 1.0)/float(n)

      x00 = 0.0
      x22 = 0.0
      y00 = 0.0
      y22 = 0.0
      do 100 i = m1,m2
      do 100 j= m1,m2
      x00 = x00+px(i,j)
      x22 = x22+px(i,j)*((-1.0)**(i+j-mn4))
      y00 = y00+ xold(i,j)
      y22 = y22+xold(i,j)*(( -1.0)**(i+j-mn4))
100   xlam( i,j )= 0.0
      x00 =1.0/y00+(1.0 - alpha)* x00
      x00 = 1.0/x00
      x22 = 1.0 / y22 + (1.0-alpha)*x22
      x22  = 1.0/X22

      do 150 m = m1, m2
      if (m.lt.mn2) go to 985
      nn1 = mn2
      go to 986
985   nn1 = mn3
986   continue
      do 150 nn = nn1, m2
150   xlam(m,nn)= x00 + x22 *(( -1.0)**( m+nn-mn4))

      x0 = px(mn2,mn2)
      y0 = xold(mn2, mn2)
      z0 = cmplx(cos(pin), sin(pin))
      z1 = ( 1.0, 0.0)
      z6 = cmplx(cos(pin*n1), -sin(pin*n1))
      z7 = (1.0, 0.0)
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
      do 800 k = 1, n21

      z2 = z1
      x1 = 0.0
      y1 = 0.0
      do 200 m = mn3, m2
      xx = real(z2)
      x1 = x1 + px( m, mn2 )*xx
      y1 = y1 + xold ( m,mn2 ) * xx
200   z2 = z2 * z1

      z3 = ( 1.0, 0.0 )
      do 700 l = 1,n
      if ( k*l.eq.1) go to 700
      if ( k.eq.1.and.l.gt.n21 ) go to 801
      if ( k.eq.n21.and.l.gt.n2 ) go to 800

      x2 = 0.0
      y2 = 0.0
      z4 = z7
      do 400 m = m1, m2
      z5 = z3
      do 300 nn = mn3, m2
      xx = real( z4* z5 )
      x2 =  x2 + px(m,nn)*xx
      y2 = y2 + xold( m, nn )*xx
300   z5=z5*z3
      if ( m.eq.m2 ) go to 400
      z4= z4 * z1
400   continue

      xkl = x0 + 2.0*( x1+ x2)
      ykl = y0 + 2.0 * ( y1 + y2 )
      xkl = 1.0 / ykl + ( 1.0 - alpha ) * xkl
      xkl = 1.0 / xkl

      z2 = z7
      do 600 m = m1, m2
      if ( m.lt.mn2 ) go to 987
      z5 = ( 1.0, 0.0 )
      nn1 = mn2
      go to 988
987   z5 = z3
      nn1 = mn3
988   continue
      do 500 nn = nn1,m2
      if (k*l.eq.n21 ) go to 989
      xlam (m,nn) = xlam(m,nn)+2.0*xkl* real(z2*z5)
      go to 500
989   xlam(m,nn)=xlam(m,nn)+xkl*real(z2*z5)
500   z5 = z5 * z3
600   z2 = z2 * z1
700   z3 = z3 * z0
801   z1 = z1 * z0
```

```
      z0 = cmplx( cos(pin),sin(pin) )
      z1 = ( 1.0, 0.0)
      z6 = cmplx( cos(pin*n1), -sin(pin*n1) )
      z7 = ( 1.0, 0.0 )

      do 800 k = 1, n21
      z2 = z1
      x1 = 0.0
      do 200 m = mn3, m2
      x1 = x1 + xlam (m,mn2)*real(z2)
 200  z2 = z2 * z1

      z3 =( 1.0, 0.0 )
      do 700 l = 1, n
      if ( k*l.eq.1 ) go to 700
      if (k.eq.1.and.l.gt.n21 ) go to 801
      if ( k.eq.n21.and.l.gt.n2 ) go to 800

      x2 = 0.0
      z4 = z7
      do 400 m = m1, m2
      z5 = z3
      do 300 nn = mn3, m2
      x2 = x2 + xlam(m,nn) * real( z4*z5)
 300  z5=z5*z3
      if ( m.eq.m2 ) go to 400
      z4 = z4 * z1
 400  continue

      xkl = x0 + 2.0 *( x1+x2 )
      if ( xkl.gt.0.0 ) go to 994
      type 222
 222  format ( 5x, ' bad solution, f( lambda ) < 0.
     *( CHECK POINT NO. 2    ( SUBROUTINE FT))   ' )
      stop

 994  continue
      xmin1 = amin1( xmin1, xkl )
      xmin2 = amax1( xmin2, xkl )
      xkl = 1.0 / xkl

      z2 = z7
      do 600 m= m1, m2
      if ( m.lt.mn2 ) go to 995
      z5 = ( 1.0, 0.0 )
      nn1 = mn2
      go to 996
 995  z5 = z3
      nn1 = mn3
 996  continue
      do 500 nn = nn1, m2
      if ( k*l.eq.n21 ) go to 997
      px( m,nn ) = px( m, nn ) +2.0 * xkl * real( z2*z5 )
      go to 500
 997  px( m,nn ) = px( m,nn ) + xkl*real( z2*z5 )
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
500    z5 = z5 * z3
600    z2 = z2 * z1
700    z3 = z3 * z0
801    z1 = z1 * z0
       z7 = z6 * conjg(z4)
800    continue

       do 900 m= m1,m2
       do 900 nn= m1,mn2
900    px( m,nn ) = px( m1+m2-m, m1+m2-nn )

       px0= px( mn2, mn2 )
       xx= px0 / xn2
       do 1000 m = m1, m2
       do 1000 nn = m1, m2
       px( m,nn ) = px( m,nn)/px0
1000   xlam(m,nn)= xlam(m,nn)*xx

       xmin2 = 1.0 / ( xmin2 * xx )
       xmin1 = xmin1 * xx

       return
       end
```

ORIGINAL PAGE IS
OF POOR QUALITY

REFERENCES

(1) Burg, J.P., A New Analysis Technique for Time Series Data, NATO Adv. Study Inst. Signal Processing, Enschede, Netherlands (1968)

(2) Burg, J.P., Maximum Entropy Spectral Analysis, Ph.D. Dissertation, Stanford University, Stanford, Calif. (1975)

(3) Haykin, S. and S. Kesbi, "Prediction - Error Filtering and Maximum - Entropy Spectral Estimation" in Nonlinear Methods of Spectral Analysis, ed. S. Haykin, Springer-Verlag 1983.

(4) Strand, O.N., "Multichannel Complex Maximum Entropy (autoregressive) Analysis", IEEE Trans. on Automatic Control, vol. AC-22, no. 4, pp. 634-640, August 1977.

(5) Fougere, P.F., A Solution to the Problem of Spontaneous Line Splitting in Maximal Entropy Power Spectrum Analysis of Complex Signals, Proc. of the RADC Spectrum Estimation Workshop, May 1978.

(6) Maximal Entropy Spectral Analysis of Wallops Wind Wave Interaction Tank Data (N.L.Bonavito, et al to appear).