

IMPERIAL COLLEGE LONDON

DEPARTMENT OF MATHEMATICS

Time Series - Coursework 2

Author:

Tudor Trita Trita

Date: November 8, 2019

1 Question 1

1.1 Part A

After running the code for 10000 instances, I computed the mean and variance of the periodogram.

The sample means of the periodogram are 3.12282332, 3.38281217, 3.72564659 for the frequencies in the question.

The real spectra $S(f)$ are 3.08283449 3.38331927 3.72693768 for the frequencies in the question.

As the two results are very close to each other, we can conclude that the results in the question agree, that asymptotically as N goes to infinity, the expectation of the periodogram is equal to $S(f)$.

The sample variances of the periodogram are 9.43457345, 11.21072155, 14.34174298 for the frequencies in the question.

The real spectra squared are 9.50386847, 11.44684925, 13.89006449.

Again, as the results are close to each other, we can conclude that the results in the question agree, that asymptotically as N goes to infinity, the variance of the periodogram is equal to $S^2(f)$.

1.2 Part B

The correlation coefficients as in the question are -0.00165803, -0.02357127, 0.0029959 respectively. As these results are close to 0, we can conclude that my results agree with the large sample results given in the question, that for N large enough, the periodograms are pairwise approximately uncorrelated.

1.3 Part C

The histograms for the periodogram sequences are presented in blue, in figures 1, 3 and 5. The histograms for 10000 instances of the sdf multiplied by a chi-squared, 2d.f. distribution divided by 2, can be seen in figures 2, 4 and 6. We note that these histograms are very similar and we can therefore conclude that the result given in the question holds for n large enough.

2 Question 2

2.1 Part A

There is a stark contrast between Yule-Walker and non-YW methods of estimation. The periodogram and direct estimate have a very large variance where as the YW methods create smooth estimates even for 1 iteration.

The direct estimate seems to perform better for larger N than the periodogram. The tapering improves the YW estimate, as the graphs are closer to the sdf than without tapering.

2.2 Part B

Overall, all methods follow the sdf closely. All estimates become closer to the true sdf the larger N is.

The biggest difference in 'accuracy' can be seen for the periodogram and the YW without tapering. These improve the most between N=64 and N=1024.

The YW method with tapering performs well even for small N, suggesting that it is the best method for estimation.

2.3 Part C

Here again, we can see that the periodogram and direct estimates have a large variance, and that variance is proportional to the value of the sdf, which we saw in the Q1 part A.

The YW methods do not give the right answer as the wrong model was assumed, though their variance is very small.

2.4 Part D

The periodogram and direct estimate methods perform well in this, though they seem more 'unstable' the larger N is. This is due to there being more timepoints, so at each timestep there will be more variability and this can be seen in their changes. Again, the YW methods, do not give the right answer, as the wrong model has been assumed.

3 Code

```
1 """Time Series CW2 code used
2
3 Tudor Trita Trita
4
5 Note: Functions are declared before main code executes at the bottom
6 in __init__=='__main__' part.
7
8 Functions are created as and when needed in CW2.
9 """
10
11 import numpy as np
12 import matplotlib.pyplot as plt
13 import numpy.random as rnd
14 import numpy.linalg as lng
15 import scipy.stats
16
17
18 def arprocess(Nr, t):
19     """Return N instances of AR(2) process."""
20     epsilon = rnd.normal(0, 1, (Nr, t))
21     X = np.zeros((Nr, t))
22     X[:, 0] = (4/3)*epsilon[:, 0]
23     X[:, 1] = 0.5*X[:, 0] + 2/(np.sqrt(3)) * epsilon[:, 1]
24     for i in range(t-2):
25         X[:, i+2] = 0.75*X[:, i+1] - 0.5*X[:, i] + epsilon[:, i]
26     return X
27
28
29 def periodogram(Nr, X, f):
30     """Return Nr instances of the periodogram."""
31     N = len(X[0, :])
32     flen = len(f)
33     nums = np.linspace(1, N, N)
34     Sper = np.zeros((Nr, flen))
35     for i in range(flen):
36         a = X[:, :] * np.exp(-1j*2*np.pi*f[i]*nums)
37         Sper[:, i] = 1/N*abs(np.sum(a, axis=1))**2
38     return Sper
39
40
41 def sdf(f):
42     """Return sdf for flen number of frequencies of AR(2) process."""
43     flen = len(f)
44     Spectra = np.zeros(flen)
45     for i in range(flen):
46         Spectra[i] = (1/abs(1-(3/4)*np.exp(-1j*2*np.pi*f[i]))
47                     + (1/2)*np.exp(-1j*2*np.pi*f[i]*2))**2)
48     return Spectra
49
50
51 def q1parta(Sper, f):
52     """Return sample mean and variance."""
```

```
53     Sper.transpose()
54     sm = np.mean(Sper, axis=0)
55     sv = np.var(Sper, axis=0)
56     return sm, sv
57
58
59 def q1partb(Sper):
60     """Return sample correlation."""
61     pearson = np.zeros((len(Sper[0, :]), 2))
62     pearson[0, :] = scipy.stats.pearsonr(Sper[:, 0], Sper[:, 1])
63     pearson[1, :] = scipy.stats.pearsonr(Sper[:, 0], Sper[:, 2])
64     pearson[2, :] = scipy.stats.pearsonr(Sper[:, 1], Sper[:, 2])
65     pearson.transpose()
66     return pearson
67
68
69 def q1partc(spectra, Sper):
70     """Plot histograms."""
71     Nr = len(Sper[:, 0])
72     epsilon = rnd.chisquare(2, (Nr, 3))
73     pdf = np.zeros((Nr, 3))
74     pdf[:, 0] = spectra[0]*epsilon[:, 0]/2
75     pdf[:, 1] = spectra[1]*epsilon[:, 1]/2
76     pdf[:, 2] = spectra[2]*epsilon[:, 2]/2
77
78     fig1 = plt.figure(figsize=(10, 5))
79     plt.hist(Sper[:, 0], bins=200, color='#0504aa')
80     plt.xlabel('Periodogram')
81     plt.ylabel('No. of appearances')
82     plt.title("Name: Tudor Trita Trita, CID:01199397 \n Figure 1:
Histogram of Periodogram for f_12")
83     fig1.savefig("fig1.png")
84     plt.show()
85
86     fig2 = plt.figure(figsize=(10, 5))
87     plt.hist(pdf[:, 0], bins=200, color='r')
88     plt.xlabel('PDF*chi-squared/2')
89     plt.ylabel('No. of appearances')
90     plt.title("Name: Tudor Trita Trita, CID:01199397 \n Figure 2:
Histogram of PDF*chi-squared/2 for f_12")
91     fig2.savefig("fig2.png")
92     plt.show()
93
94     fig3 = plt.figure(figsize=(10, 5))
95     plt.hist(Sper[:, 1], bins=200, color='#0504aa')
96     plt.xlabel('Periodogram')
97     plt.ylabel('No. of appearances')
98     plt.title("Name: Tudor Trita Trita, CID:01199397 \n Figure 3:
Histogram of Periodogram for f_13")
99     fig3.savefig("fig3.png")
100    plt.show()
101
102    fig4 = plt.figure(figsize=(10, 5))
103    plt.hist(pdf[:, 1], bins=200, color='r')
104    plt.xlabel('PDF*chi-squared/2')
```

```

105     plt.ylabel('No. of appearances')
106     plt.title("Name: Tudor Trita Trita , CID:01199397 \n Figure 4:
Histogram of PDF*chi-squared/2 for f_13")
107     fig4.savefig("fig4.png")
108     plt.show()
109
110     fig5 = plt.figure(figsize=(10, 5))
111     plt.hist(Sper[:, 2], bins=200, color='#0504aa')
112     plt.xlabel('Periodogram')
113     plt.ylabel('No. of appearances')
114     plt.title("Name: Tudor Trita Trita , CID:01199397 \n Figure 5:
Histogram of Periodogram for f_14")
115     fig5.savefig("fig5.png")
116     plt.show()
117
118     fig6 = plt.figure(figsize=(10, 5))
119     plt.hist(pdf[:, 2], bins=200, color='r')
120     plt.xlabel('PDF*chi-squared/2')
121     plt.ylabel('No. of appearances')
122     plt.title("Name: Tudor Trita Trita , CID:01199397 \n Figure 6:
Histogram of PDF*chi-squared/2 for f_14")
123     fig6.savefig("fig6.png")
124     plt.show()
125
126     return None
127
128
129 # Question 2 functions below:
130
131
132 def dirspecest(Nr, X, f):
133     """Return direct spectral estimator for Q2."""
134     N = len(X[0, :])
135     flen = len(f)
136     nums = np.linspace(1, N, N)
137     Dspec = np.zeros((Nr, flen))
138     #htaper:
139     ht = 0.5*np.sqrt(8/(3*(N+1)))*(1-np.cos(2*np.pi*nums/(N+1)))
140
141     for i in range(flen):
142         c = ht*X[:, :]*np.exp(-1j*2*np.pi*f[i]*nums)
143         Dspec[:, i] = abs(np.sum(c, axis=1))**2
144     return Dspec
145
146
147 def ywnotaper(Nr, X, f):
148     """Compute YW estimate without using tapering."""
149     N = len(X[0, :])
150     svec = np.zeros(3)
151     flen = len(f)
152     YWno = np.zeros((Nr, flen))
153
154     for k in range(Nr):
155         # S estimates finished
156         for i in range(3):

```

```
157         for j in range(N-i):
158             svec[i] = svec[i] + (1/N)*X[k][j]*X[k][j+i]
159
160         gammasmall = np.array([svec[1], svec[2]])
161         gammacap = np.array([[svec[0], svec[1]], [svec[1], svec[0]]])
162         thi = np.dot(lng.inv(gammacap), gammasmall)
163
164         sigma = svec[0] - thi[0]*svec[1] - thi[1]*svec[2]
165
166         for i in range(flen):
167             YWno[k,i] = (sigma/(abs(1-(thi[0]*np.exp(-1j*2*np.pi*f[i]))+
168                                     thi[1]*np.exp(-1j*2*np.pi*f[i]*2))))**2))
169         svec = np.zeros(3)
170     return YWno
171
172
173 def ywwithtaper(Nr, X, f):
174     """Compute YW estimate using tapering."""
175     N = len(X[0,:])
176     svec = np.zeros(3)
177     flen = len(f)
178     nums = np.linspace(1, N, N)
179     YWwith = np.zeros((Nr, flen))
180     htaper = 0.5*np.sqrt(8/(3*(N+1)))*(1-np.cos(2*np.pi*nums/(N+1)))
181
182     for k in range(Nr):
183         for i in range(3):
184             for j in range(N-i):
185                 svec[i] = svec[i] + htaper[j]*X[k][j]*htaper[j+i]*X[k][j+
186 i]
187
188         gammasmall = np.array([svec[1], svec[2]])
189         gammacap = np.array([[svec[0], svec[1]], [svec[1], svec[0]]])
190         thi = np.dot(lng.inv(gammacap), gammasmall)
191
192         sigma = svec[0] - thi[0]*svec[1] - thi[1]*svec[2]
193
194         for i in range(flen):
195             YWwith[k,i] = (sigma/(abs(1-(thi[0]*np.exp(-1j*2*np.pi*f[i]))
196                                     +thi[1]*np.exp(-1j*2*np.pi*f[i]*2))))**2))
197         svec = np.zeros(3)
198     return YWwith
199
200 def q2parta():
201     """Content of part A of question 2."""
202     real1 = arprocess(1, 64)
203     real2 = arprocess(1, 256)
204     real3 = arprocess(1, 1024)
205
206     f1 = np.linspace(1,32,32)
207     f1 = f1/64
208
209     f2 = np.linspace(1,128,128)
210     f2 = f2/256
```

```

211
212     f3 = np.linspace(1,512,512)
213     f3 = f3/1024
214
215     spectra1 = sdf(f1)
216     spectra2 = sdf(f2)
217     spectra3 = sdf(f3)
218
219
220     Sper1 = periodogram(1, real1, f1)
221     Sper2 = periodogram(1, real2, f2)
222     Sper3 = periodogram(1, real3, f3)
223
224     Dspec1 = dirspecest(1, real1, f1)
225     Dspec2 = dirspecest(1, real2, f2)
226     Dspec3 = dirspecest(1, real3, f3)
227
228     YWnot1 = ywnotaper(1, real1, f1)
229     YWnot2 = ywnotaper(1, real2, f2)
230     YWnot3 = ywnotaper(1, real3, f3)
231
232     YWwith1 = ywwithtaper(1, real1, f1)
233     YWwith2 = ywwithtaper(1, real2, f2)
234     YWwith3 = ywwithtaper(1, real3, f3)
235
236
237
238     fig7, axes = plt.subplots(nrows=4, ncols=3, figsize=(15, 10))
239     # Row 1: the periodogram estimate of the sdf.
240     axes[0, 0].plot(f1, Sper1.transpose(), 'b',
241                    f1, spectra1, 'r')
242     axes[0, 1].plot(f2, Sper2.transpose(), 'b',
243                    f2, spectra2, 'r')
244     axes[0, 2].plot(f3, Sper3.transpose(), 'b',
245                    f3, spectra3, 'r')
246
247     axes[1, 0].plot(f1, Dspec1.transpose(), 'b',
248                    f1, spectra1, 'r')
249     axes[1, 1].plot(f2, Dspec2.transpose(), 'b',
250                    f2, spectra2, 'r')
251     axes[1, 2].plot(f3, Dspec3.transpose(), 'b',
252                    f3, spectra3, 'r')
253
254     axes[2, 0].plot(f1, YWnot1.transpose(), 'b',
255                    f1, spectra1, 'r')
256     axes[2, 1].plot(f2, YWnot2.transpose(), 'b',
257                    f2, spectra2, 'r')
258     axes[2, 2].plot(f3, YWnot3.transpose(), 'b',
259                    f3, spectra3, 'r')
260
261     axes[3, 0].plot(f1, YWwith1.transpose(), 'b',
262                    f1, spectra1, 'r')
263     axes[3, 1].plot(f2, YWwith2.transpose(), 'b',
264                    f2, spectra2, 'r')
265     axes[3, 2].plot(f3, YWwith3.transpose(), 'b',

```

```
266             f3, spectra3, 'r')
267
268     axes[0, 0].set_title('N = 64')
269     axes[0, 1].set_title('N = 256')
270     axes[0, 2].set_title('N = 1024')
271     fig7.suptitle("Name: Tudor Trita Trita, CID:01199397 \n Figure 7:
Plots for Question 2 Part A. Red line = sdf, Blue line = current
method.")
272     plt.savefig('fig7.png')
273     plt.show()
274
275     return None
276
277
278 def q2partb():
279     """Content of part A of question 2."""
280     Nr = 10000
281     real1 = arprocess(Nr, 64)
282     real2 = arprocess(Nr, 256)
283     real3 = arprocess(Nr, 1024)
284
285     f1 = np.linspace(1,32,32)
286     f1 = f1/64
287
288     f2 = np.linspace(1,128,128)
289     f2 = f2/256
290
291     f3 = np.linspace(1,512,512)
292     f3 = f3/1024
293
294     spectra1 = sdf(f1)
295     spectra2 = sdf(f2)
296     spectra3 = sdf(f3)
297
298     Sper1 = periodogram(Nr, real1, f1)
299     Sper2 = periodogram(Nr, real2, f2)
300     Sper3 = periodogram(Nr, real3, f3)
301     Sper1 = np.mean(Sper1, axis=0)
302     Sper2 = np.mean(Sper2, axis=0)
303     Sper3 = np.mean(Sper3, axis=0)
304
305     Dspec1 = dirspecest(Nr, real1, f1)
306     Dspec2 = dirspecest(Nr, real2, f2)
307     Dspec3 = dirspecest(Nr, real3, f3)
308     Dspec1 = np.mean(Dspec1, axis=0)
309     Dspec2 = np.mean(Dspec2, axis=0)
310     Dspec3 = np.mean(Dspec3, axis=0)
311
312     YWnot1 = ywnotaper(Nr, real1, f1)
313     YWnot2 = ywnotaper(Nr, real2, f2)
314     YWnot3 = ywnotaper(Nr, real3, f3)
315     YWnot1 = np.mean(YWnot1, axis=0)
316     YWnot2 = np.mean(YWnot2, axis=0)
317     YWnot3 = np.mean(YWnot3, axis=0)
318
```

```

319     YWwith1 = ywwithtaper(Nr, real1, f1)
320     YWwith2 = ywwithtaper(Nr, real2, f2)
321     YWwith3 = ywwithtaper(Nr, real3, f3)
322     YWwith1 = np.mean(YWwith1, axis=0)
323     YWwith2 = np.mean(YWwith2, axis=0)
324     YWwith3 = np.mean(YWwith3, axis=0)
325
326     # Figure 8: n=64
327     fig8 = plt.figure(figsize=(10, 7))
328     plt.plot(f1, spectral1, 'k', label='real sdf')
329     plt.plot(f1, Sper1, 'b', label='periodogram')
330     plt.plot(f1, Dspec1, 'g', label='direct est.')
331     plt.plot(f1, YWnot1, 'y', label='YW no taper')
332     plt.plot(f1, YWwith1, 'r', label='YW with taper')
333     plt.legend(loc='upper right')
334     plt.xlabel('f')
335     plt.title("Name: Tudor Trita Trita, CID:01199397 \n Figure 8: Plot N
=64 for Question 2 Part B")
336     fig8.savefig('fig8.png')
337     plt.show()
338
339     # Figure 9: n=256
340     fig9 = plt.figure(figsize=(10, 7))
341     plt.plot(f2, spectra2, 'k', label='real sdf')
342     plt.plot(f2, Sper2, 'b', label='periodogram')
343     plt.plot(f2, Dspec2, 'g', label='direct est.')
344     plt.plot(f2, YWnot2, 'y', label='YW no taper')
345     plt.plot(f2, YWwith2, 'r', label='YW with taper')
346     plt.legend(loc='upper right')
347     plt.xlabel('f')
348     plt.title("Name: Tudor Trita Trita, CID:01199397 \n Figure 9: Plot N
=256 for Question 2 Part B")
349     fig9.savefig('fig9.png')
350     plt.show()
351
352     # Figure 10: n=1024
353     fig10 = plt.figure(figsize=(10, 7))
354     plt.plot(f3, spectra3, 'k', label='real sdf')
355     plt.plot(f3, Sper3, 'b', label='periodogram')
356     plt.plot(f3, Dspec3, 'g', label='direct est.')
357     plt.plot(f3, YWnot3, 'y', label='YW no taper')
358     plt.plot(f3, YWwith3, 'r', label='YW with taper')
359     plt.legend(loc='upper right')
360     plt.xlabel('f')
361     plt.title("Name: Tudor Trita Trita, CID:01199397 \n Figure 10: Plot N
=1024 for Question 2 Part B")
362     fig10.savefig('fig10.png')
363     plt.show()
364     return None
365
366 def mamodel(Nr, t):
367     """Return Nr instances of MA(3) model."""
368     epsilon = rnd.normal(0, 1, (Nr, t))
369     X = np.zeros((Nr, t))
370

```

```
371     X[:, 0] = epsilon[:, 0]
372     X[:, 1] = epsilon[:, 1] + 0.5*epsilon[:, 0]
373     X[:, 2] = epsilon[:, 2] + 0.5*epsilon[:, 1] - 0.25*epsilon[:, 0]
374     X[:, 3:] = (epsilon[:, 3:] + 0.5*epsilon[:, 2:-1] - 0.25*epsilon[:,
1:-2]
375                 + 0.5*epsilon[:, 0:-3])
376     return X
377
378
379 def sdfma(f):
380     """Return sdf for flen number of frequencies of MA(3) process."""
381     flen = len(f)
382     sdf = np.zeros(flen)
383     for i in range(flen):
384         sdf[i] = (abs(1 + (1/2)*np.exp(-1j*2*np.pi*f[i]) -
385                     (1/4)*np.exp(-1j*2*np.pi*f[i]*2) +
386                     (1/2)*np.exp(-1j*2*np.pi*f[i]*3))**2)
387     return sdf
388
389
390 def q2partc():
391     """Content of part A of question 2."""
392     model1 = mamodel(1, 64)
393     model2 = mamodel(1, 256)
394     model3 = mamodel(1, 1024)
395     real1 = arprocess(1, 64)
396     real2 = arprocess(1, 256)
397     real3 = arprocess(1, 1024)
398
399     f1 = np.linspace(1, 32, 32)
400     f1 = f1/64
401
402     f2 = np.linspace(1, 128, 128)
403     f2 = f2/256
404
405     f3 = np.linspace(1, 512, 512)
406     f3 = f3/1024
407
408     spectra1 = sdfma(f1)
409     spectra2 = sdfma(f2)
410     spectra3 = sdfma(f3)
411
412     Sper1 = periodogram(1, model1, f1)
413     Sper2 = periodogram(1, model2, f2)
414     Sper3 = periodogram(1, model3, f3)
415
416     Dspec1 = dirspecest(1, model1, f1)
417     Dspec2 = dirspecest(1, model2, f2)
418     Dspec3 = dirspecest(1, model3, f3)
419
420     YWnot1 = ywnotaper(1, real1, f1)
421     YWnot2 = ywnotaper(1, real2, f2)
422     YWnot3 = ywnotaper(1, real3, f3)
423
424     YWwith1 = ywwihtaper(1, real1, f1)
```

```

425     YWwith2 = ywwithtaper(1, real2, f2)
426     YWwith3 = ywwithtaper(1, real3, f3)
427
428     fig11, axes = plt.subplots(nrows=4, ncols=3, figsize=(15, 10))
429     axes[0, 0].plot(f1, Sper1.transpose(), 'b',
430                    f1, spectra1, 'r')
431     axes[0, 1].plot(f2, Sper2.transpose(), 'b',
432                    f2, spectra2, 'r')
433     axes[0, 2].plot(f3, Sper3.transpose(), 'b',
434                    f3, spectra3, 'r')
435
436     axes[1, 0].plot(f1, Dspec1.transpose(), 'b',
437                    f1, spectra1, 'r')
438     axes[1, 1].plot(f2, Dspec2.transpose(), 'b',
439                    f2, spectra2, 'r')
440     axes[1, 2].plot(f3, Dspec3.transpose(), 'b',
441                    f3, spectra3, 'r')
442
443     axes[2, 0].plot(f1, YWnot1.transpose(), 'b',
444                    f1, spectra1, 'r')
445     axes[2, 1].plot(f2, YWnot2.transpose(), 'b',
446                    f2, spectra2, 'r')
447     axes[2, 2].plot(f3, YWnot3.transpose(), 'b',
448                    f3, spectra3, 'r')
449
450     axes[3, 0].plot(f1, YWwith1.transpose(), 'b',
451                    f1, spectra1, 'r')
452     axes[3, 1].plot(f2, YWwith2.transpose(), 'b',
453                    f2, spectra2, 'r')
454     axes[3, 2].plot(f3, YWwith3.transpose(), 'b',
455                    f3, spectra3, 'r')
456     axes[0, 0].set_title('N = 64')
457     axes[0, 1].set_title('N = 256')
458     axes[0, 2].set_title('N = 1024')
459     fig11.suptitle("Name: Tudor Trita Trita, CID:01199397 \n Figure 11:
Plots for Question 2 Part C. Red line = sdf, Blue line = current
method.")
460     plt.savefig('fig11.png')
461     plt.show()
462
463     return None
464
465
466 def q2partd():
467     """Plot of Q2 part d."""
468     Nr = 10000
469     model1 = mamodel(Nr, 64)
470     model2 = mamodel(Nr, 256)
471     model3 = mamodel(Nr, 1024)
472     real1 = arprocess(Nr, 64)
473     real2 = arprocess(Nr, 256)
474     real3 = arprocess(Nr, 1024)
475
476     f1 = np.linspace(1,32,32)
477     f1 = f1/64

```

```
478
479     f2 = np.linspace(1,128,128)
480     f2 = f2/256
481
482     f3 = np.linspace(1,512,512)
483     f3 = f3/1024
484
485     spectra1 = sdfma(f1)
486     spectra2 = sdfma(f2)
487     spectra3 = sdfma(f3)
488
489     Sper1 = periodogram(Nr, model1, f1)
490     Sper2 = periodogram(Nr, model2, f2)
491     Sper3 = periodogram(Nr, model3, f3)
492     Sper1 = np.mean(Sper1, axis=0)
493     Sper2 = np.mean(Sper2, axis=0)
494     Sper3 = np.mean(Sper3, axis=0)
495
496     Dspec1 = dirspecest(Nr, model1, f1)
497     Dspec2 = dirspecest(Nr, model2, f2)
498     Dspec3 = dirspecest(Nr, model3, f3)
499     Dspec1 = np.mean(Dspec1, axis=0)
500     Dspec2 = np.mean(Dspec2, axis=0)
501     Dspec3 = np.mean(Dspec3, axis=0)
502
503     YWnot1 = ywnotaper(Nr, real1, f1)
504     YWnot2 = ywnotaper(Nr, real2, f2)
505     YWnot3 = ywnotaper(Nr, real3, f3)
506     YWnot1 = np.mean(YWnot1, axis=0)
507     YWnot2 = np.mean(YWnot2, axis=0)
508     YWnot3 = np.mean(YWnot3, axis=0)
509
510     YWwith1 = ywwithtaper(Nr, real1, f1)
511     YWwith2 = ywwithtaper(Nr, real2, f2)
512     YWwith3 = ywwithtaper(Nr, real3, f3)
513     YWwith1 = np.mean(YWwith1, axis=0)
514     YWwith2 = np.mean(YWwith2, axis=0)
515     YWwith3 = np.mean(YWwith3, axis=0)
516
517     # Figure 12: n=64
518     plt.figure(figsize=(10, 7))
519     plt.plot(f1, spectra1, 'k', label='real sdf')
520     plt.plot(f1, Sper1, 'b', label='periodogram')
521     plt.plot(f1, Dspec1, 'g', label='direct est.')
522     plt.plot(f1, YWnot1, 'y', label='YW no taper')
523     plt.plot(f1, YWwith1, 'r', label='YW with taper')
524     plt.legend(loc='upper right')
525     plt.xlabel('f')
526     plt.title("Name: Tudor Trita Trita, CID:01199397 \n Figure 12: Plot N
=64 for Question 2 Part D")
527     plt.savefig('fig12.png')
528     plt.show()
529
530     # Figure 13: n=256
531     plt.figure(figsize=(10, 7))
```

```

532     plt.plot(f2, spectra2, 'k', label='real sdf')
533     plt.plot(f2, Sper2, 'b', label='periodogram')
534     plt.plot(f2, Dspec2, 'g', label='direct est.')
535     plt.plot(f2, YWnot2, 'y', label='YW no taper')
536     plt.plot(f2, YWwith2, 'r', label='YW with taper')
537     plt.legend(loc='upper right')
538     plt.xlabel('f')
539     plt.title("Name: Tudor Trita Trita, CID:01199397 \n Figure 13: Plot N
=64 for Question 2 Part D")
540     plt.savefig('fig13.png')
541     plt.show()
542
543     # Figure 14: n=1024
544     plt.figure(figsize=(10, 7))
545     plt.plot(f3, spectra3, 'k', label='real sdf')
546     plt.plot(f3, Sper3, 'b', label='periodogram')
547     plt.plot(f3, Dspec3, 'g', label='direct est.')
548     plt.plot(f3, YWnot3, 'y', label='YW no taper')
549     plt.plot(f3, YWwith3, 'r', label='YW with taper')
550     plt.legend(loc='upper right')
551     plt.xlabel('f')
552     plt.title("Name: Tudor Trita Trita, CID:01199397 \n Figure 14: Plot N
=64 for Question 2 Part D")
553     plt.savefig('fig14.png')
554     plt.show()
555     return None
556
557
558 if __name__ == '__main__':
559     #Question 1:
560
561     #Setting up parameters:
562     Nr, t = 10000, 128
563
564     #Setting up 10k instances of AR(2) process:
565     X = arprocess(Nr, t)
566     f = np.array([12/128, 13/128, 14/128])
567
568     #Fetching periodogram for 10k instances and all three frequencies
569     Sper = periodogram(Nr, X, f) #Returns 10kx3 array
570     sm, sv = qlparta(Sper, f) #Calculating average of periodogram
571
572     spectra = sdf(f)
573     spectra2 = spectra**2
574
575     print("Sample means of periodogram are ", sm, "respectively.")
576     print("S(f) for f = 12/128, 13/128, 14/128 is ", spectra)
577     print("Sample variances of periodogram are ", sv, "respectively.")
578     print("S(f)^2 for f = 12/128, 13/128, 14/128 is", spectra2)
579
580     #Fetching correlation coefficients
581     pearson = qlpartb(Sper)
582     print("Pearson coefficients are ", pearson[:, 0], "respectively.")
583
584     #Comment/Uncomment for part c histograms to show:

```

```
585     q1partc(spectra , Sper)
586
587     #Question 2
588
589     #Comment/Uncomment for part a plot to show:
590     q2parta()
591
592     #Comment/Uncomment for part b plots to show: (WARNING: TAKES 3-4mins)
593     q2partb()
594
595     #Comment/Uncomment for part c plot to show:
596     q2partc()
597
598     #Comment/Uncomment for part d plots to show: (WARNING: TAKES 3-4mins)
599     q2partd()
600     print("Everything complete.")
```