Imperial College London

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

```
"""Time Series CW2 code used

Tudor Trita Trita

Note: Functions are declared before main code executes at the bottom
in __init__=='__main__' part.

Functions are created as and when needed in CW2.
"""

import numpy as np
import matplotlib.pyplot as plt
import numpy.random as rnd
import numpy.linalg as lng
import scipy.stats


def arprocess(Nr, t):
    """Return N instances of AR(2) process."""
    epsilon = rnd.normal(0, 1, (Nr, t))
    X = np.zeros((Nr, t))
    X[:, 0] = (4/3)*epsilon[:, 0]
    X[:, 1] = 0.5*X[:, 0] + 2/(np.sqrt(3)) * epsilon[:, 1]
    for i in range(t-2):
        X[:, i+2] = 0.75*X[:, i+1] - 0.5*X[:, i] + epsilon[:, i]
    return X


def periodogram(Nr, X, f):
    """Return Nr instances of the periodogram."""
    N = len(X[0, :])
    flen = len(f)
    nums = np.linspace(1, N, N)
    Sper = np.zeros((Nr, flen))
    for i in range(flen):
        a = X[:, :]*np.exp(-1j*2*np.pi*f[i]*nums)
        Sper[:, i] = 1/N*abs(np.sum(a, axis=1))**2
    return Sper


def sdf(f):
    """Return sdf for flen number of frequencies of AR(2) process."""
    flen = len(f)
    Spectra = np.zeros(flen)
    for i in range(flen):
        Spectra[i] = (1/abs(1-(3/4)*np.exp(-1j*2*np.pi*f[i])
                      + (1/2)*np.exp(-1j*2*np.pi*f[i]*2))**2)
    return Spectra


def q1parta(Sper, f):
    """Return sample mean and variance."""
```

```
    Sper.transpose()
    sm = np.mean(Sper, axis=0)
    sv = np.var(Sper, axis=0)
    return sm, sv


def q1partb(Sper):
    """Return sample correlation."""
    pearson = np.zeros((len(Sper[0, :]), 2))
    pearson[0, :] = scipy.stats.pearsonr(Sper[:, 0], Sper[:, 1])
    pearson[1, :] = scipy.stats.pearsonr(Sper[:, 0], Sper[:, 2])
    pearson[2, :] = scipy.stats.pearsonr(Sper[:, 1], Sper[:, 2])
    pearson.transpose()
    return pearson


def q1partc(spectra, Sper):
    """Plot histograms."""
    Nr = len(Sper[:,0])
    epsilon = rnd.chisquare(2, (Nr, 3))
    pdf = np.zeros((Nr,3))
    pdf[:,0] = spectra[0]*epsilon[:,0]/2
    pdf[:,1] = spectra[1]*epsilon[:,1]/2
    pdf[:,2] = spectra[2]*epsilon[:,2]/2

    fig1 = plt.figure(figsize=(10, 5))
    plt.hist(Sper[:, 0], bins=200, color='#0504aa')
    plt.xlabel('Periodogram')
    plt.ylabel('No. of appearances')
    plt.title("Name: Tudor Trita Trita, CID:01199397 \n Figure 1:
    Histogram of Periodogram for f_12")
    fig1.savefig("fig1.png")
    plt.show()

    fig2 = plt.figure(figsize=(10, 5))
    plt.hist(pdf[:, 0], bins=200, color='r')
    plt.xlabel('PDF*chi-squared/2')
    plt.ylabel('No. of appearances')
    plt.title("Name: Tudor Trita Trita, CID:01199397 \n Figure 2:
    Histogram of PDF*chi-squared/2 for f_12")
    fig2.savefig("fig2.png")
    plt.show()

    fig3 = plt.figure(figsize=(10, 5))
    plt.hist(Sper[:, 1], bins=200, color='#0504aa')
    plt.xlabel('Periodogram')
    plt.ylabel('No. of appearances')
    plt.title("Name: Tudor Trita Trita, CID:01199397 \n Figure 3:
    Histogram of Periodogram for f_13")
    fig3.savefig("fig3.png")
    plt.show()

    fig4 = plt.figure(figsize=(10, 5))
    plt.hist(pdf[:, 1], bins=200, color='r')
    plt.xlabel('PDF*chi-squared/2')
```

```
    plt.ylabel('No. of appearances')
    plt.title("Name: Tudor Trita Trita, CID:01199397 \n Figure 4: Histogram of PDF*chi-squared/2 for f_13")
    fig4.savefig("fig4.png")
    plt.show()

    fig5 = plt.figure(figsize=(10, 5))
    plt.hist(Sper[:, 2], bins=200, color='#0504aa')
    plt.xlabel('Periodogram')
    plt.ylabel('No. of appearances')
    plt.title("Name: Tudor Trita Trita, CID:01199397 \n Figure 5: Histogram of Periodogram for f_14")
    fig5.savefig("fig5.png")
    plt.show()

    fig6 = plt.figure(figsize=(10, 5))
    plt.hist(pdf[:, 2], bins=200, color='r')
    plt.xlabel('PDF*chi-squared/2')
    plt.ylabel('No. of appearances')
    plt.title("Name: Tudor Trita Trita, CID:01199397 \n Figure 6: Histogram of PDF*chi-squared/2 for f_14")
    fig6.savefig("fig6.png")
    plt.show()

    return None


# Question 2 functions below:


def dirspecest(Nr, X, f):
    """Return direct spectral estimator for Q2."""
    N = len(X[0, :])
    flen = len(f)
    nums = np.linspace(1, N, N)
    Dspec = np.zeros((Nr, flen))
    #htaper:
    ht = 0.5*np.sqrt(8/(3*(N+1)))*(1-np.cos(2*np.pi*nums/(N+1)))

    for i in range(flen):
        c = ht*X[:, :]*np.exp(-1j*2*np.pi*f[i]*nums)
        Dspec[:, i] = abs(np.sum(c, axis=1))**2
    return Dspec


def ywnotaper(Nr, X, f):
    """Compute YW estimate without using tapering."""
    N = len(X[0,:])
    svec = np.zeros(3)
    flen = len(f)
    YWno = np.zeros((Nr, flen))

    for k in range(Nr):
    # S estimates finished
        for i in range(3):
```

```
            for j in range(N-i):
                svec[i] = svec[i] + (1/N)*X[k][j]*X[k][j+i]

        gammasmall = np.array([svec[1], svec[2]])
        gammacap = np.array([[svec[0], svec[1]], [svec[1], svec[0]]])
        thi = np.dot(lng.inv(gammacap),gammasmall)

        sigma = svec[0] - thi[0]*svec[1] - thi[1]*svec[2]

        for i in range(flen):
            YWno[k,i] = (sigma/(abs(1-(thi[0]*np.exp(-1j*2*np.pi*f[i])+
                         thi[1]*np.exp(-1j*2*np.pi*f[i]*2)))**2))
        svec = np.zeros(3)
    return YWno


def ywwithtaper(Nr, X, f):
    """Compute YW estimate using tapering."""
    N = len(X[0,:])
    svec = np.zeros(3)
    flen = len(f)
    nums = np.linspace(1, N, N)
    YWwith = np.zeros((Nr, flen))
    htaper = 0.5*np.sqrt(8/(3*(N+1)))*(1-np.cos(2*np.pi*nums/(N+1)))

    for k in range(Nr):
        for i in range(3):
            for j in range(N-i):
                svec[i] = svec[i] + htaper[j]*X[k][j]*htaper[j+i]*X[k][j+i]

        gammasmall = np.array([svec[1], svec[2]])
        gammacap = np.array([[svec[0], svec[1]], [svec[1], svec[0]]])
        thi = np.dot(lng.inv(gammacap),gammasmall)

        sigma = svec[0] - thi[0]*svec[1] - thi[1]*svec[2]

        for i in range(flen):
            YWwith[k,i] = (sigma/(abs(1-(thi[0]*np.exp(-1j*2*np.pi*f[i])
                           +thi[1]*np.exp(-1j*2*np.pi*f[i]*2)))**2))
        svec = np.zeros(3)
    return YWwith


def q2parta():
    """Content of part A of question 2."""
    real1 = arprocess(1, 64)
    real2 = arprocess(1, 256)
    real3 = arprocess(1, 1024)

    f1 = np.linspace(1,32,32)
    f1 = f1/64

    f2 = np.linspace(1,128,128)
    f2 = f2/256
```

```

    f3 = np.linspace(1,512,512)
    f3 = f3/1024

    spectra1 = sdf(f1)
    spectra2 = sdf(f2)
    spectra3 = sdf(f3)


    Sper1 = periodogram(1, real1, f1)
    Sper2 = periodogram(1, real2, f2)
    Sper3 = periodogram(1, real3, f3)

    Dspec1 = dirspecest(1, real1, f1)
    Dspec2 = dirspecest(1, real2, f2)
    Dspec3 = dirspecest(1, real3, f3)

    YWnot1 = ywnotaper(1, real1, f1)
    YWnot2 = ywnotaper(1, real2, f2)
    YWnot3 = ywnotaper(1, real3, f3)

    YWwith1 = ywwithtaper(1, real1, f1)
    YWwith2 = ywwithtaper(1, real2, f2)
    YWwith3 = ywwithtaper(1, real3, f3)



    fig7, axes = plt.subplots(nrows=4, ncols=3, figsize=(15, 10))
    # Row 1: the periodogram estimate of the sdf.
    axes[0, 0].plot(f1, Sper1.transpose(), 'b',
                    f1, spectra1, 'r')
    axes[0, 1].plot(f2, Sper2.transpose(), 'b',
                    f2, spectra2, 'r')
    axes[0, 2].plot(f3, Sper3.transpose(), 'b',
                    f3, spectra3, 'r')

    axes[1, 0].plot(f1, Dspec1.transpose(), 'b',
                    f1, spectra1, 'r')
    axes[1, 1].plot(f2, Dspec2.transpose(), 'b',
                    f2, spectra2, 'r')
    axes[1, 2].plot(f3, Dspec3.transpose(), 'b',
                    f3, spectra3, 'r')

    axes[2, 0].plot(f1, YWnot1.transpose(), 'b',
                    f1, spectra1, 'r')
    axes[2, 1].plot(f2, YWnot2.transpose(), 'b',
                    f2, spectra2, 'r')
    axes[2, 2].plot(f3, YWnot3.transpose(), 'b',
                    f3, spectra3, 'r')

    axes[3, 0].plot(f1, YWwith1.transpose(), 'b',
                    f1, spectra1, 'r')
    axes[3, 1].plot(f2, YWwith2.transpose(), 'b',
                    f2, spectra2, 'r')
    axes[3, 2].plot(f3, YWwith3.transpose(), 'b',
```

```
                        f3, spectra3, 'r')

    axes[0, 0].set_title('N = 64')
    axes[0, 1].set_title('N = 256')
    axes[0, 2].set_title('N = 1024')
    fig7.suptitle("Name: Tudor Trita Trita, CID:01199397 \n Figure 7:
    Plots for Question 2 Part A. Red line = sdf, Blue line = current
    method.")
    plt.savefig('fig7.png')
    plt.show()

    return None


def q2partb():
    """Content of part A of question 2."""
    Nr = 10000
    real1 = arprocess(Nr, 64)
    real2 = arprocess(Nr, 256)
    real3 = arprocess(Nr, 1024)

    f1 = np.linspace(1,32,32)
    f1 = f1/64

    f2 = np.linspace(1,128,128)
    f2 = f2/256

    f3 = np.linspace(1,512,512)
    f3 = f3/1024

    spectra1 = sdf(f1)
    spectra2 = sdf(f2)
    spectra3 = sdf(f3)

    Sper1 = periodogram(Nr, real1, f1)
    Sper2 = periodogram(Nr, real2, f2)
    Sper3 = periodogram(Nr, real3, f3)
    Sper1 = np.mean(Sper1, axis=0)
    Sper2 = np.mean(Sper2, axis=0)
    Sper3 = np.mean(Sper3, axis=0)

    Dspec1 = dirspecest(Nr, real1, f1)
    Dspec2 = dirspecest(Nr, real2, f2)
    Dspec3 = dirspecest(Nr, real3, f3)
    Dspec1 = np.mean(Dspec1, axis=0)
    Dspec2 = np.mean(Dspec2, axis=0)
    Dspec3 = np.mean(Dspec3, axis=0)

    YWnot1 = ywnotaper(Nr, real1, f1)
    YWnot2 = ywnotaper(Nr, real2, f2)
    YWnot3 = ywnotaper(Nr, real3, f3)
    YWnot1 = np.mean(YWnot1, axis=0)
    YWnot2 = np.mean(YWnot2, axis=0)
    YWnot3 = np.mean(YWnot3, axis=0)

```

```
    YWwith1 = ywwithtaper(Nr, real1, f1)
    YWwith2 = ywwithtaper(Nr, real2, f2)
    YWwith3 = ywwithtaper(Nr, real3, f3)
    YWwith1 = np.mean(YWwith1, axis=0)
    YWwith2 = np.mean(YWwith2, axis=0)
    YWwith3 = np.mean(YWwith3, axis=0)

    # Figure 8: n=64
    fig8 = plt.figure(figsize=(10, 7))
    plt.plot(f1, spectra1, 'k', label='real sdf')
    plt.plot(f1, Sper1, 'b', label='periodogram')
    plt.plot(f1, Dspec1, 'g', label='direct est.')
    plt.plot(f1, YWnot1, 'y', label='YW no taper')
    plt.plot(f1, YWwith1, 'r', label='YW with taper')
    plt.legend(loc='upper right')
    plt.xlabel('f')
    plt.title("Name: Tudor Trita Trita, CID:01199397 \n Figure 8: Plot N
   =64 for Question 2 Part B")
    fig8.savefig('fig8.png')
    plt.show()

    # Figure 9: n=256
    fig9 = plt.figure(figsize=(10, 7))
    plt.plot(f2, spectra2, 'k', label='real sdf')
    plt.plot(f2, Sper2, 'b', label='periodogram')
    plt.plot(f2, Dspec2, 'g', label='direct est.')
    plt.plot(f2, YWnot2, 'y', label='YW no taper')
    plt.plot(f2, YWwith2, 'r', label='YW with taper')
    plt.legend(loc='upper right')
    plt.xlabel('f')
    plt.title("Name: Tudor Trita Trita, CID:01199397 \n Figure 9: Plot N
   =256 for Question 2 Part B")
    fig9.savefig('fig9.png')
    plt.show()

    # Figure 10: n=1024
    fig10 = plt.figure(figsize=(10, 7))
    plt.plot(f3, spectra3, 'k', label='real sdf')
    plt.plot(f3, Sper3, 'b', label='periodogram')
    plt.plot(f3, Dspec3, 'g', label='direct est.')
    plt.plot(f3, YWnot3, 'y', label='YW no taper')
    plt.plot(f3, YWwith3, 'r', label='YW with taper')
    plt.legend(loc='upper right')
    plt.xlabel('f')
    plt.title("Name: Tudor Trita Trita, CID:01199397 \n Figure 10: Plot N
   =1024 for Question 2 Part B")
    fig10.savefig('fig10.png')
    plt.show()
    return None


def mamodel(Nr, t):
    """Return Nr instances of MA(3) model."""
    epsilon = rnd.normal(0, 1, (Nr, t))
    X = np.zeros((Nr, t))
```

```
    X[:, 0] = epsilon[:, 0]
    X[:, 1] = epsilon[:, 1] + 0.5*epsilon[:, 0]
    X[:, 2] = epsilon[:, 2] + 0.5*epsilon[:, 1] - 0.25*epsilon[:, 0]
    X[:, 3:] = (epsilon[:, 3:] + 0.5*epsilon[:, 2:-1] - 0.25*epsilon[:,
    1:-2]
                + 0.5*epsilon[:, 0:-3])
    return X


def sdfma(f):
    """Return sdf for flen number of frequencies of MA(3) process."""
    flen = len(f)
    sdf = np.zeros(flen)
    for i in range(flen):
        sdf[i] = (abs(1 + (1/2)*np.exp(-1j*2*np.pi*f[i]) -
                      (1/4)*np.exp(-1j*2*np.pi*f[i]*2) +
                      (1/2)*np.exp(-1j*2*np.pi*f[i]*3))**2)
    return sdf


def q2partc():
    """Content of part A of question 2."""
    model1 = mamodel(1, 64)
    model2 = mamodel(1, 256)
    model3 = mamodel(1, 1024)
    real1 = arprocess(1, 64)
    real2 = arprocess(1, 256)
    real3 = arprocess(1, 1024)

    f1 = np.linspace(1, 32, 32)
    f1 = f1/64

    f2 = np.linspace(1, 128, 128)
    f2 = f2/256

    f3 = np.linspace(1, 512, 512)
    f3 = f3/1024

    spectra1 = sdfma(f1)
    spectra2 = sdfma(f2)
    spectra3 = sdfma(f3)

    Sper1 = periodogram(1, model1, f1)
    Sper2 = periodogram(1, model2, f2)
    Sper3 = periodogram(1, model3, f3)

    Dspec1 = dirspecest(1, model1, f1)
    Dspec2 = dirspecest(1, model2, f2)
    Dspec3 = dirspecest(1, model3, f3)

    YWnot1 = ywnotaper(1, real1, f1)
    YWnot2 = ywnotaper(1, real2, f2)
    YWnot3 = ywnotaper(1, real3, f3)

    YWwith1 = ywwithtaper(1, real1, f1)
```

```
    YWwith2 = ywwithtaper(1, real2, f2)
    YWwith3 = ywwithtaper(1, real3, f3)

    fig11, axes = plt.subplots(nrows=4, ncols=3, figsize=(15, 10))
    axes[0, 0].plot(f1, Sper1.transpose(), 'b',
                    f1, spectra1, 'r')
    axes[0, 1].plot(f2, Sper2.transpose(), 'b',
                    f2, spectra2, 'r')
    axes[0, 2].plot(f3, Sper3.transpose(), 'b',
                    f3, spectra3, 'r')

    axes[1, 0].plot(f1, Dspec1.transpose(), 'b',
                    f1, spectra1, 'r')
    axes[1, 1].plot(f2, Dspec2.transpose(), 'b',
                    f2, spectra2, 'r')
    axes[1, 2].plot(f3, Dspec3.transpose(), 'b',
                    f3, spectra3, 'r')

    axes[2, 0].plot(f1, YWnot1.transpose(), 'b',
                    f1, spectra1, 'r')
    axes[2, 1].plot(f2, YWnot2.transpose(), 'b',
                    f2, spectra2, 'r')
    axes[2, 2].plot(f3, YWnot3.transpose(), 'b',
                    f3, spectra3, 'r')

    axes[3, 0].plot(f1, YWwith1.transpose(), 'b',
                    f1, spectra1, 'r')
    axes[3, 1].plot(f2, YWwith2.transpose(), 'b',
                    f2, spectra2, 'r')
    axes[3, 2].plot(f3, YWwith3.transpose(), 'b',
                    f3, spectra3, 'r')
    axes[0, 0].set_title('N = 64')
    axes[0, 1].set_title('N = 256')
    axes[0, 2].set_title('N = 1024')
    fig11.suptitle("Name: Tudor Trita Trita, CID:01199397 \n Figure 11:
   Plots for Question 2 Part C. Red line = sdf, Blue line = current
   method.")
    plt.savefig('fig11.png')
    plt.show()

    return None


def q2partd():
    """Plot of Q2 part d."""
    Nr = 10000
    model1 = mamodel(Nr, 64)
    model2 = mamodel(Nr, 256)
    model3 = mamodel(Nr, 1024)
    real1 = arprocess(Nr, 64)
    real2 = arprocess(Nr, 256)
    real3 = arprocess(Nr, 1024)

    f1 = np.linspace(1,32,32)
    f1 = f1/64
```

```
    f2 = np.linspace(1,128,128)
    f2 = f2/256

    f3 = np.linspace(1,512,512)
    f3 = f3/1024

    spectra1 = sdfma(f1)
    spectra2 = sdfma(f2)
    spectra3 = sdfma(f3)

    Sper1 = periodogram(Nr, model1, f1)
    Sper2 = periodogram(Nr, model2, f2)
    Sper3 = periodogram(Nr, model3, f3)
    Sper1 = np.mean(Sper1, axis=0)
    Sper2 = np.mean(Sper2, axis=0)
    Sper3 = np.mean(Sper3, axis=0)

    Dspec1 = dirspecest(Nr, model1, f1)
    Dspec2 = dirspecest(Nr, model2, f2)
    Dspec3 = dirspecest(Nr, model3, f3)
    Dspec1 = np.mean(Dspec1, axis=0)
    Dspec2 = np.mean(Dspec2, axis=0)
    Dspec3 = np.mean(Dspec3, axis=0)

    YWnot1 = ywnotaper(Nr, real1, f1)
    YWnot2 = ywnotaper(Nr, real2, f2)
    YWnot3 = ywnotaper(Nr, real3, f3)
    YWnot1 = np.mean(YWnot1, axis=0)
    YWnot2 = np.mean(YWnot2, axis=0)
    YWnot3 = np.mean(YWnot3, axis=0)

    YWwith1 = ywwithtaper(Nr, real1, f1)
    YWwith2 = ywwithtaper(Nr, real2, f2)
    YWwith3 = ywwithtaper(Nr, real3, f3)
    YWwith1 = np.mean(YWwith1, axis=0)
    YWwith2 = np.mean(YWwith2, axis=0)
    YWwith3 = np.mean(YWwith3, axis=0)

    # Figure 12: n=64
    plt.figure(figsize=(10, 7))
    plt.plot(f1, spectra1, 'k', label='real sdf')
    plt.plot(f1, Sper1, 'b', label='periodogram')
    plt.plot(f1, Dspec1, 'g', label='direct est.')
    plt.plot(f1, YWnot1, 'y', label='YW no taper')
    plt.plot(f1, YWwith1, 'r', label='YW with taper')
    plt.legend(loc='upper right')
    plt.xlabel('f')
    plt.title("Name: Tudor Trita Trita, CID:01199397 \n Figure 12: Plot N
   =64 for Question 2 Part D")
    plt.savefig('fig12.png')
    plt.show()

    # Figure 13: n=256
    plt.figure(figsize=(10, 7))
```

```
    plt.plot(f2, spectra2, 'k', label='real sdf')
    plt.plot(f2, Sper2, 'b', label='periodogram')
    plt.plot(f2, Dspec2, 'g', label='direct est.')
    plt.plot(f2, YWnot2, 'y', label='YW no taper')
    plt.plot(f2, YWwith2, 'r', label='YW with taper')
    plt.legend(loc='upper right')
    plt.xlabel('f')
    plt.title("Name: Tudor Trita Trita, CID:01199397 \n Figure 13: Plot N
   =64 for Question 2 Part D")
    plt.savefig('fig13.png')
    plt.show()

    # Figure 14: n=1024
    plt.figure(figsize=(10, 7))
    plt.plot(f3, spectra3, 'k', label='real sdf')
    plt.plot(f3, Sper3, 'b', label='periodogram')
    plt.plot(f3, Dspec3, 'g', label='direct est.')
    plt.plot(f3, YWnot3, 'y', label='YW no taper')
    plt.plot(f3, YWwith3, 'r', label='YW with taper')
    plt.legend(loc='upper right')
    plt.xlabel('f')
    plt.title("Name: Tudor Trita Trita, CID:01199397 \n Figure 14: Plot N
   =64 for Question 2 Part D")
    plt.savefig('fig14.png')
    plt.show()
    return None


if __name__ == '__main__':
    #Question 1:

    #Setting up parameters:
    Nr, t = 10000, 128

    #Setting up 10k instances of AR(2) process:
    X = arprocess(Nr, t)
    f = np.array([12/128, 13/128, 14/128])

    #Fetching periodogram for 10k instances and all three frequencies
    Sper = periodogram(Nr, X, f) #Returns 10kx3 array
    sm, sv = q1parta(Sper, f) #Calculating average of periodogram

    spectra = sdf(f)
    spectra2 = spectra**2

    print("Sample means of periodogram are ", sm, "respectively.")
    print("S(f) for f = 12/128, 13/128, 14/128 is ", spectra)
    print("Sample variances of periodogram are ", sv, "respectively.")
    print("S(f)^2 for f = 12/128, 13/128, 14/128 is", spectra2)

    #Fetching correlation coefficients
    pearson = q1partb(Sper)
    print("Pearson coefficients are ", pearson[:, 0], "respectively.")

    #Comment/Uncomment for part c histograms to show:
```

```
    q1partc(spectra, Sper)

    #Question 2

    #Comment/Uncomment for part a plot to show:
    q2parta()

    #Comment/Uncomment for part b plots to show: (WARNING: TAKES 3-4mins)
    q2partb()

    #Comment/Uncomment for part c plot to show:
    q2partc()

    #Comment/Uncomment for part d plots to show: (WARNING: TAKES 3-4mins)
    q2partd()
    print("Everything complete.")
```