Scientific Computation Project 3

02052671

December 15, 2023

Part 1

First, check if we can identify any trends by simply plotting the data. We will plot 1 factor on the x axis, while taking the average of the other 2 factors.

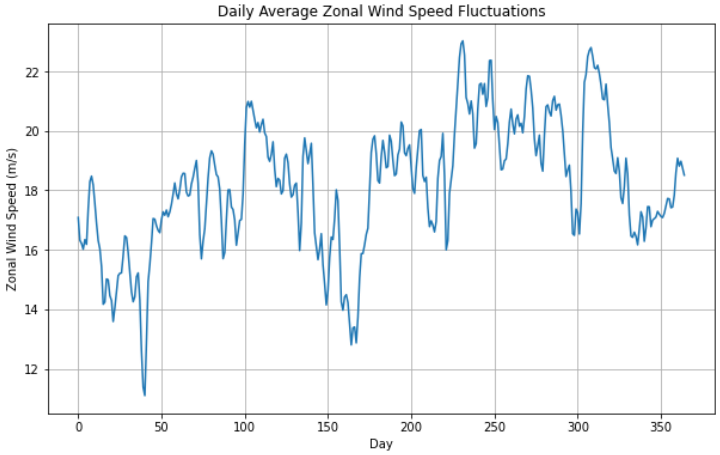


Figure 1: Figure 1.1

From the first plot, it is hard to identify any nontrivial trends in zonal wind speeds.

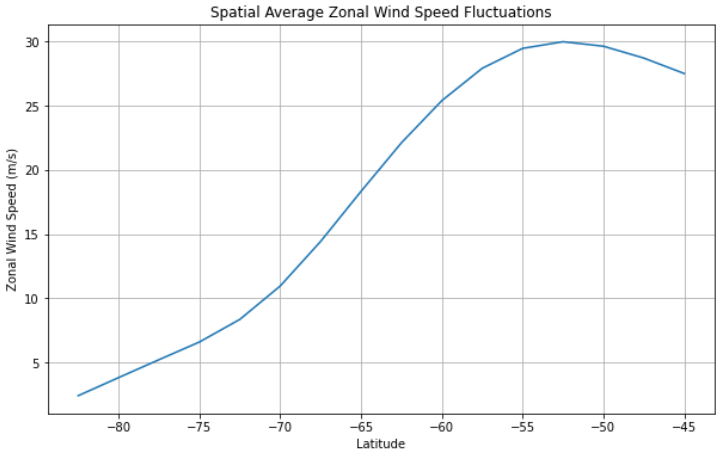


Figure 2: Figure 1.2

From the second plot, we notice that the zonal wind speed becomes stronger as the latitude increases.

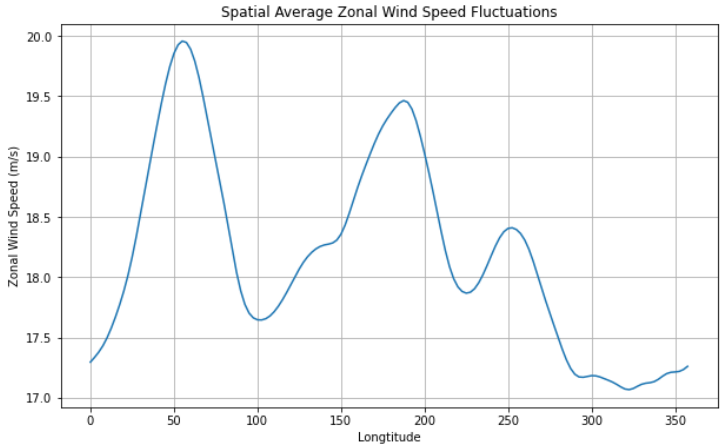


Figure 3: Figure 1.3

For the third plot, we can identify some peaks and troughs in certain longitudes.
Next we will analyse the frequencies of the given data set. In order to identify the frequencies within the data set. We need to do spectral analysis using Discrete Fourier Transform. Before we apply DFT to the data set, we need to manipulate the data so that it is suitable for DFT. First, we will make u a 2d array by flattening axis = 1 and 2. Therefore, the reshaped *u_reshaped* will be a $2304(= 16 \times 144) \times 365$. Then, we will take the mean of all the values inside the rows. *u_mean* is an array of 365 values. Finally, we need to standardize data for to make it suitable for DFT. We now apply DFT to our manipulated data set.

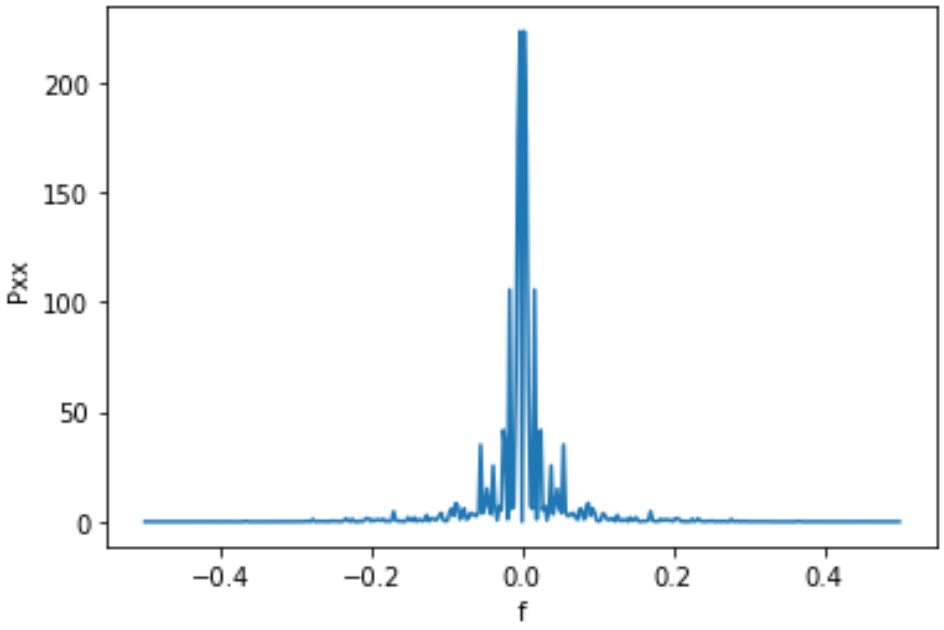


Figure 4: Figure 1.4

We notice the that the powers of the frequencies are centered around the middle of the plot. If we look deeper, the frequency corresponding to the maximum value of Pxx is $0.002739\cdots$ which is exactly 365 days. This shows that wind speed has a yearly periodic property.
Finally, we will apply PCA to analyze the wind data associated with large variations in u with respect to time. To apply PCA to the data set, manipulate u the same way we did before but this time we take the transpose. Therefore, the *u_reshaped* will be a 365 X $2304(365 \times 2304(= 16 \times 144))$ 2d array.Since PCA assumes zero mean data sets, subtract the mean from *u_reshaped* We now apply PCA using SVD and find the PCA applied matrix *u_tilde*. We will then check the the total variance and the covariance

betweens rows in *u_tilde*.
New total variance= 80529.84754909402
old total variance 80529.84754909402
Maximum covariance= 3.61199812068202e-11
We can see that the total variance is equal to the original data set and the covariance between rows is very small so we know that we applied PCA properly. The variance of the first principal component is 44586.305177465234. This means that the first principal component accounts for approximately 55.37%($= \frac{44586.305177465234}{80529.84754909402} \times 100$) of the variance of the entire data set.
The second principal component accounts for approximately 3.96%($= \frac{3188.0622977310377}{80529.84754909402}$)
Therefore, we will only look at the first few principal components to identify if there any significant trends in the data.
The first row of matrix T(or U^T) of the SVD of the wind data correspond to the first principal component of the PCA applied wind speed data(linear combination of day $= 1 \sim$ day $= 365$). If we take the standard deviation of the coefficients of the linear combination of the first principal component, we notice that the standard deviation(≈ 0.0096964440) of the coefficients are very low. This mean that every single day in a year contributes to the variation of the data evenly without a certain time frame significantly impacting the trend of the wind speed. If reconstruct u with the first principal component, we can easily see that most of the trends that we can see in the entire u is preserved.

Now we plot the first principal component.

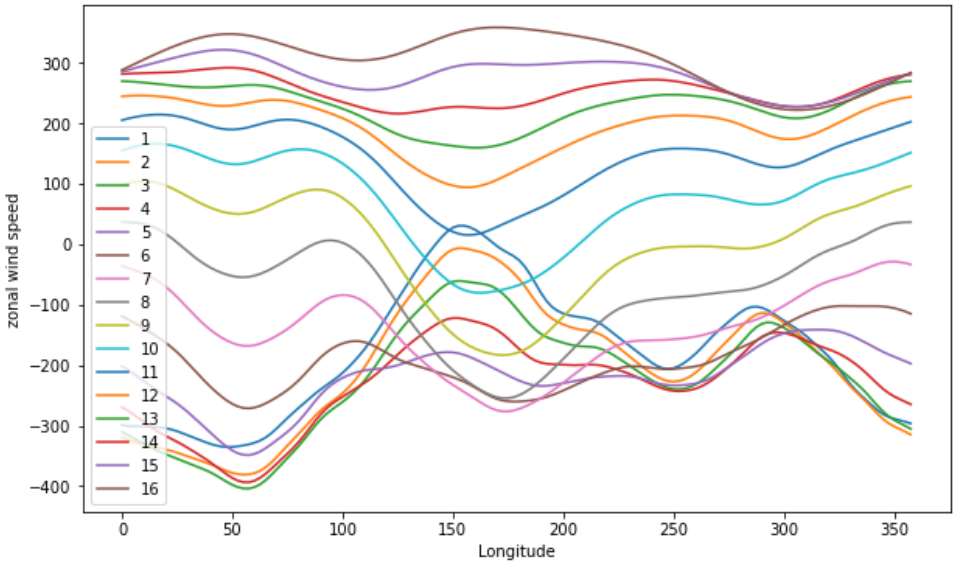


Figure 5: Figure 1.6: First Principal Component, points on the same line of same latitude

Each line in the plot represents the zonal wind speed for a constant latitude. As we can see, we some lines with similar shapes. Let's divide into three different groups.

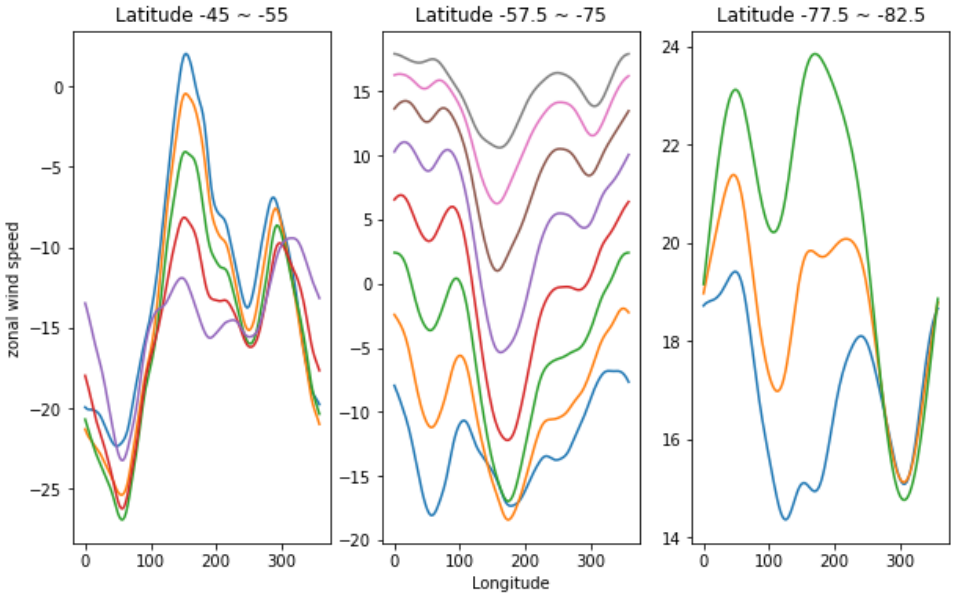


Figure 6: Figure 1.7: First Principal Component

We can see that the 3 groups have similar shapes of zonal wind shape functions. We can similar trends in other principal components.

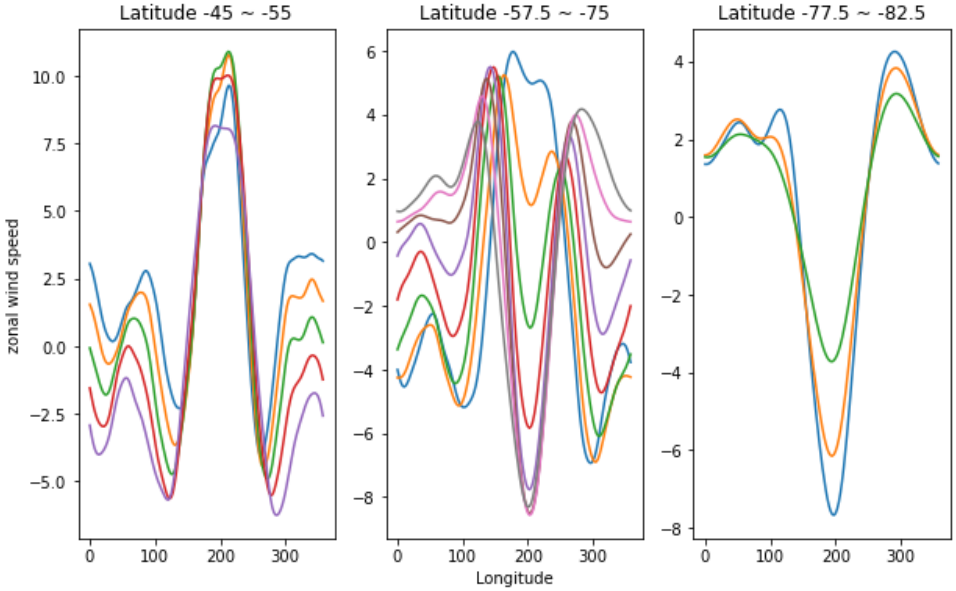


Figure 7: Figure 1.8: Second Principal Component

Therefore, we can take the space that the data set is considering, divide it into three different areas with lines orthogonal to the latitude, and conclude that the each area show a unique trend in wind speed throughout the entire year.

Part 2

2.

From the system of equations that represents method 2, take the LHS.

$$\begin{pmatrix} \cdots & & \cdots \\ & \tilde{f}_{1/2,j} & \\ & \vdots & \\ \cdots & \alpha\tilde{f}_{i-1,j} + \tilde{f}_{i,j} + \alpha\tilde{f}_{i+1,j} & \cdots \\ \cdots & \alpha\tilde{f}_{i,j} + \tilde{f}_{i+1,j} + \alpha\tilde{f}_{i+2,j} & \cdots \\ & \vdots & \\ \cdots & \tilde{f}_{m-3/2,j} & \cdots \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 & \cdots & \cdots & 0 \\ \alpha & 1 & \alpha & & & & & \\ 0 & \alpha & 1 & \alpha & & & & \\ \vdots & & \alpha & 1 & \ddots & & & \\ \vdots & & & \ddots & \ddots & & & \\ \vdots & & & & & 1 & \alpha & \\ & & & & & \alpha & 1 & \alpha \\ 0 & & & & & & 0 & 1 \end{pmatrix} \tilde{f}$$

This is equivalent to the LHS of the system of equations of method 2 in matrix form. Therefore, the interpolation using method 2 in matrix form is the following.

$$\begin{pmatrix} 1 & 0 & 0 & \cdots & 0 & \cdots & \cdots & 0 \\ \alpha & 1 & \alpha & & & & & \\ 0 & \alpha & 1 & \alpha & & & & \\ \vdots & & \alpha & 1 & \ddots & & & \\ \vdots & & & \ddots & \ddots & & & \\ \vdots & & & & & 1 & \alpha & \\ & & & & & \alpha & 1 & \alpha \\ 0 & & & & & & 0 & 1 \end{pmatrix} \tilde{f} = \begin{pmatrix} \tilde{a} & \tilde{b} & \tilde{c} & \tilde{d} & 0 & & \cdots & & 0 \\ b/2 & a/2 & a/2 & b/2 & & & & & \\ 0 & \ddots & \ddots & \ddots & \ddots & & & & \\ & & & & & & & & \\ \vdots & & & \ddots & \ddots & \ddots & \ddots & & \\ & & & & & & & & \\ & & & & & b/2 & a/2 & a/2 & b/2 \\ 0 & & & & & \tilde{a} & \tilde{b} & \tilde{c} & \tilde{d} \end{pmatrix} \text{f}$$

Represent the matrix equation above using banded matrix and solve using *sp.linalg.solve_banded*.
Now we will compare method 1 and method 2.
Method 1, The explicit method is a simple method to interpolate data. On the other hand, Method 2, the implicit method is a higher order interpolation and while it is more complex and requires more cost, it will be much more accurate overall. We will test this using some numerical tests.
Method 1 Absolute error: 0.020141137304302004
Method 2 Absolute error: 5.1806671508281e-07
Method 1 Wall time: 4.5959949493408206e-06
Method 2 Wall time: 0.0003765707015991211
As we expected, method 1 runs much faster than method 2 on average, method 2 is much more accurate.
To test the 2 methods of interpolation, I used the following function to define f.
$f(x, y) = cos(\pi x) + sin(\pi y)$

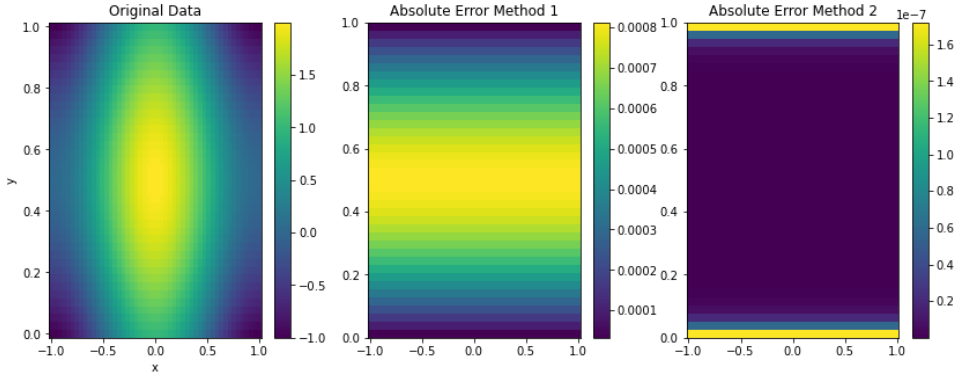


Figure 8: Figure 2.1

As we can see from the figure, overall, Method 1 has high error areas around the middle of the plot where the original data is at it's peaks and troughs. For Method 2, there is low error around most of the areas except for the boundaries.

Part 3

1.

We start by looking at the contour maps of u for different values of c. For values less than 1.3, we notice that the system stays stable for the most part. However, if c becomes greater than 1.3, we see that the system becomes more and more unstable as c increases. Therefore, we will studying the behavior of the solutions for c = 0.5, 1.0, 1.3, 1.5
With this knowledge, we will first look at the dominant frequency of the simulations to see if there are any trends.

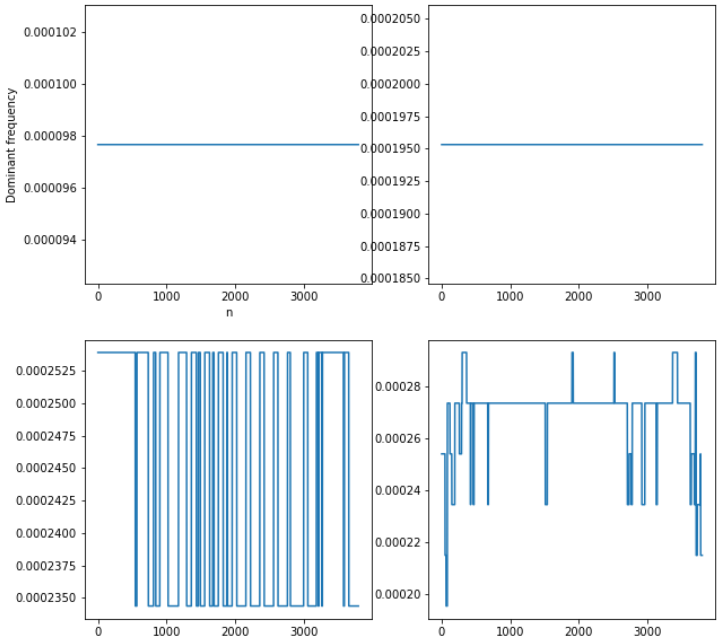


Figure 9: Figure 3.1

From the figure, we notice that the dominant frequency is constant for c = 0.5 and c = 1.0. However, for c = 1.3 and c = 1.5, the dominant frequencies for each simulation are unstable, suggesting that there are could be chaotic behavior within the solutions of the system for high c values.

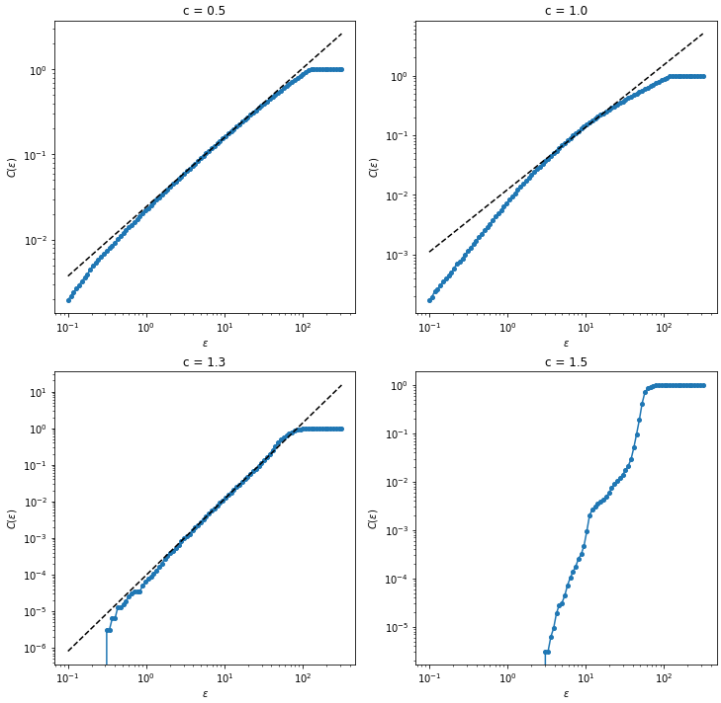
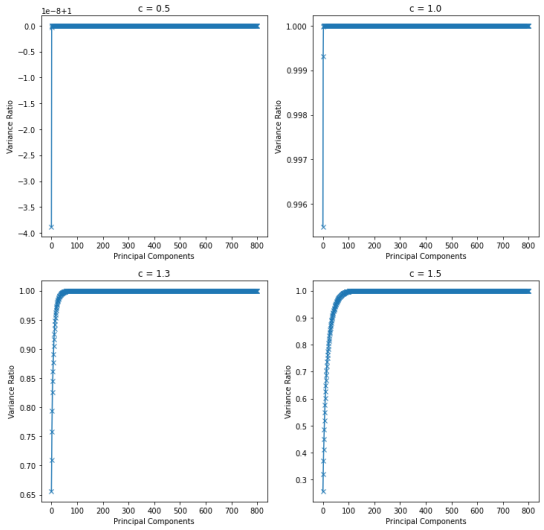
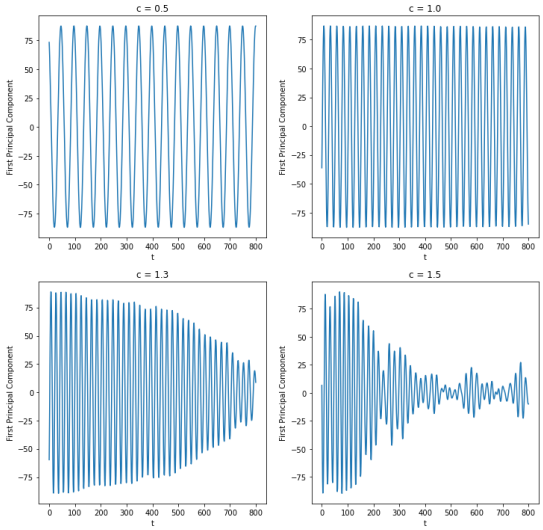


Figure 10: Figure 3.2



(a) Figure 3.3



(b) Figure 3.4

From figure 3.3 and 3.4, we notice that for c = 0.5, the first principal component accounts for most of the variation of the entire simulation. This leads to the conclusion that c = 0.5 is very close to simple sine oscillations(= simple sinusoidal oscillations) because other oscillations that make the curve noisy have very little significance. For c = 1.0, it is similar to c = 0.5 but to a lesser extent because the first principal component accounts for less variation compared to the case for c = 0.5. For values greater than or equal to c = 1.3, it becomes very difficult to explain the behavior of the solutions because the multiple principal components have considerable significance.

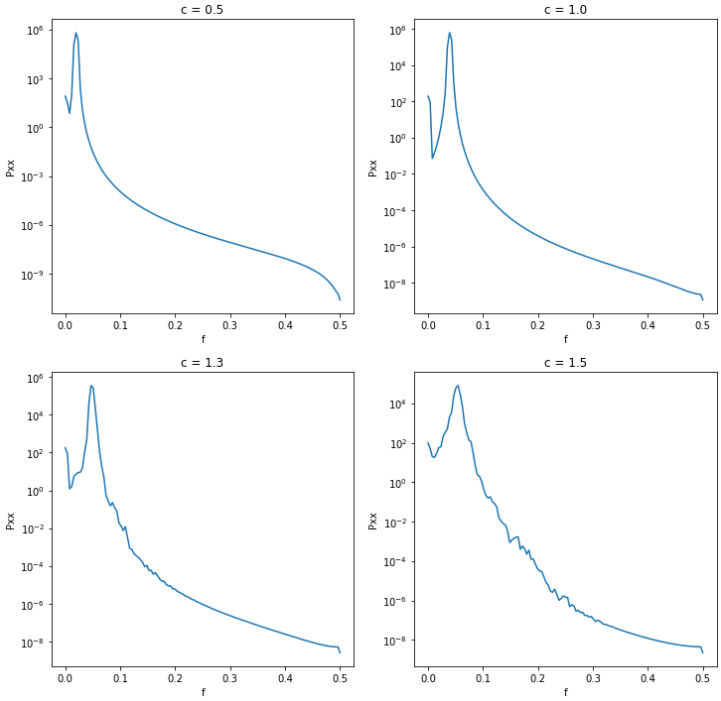


Figure 12: Figure 3.5

As we can see from Figure 3.5 which are plots that I obtained using the welsh method, the line of power spectral densities is less smoother for c = 1.3 and c = 1.5.

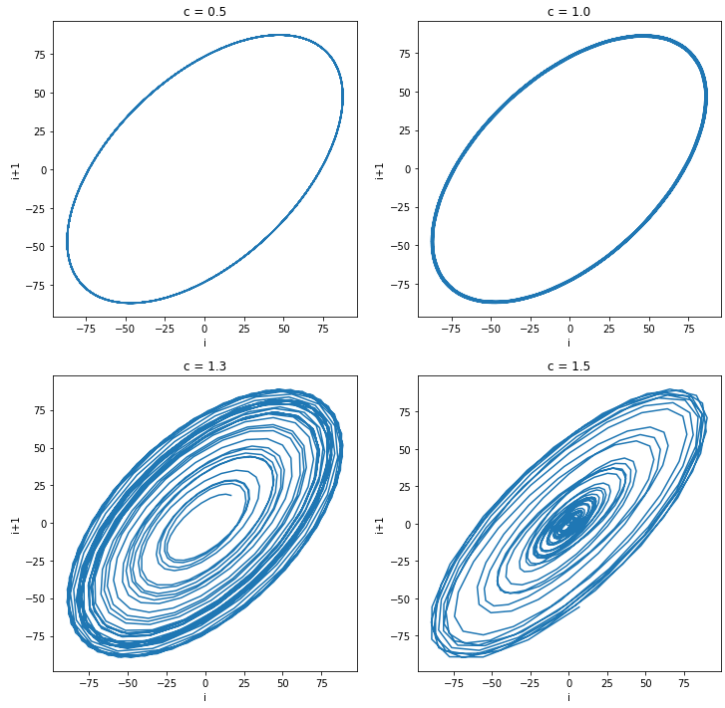


Figure 13: Figure 3.6

We will now perform time-delay reconstructions of nonlinear dynamics introduced in lab 9. For c = 0.5 and c = 1.0, we see a simple oval graph, representing the simple sinusoidal oscillations of the two solutions. For c = 1.3 and c = 1.5, we can see the solutions slowly converging to the centre of the oval suggesting that the systems could potentially be chaotic.

2.

```
l1 , v1 = np.linalg.eigh(A.T.dot(A))
v2 = A.dot(v1)
A2 = (v2[:,:x]).dot((v1[:,:x]).T)
e = np.sum((A2.real-A)**2)
```

First, we will analyse the 4 lines of code. The first line is calculating the eigenvalues(l1) and the eigenvectors(v1) of $A^T \cdot A$. In the second line, eigenvectors(v1) are multiplied to A for linear transformation. In the third line, we are reconstructing the matrix $A2$ with the using the first x number of columns. Finally, in the last line, calculate the squared error of the reconstructed matrix $A2$.
The final 4 lines of code produces results equivalent to the low-rank approximation using SVD.
$v2 \cdot v1^T = A \cdot v1 \cdot v1^T = A \cdot I = A$
We are taking x number of columns from v1 and v2 which results in a lower rank matrix. In terms of efficiency, we can say that the 4 lines of code in *part3q2* is efficient compared to the method using SVD because when we are performing low-rank approximation. We only need the eigenvector of $A^T \cdot A$ and *np.linalg.eigh* is sufficient for that task. Therefore, the computational cost would be lower than the case where we are using *np.linalg.svd*.
