Scientific Computation Project 3

02099078

December 15, 2023

Part 1

We aim to apply PCA to help understand how the daily average zonal wind speed fluctuates throughout the year. PCA cannot be applied to the data as it is given as an array in 3D, so we must first flatten it. This consists of reshaping the array to the shape (365, 16*144). The data is then centred using the mean so that the data now has a mean of 0, and then the singular value decomposition (SVD) of the array is computed.

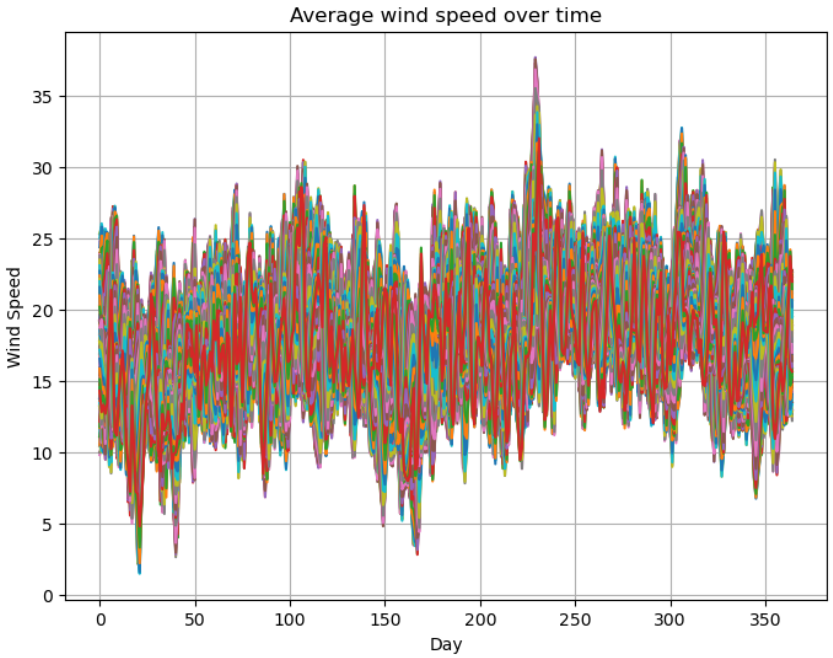

Figure 1: Average wind speed over time in the southern hemisphere.

Figure 1 shows the average wind speed over time throughout the year. It shows that the highest average value to be approximately 38 at around day 230, and the lowest average value to be approximately 2 at around day 20. Overall, the second half of the year appears to have stronger winds than the first half of the year. This is what we would expect since the data is regarding the southern hemisphere.

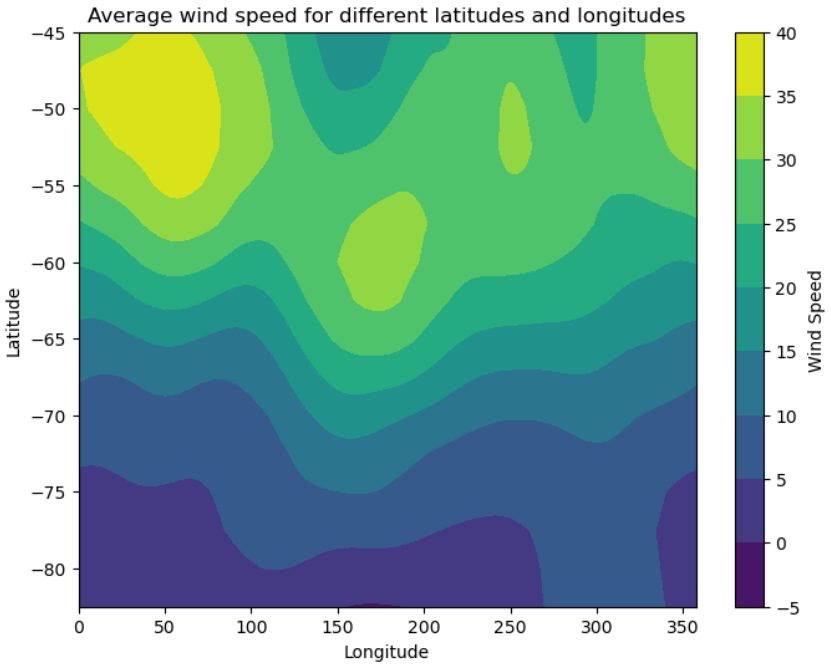

Figure 2: Average wind speed for latitudes and longitudes in the southern hemisphere.

Figure 2 shows the average wind speed for each latitude and longitude. It shows that the highest average value shown in Figure 1 occurs at approximately -50 latitude and 50 longitude. The contour plot shows that for very negative latitudes in Antarctica, the wind speed is very low, but as we get closer to the latitudes near the top of figure 2, the wind speed becomes very high. The contour plot seems to be roughly vertically symmetrical and so we can conclude that longitude does not seem to affect wind speed, whereas increasing latitude does increase wind speed.

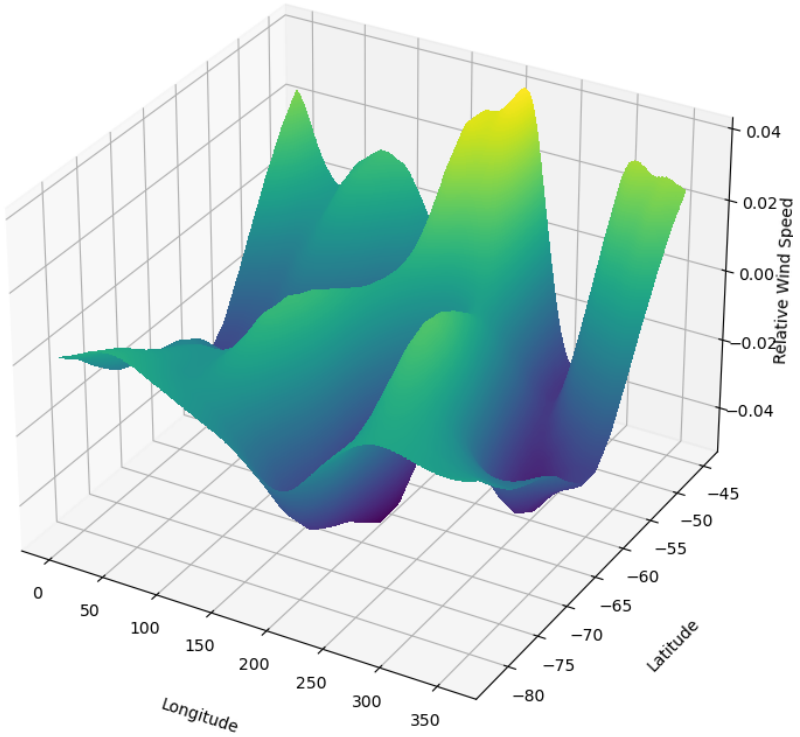

Figure 3: Relative wind speed for latitudes and longitudes in the southern hemisphere.

To create figure 3, we apply PCA and then the first eigenvector is reshaped in order to allow us to plot the relative wind speed for each latitude and longitude. This is useful as it now tells us information about the direction of the air flow. There appears to be a lot of positive zonal wind speed in areas of high latitude and central longitude (around 200 longitude). These are areas where we have eastward air flow. On the other hand, there seems to be a lot of negative zonal wind speed, corresponding to westward air flow, where there are dark spots shown on the plot. This happens when either latitude is very low and longitude is central or when latitude is central and longitude is very low/high.

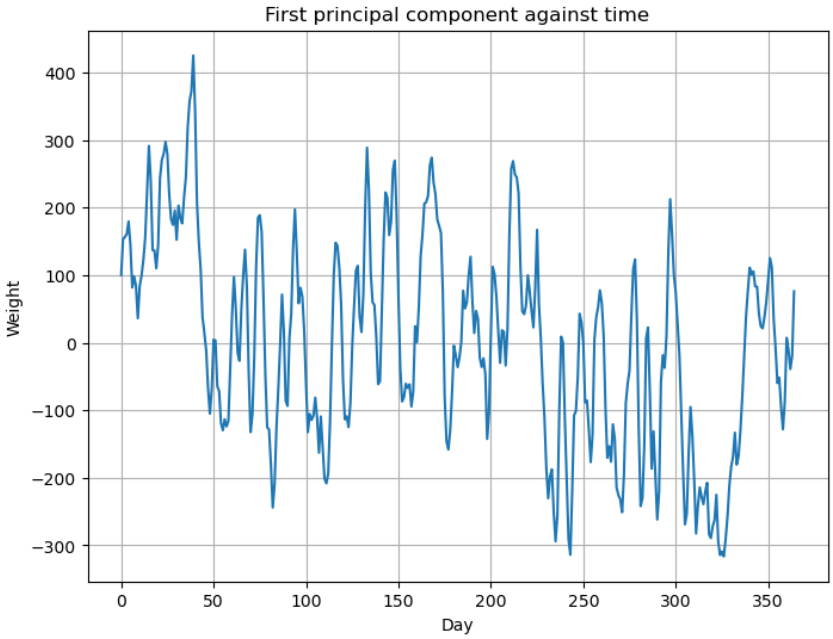

Figure 4: First principal component's components against time.

Plotting the first principal component's weight against the day of the year gives us useful information regarding the accuracy of our results in figure 3. Days with a larger weight will be more accurately represented by the first principal component. This is because they have a larger impact on the relative wind speed data and they capture more of the variance present in the original data. Interestingly, the graph in figure 4 shows that the weight decreases as we move towards the end of the year, meaning that the wind speeds are less accurate for these times.

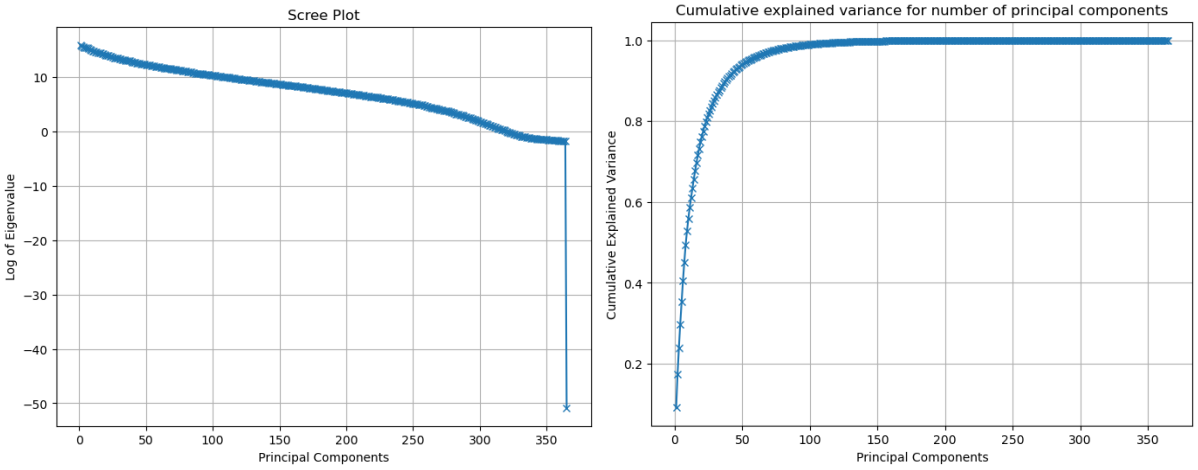

Figure 5: Scree plot of the log of eigenvalues against the number of principal components and the explained variance.

Next, we create a scree plot to determine how representative our results in figure 3 are compared to the original data. Figure 5 shows that the eigenvalues are decreasing but not at a very high rate and so the first eigenvector alone is not enough to accurately represent the data. We need a large number of principal components for this. In particular, we would ideally want to select the number of components that has a cumulative sum of explained variance close to 80%.

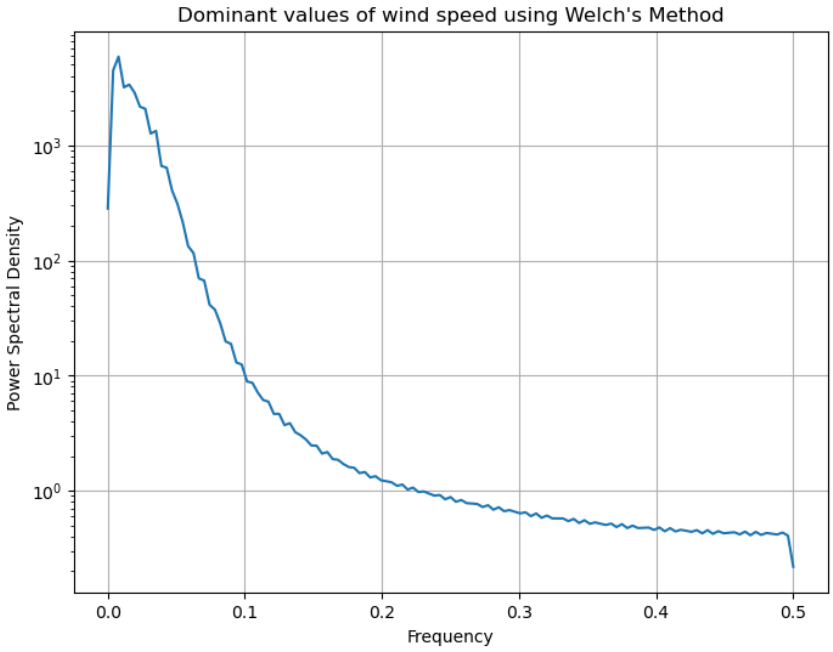

Figure 6: Welch's Method applied to wind speed data.

Finally, we use Welch's method to find the most prominent frequencies in the wind speed data. The peak of the graph is at f = 0.0078125, and so $1/f = 128.0$ time steps or 0.3506849315 days, meaning the data exhibits periodic behaviour. Experimenting with the plot_field function at various times helped to determine that the wind is moving eastwards. Therefore, we can conclude from our analysis that the wind can be modelled as a sinusoidal wave generally moving eastwards.

Part 2

2.

Method 1 involves performing addition for each element in the m x n array f. This results in the time complexity of method 1 being $O(mn) = O(m^2) = O(n^2)$ as it is solely dependent on the size of the array. Method 2 involves using banded matrices. Using solve_banded to solve the banded matrix equation is the most significant contributor to the computational cost of method 2 (since setting up the banded matrix has linear time complexity). This has a time complexity of $O(m^2) = O(n^2)$, which is the same order as that of method 1.

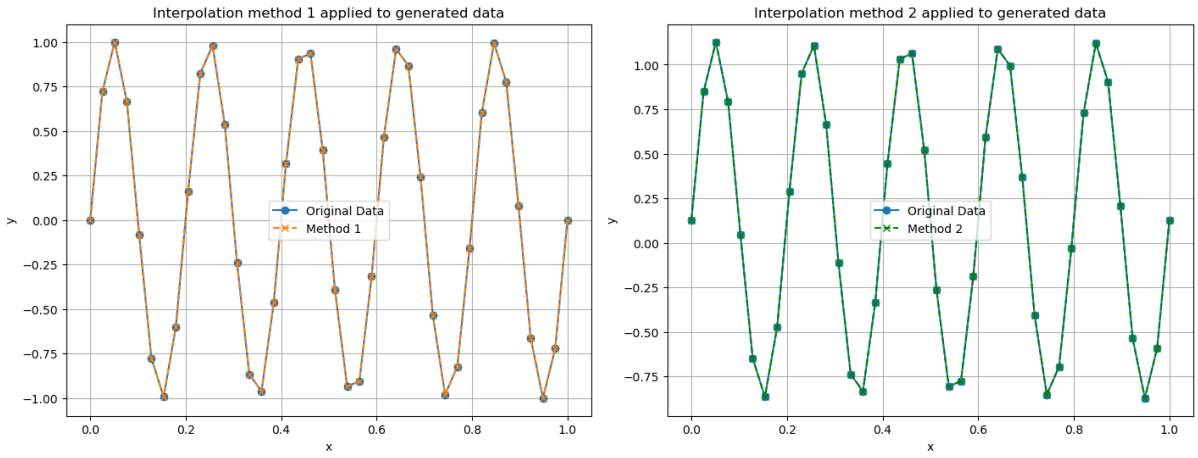

Figure 7: Interpolation methods applied to generated data.

To analyse the accuracy, we use finite difference approximations to find the order of the error terms for both methods. Computing these by hand, we find that the error term for method 1 is $O(h^2)$, whereas the error term for method 2 is $O(h^6)$, where h is the grid spacing. This means that method 2 provides a much higher level of accuracy than method 1 in general. However, figure 7 shows that for simple, smooth data, method 1 is still very accurate, and is a simpler alternative. For data containing many variations and sudden changes, it becomes much more clear that method 2 is significantly more accurate.

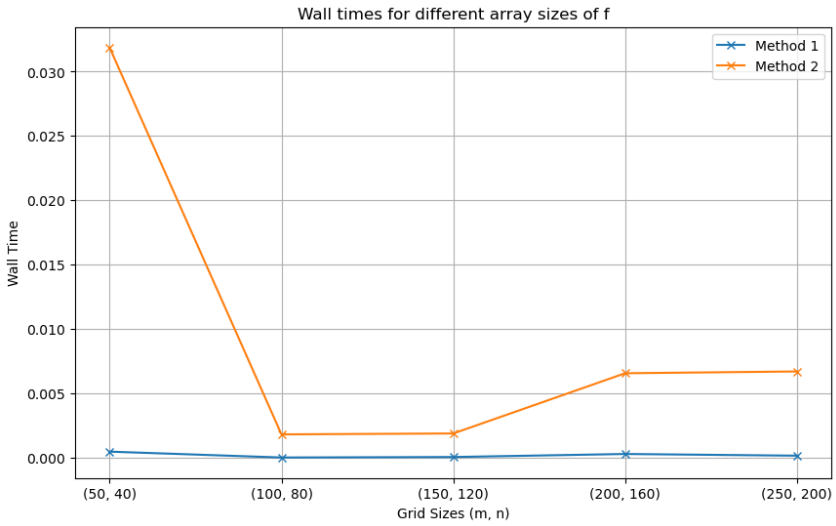

Figure 8: Wall times using both interpolation methods for different array sizes.

To compare the efficiency, we take into account the wall time differences between the two methods, as well as memory costs. Figure 8 shows the wall times for different sizes of the array f. It is clear to see that method 1 is quicker than method 2 for all array sizes. However, method 2 is more efficient with regards to memory due to the structure of the banded matrix.

Overall, both methods have their advantages. Method 1 is quicker and interpolates smooth data very well, whereas method 2 is a bit slower but more memory efficient, and interpolates non-linear data much better.

Part 3

1.

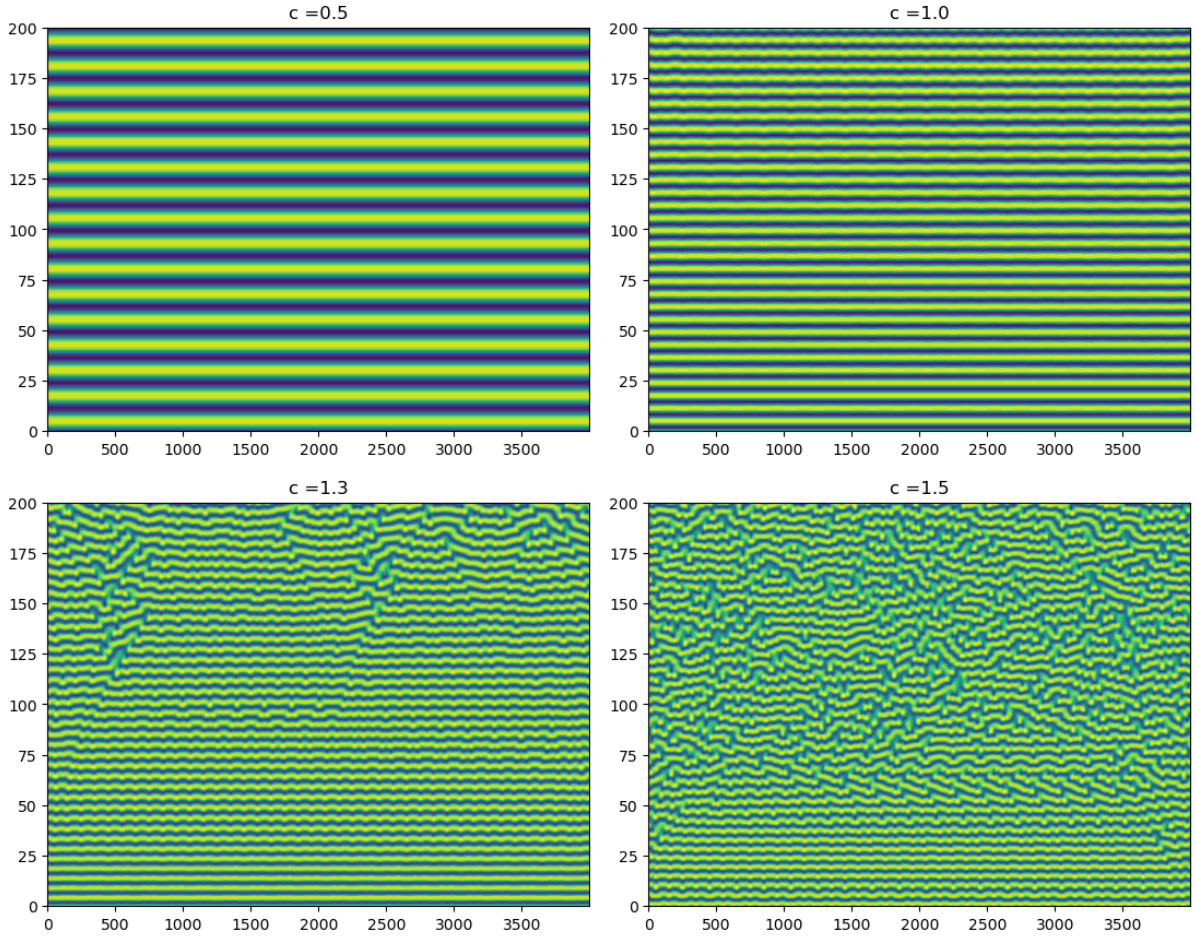

Figure 9: Contour Plot for 4 values of c.

Figure 9 shows contour plots for four different values of c. I picked these values since they all have distinct differences which help to describe the system. The x axis shows the trajectories u_i and the y axis shows values of time up to t = 200. We can see on the plot light and dark horizontal lines that represent the peaks and troughs of the trajectory oscillations. For c = 0.5, these lines are clear for all times, meaning all the oscillations are aligned in phase. For c = 1.0, these lines are initially clear, but then as time increases, they seem to get slightly blurry. This means the peaks and troughs are no longer aligned for each trajectory, and so the oscillations are not in phase. For c = 1.3, we begin to see large blurs where the peaks and troughs become very random and unpredictable. These blurs represent chaotic behaviour and for larger times, it is clear to see that the system is more chaotic. The oscillations appear to be periodic until roughly t = 100, where they then become aperiodic. For c = 1.5, a large part of the graph is covered by large blurs, and we have chaos at all time values.

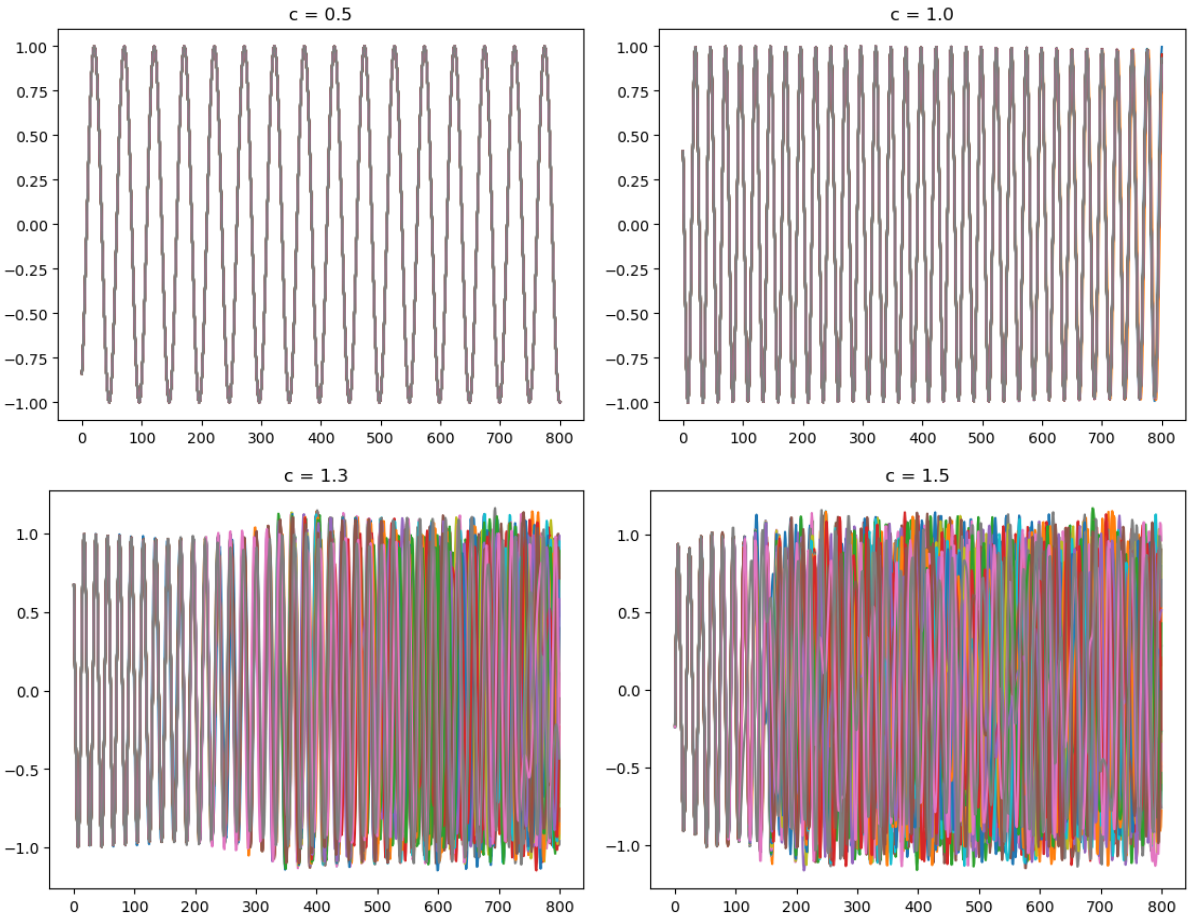

Figure 10: Trajectories for 4 values of c.

To see the behaviour of the trajectories more clearly, I plotted them for these same four values of c. Figure 10 shows that for c = 0.5 and c = 1.0, we can see simple sinusoidal behaviour whereas for c = 1.3 and c = 1.5, we can see the system is very chaotic. This agrees with what the plots in Figure 9 showed us.

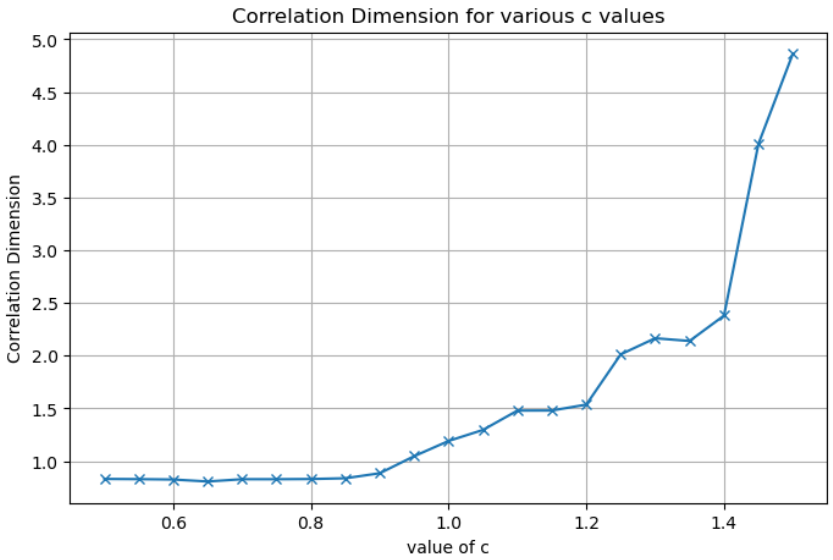

Figure 11: Correlation Dimension for 10 values of c between 0.5 and 1.5

Next, I looked to investigate which values of c display chaotic behaviour. For this, I computed the correlation dimension for ten values between 0.5 and 1.5 and plotted them on the same graph, as shown in Figure 11. From lectures, we know that for a system to be considered chaotic, it must have a correlation dimension of at least 2. Our computation gives that for c = 1.3, the correlation dimension is 2.1651410483718423. Moreover, Figure 11 shows us approximately that for values of c greater than 1.3, the system is chaotic as the correlation dimension is greater than 2, and for values of c less than 1.3, the system is simple sinusoidal as the correlation dimension is less than 2. We can also see that increasing c in the range between 0.5 and 1.5 results in the system becoming more chaotic, and the rate at which the system becomes more chaotic increases as c increases.

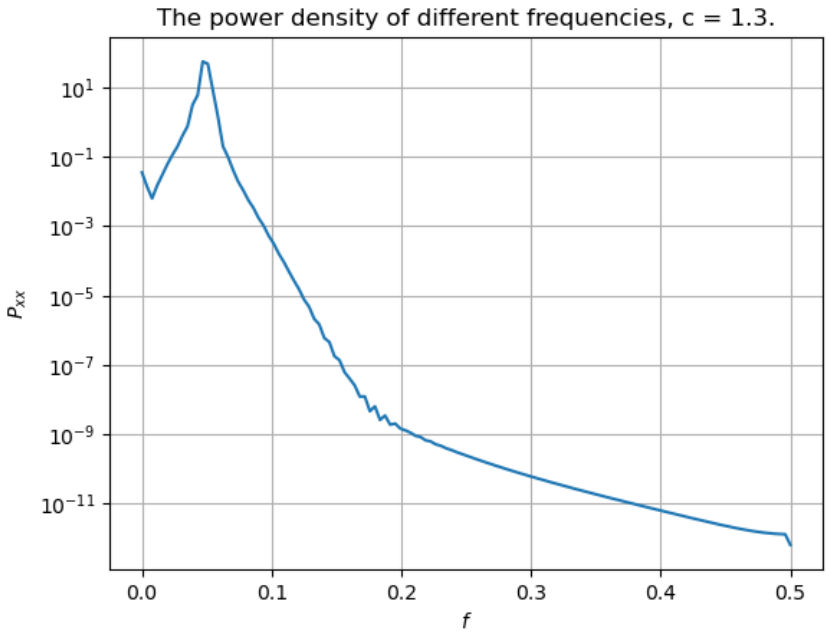

Figure 12: Welch's method for c = 1.3

Finally, I used Welch's method to plot the most dominant frequencies for c = 1.3. As shown in figure 12, we get that the peak of the graph is at f = 0.046875, and so $1/f = 21.333333333333332$ time steps or 5.33333333333s. This matches up with figure 9, where the time between consecutive light horizontal lines appears to be the same.

2.

The first line of the code takes A, the matrix representing the solution to the system of ODEs. It is multiplied by the transpose of A, forming a symmetric matrix. The eigenvalues 'l1' and eigenvectors 'v1' of the symmetric matrix are then calculated. This is useful for dimensionality reduction.

The second line of the code performs matrix multiplication on A with the eigenvectors calculated in the previous line. This creates a new set of vectors 'v2' that have been transformed.

The third line constructs a new matrix 'A2' by matrix multiplying the first x columns of 'v2' with the transpose of the first x columns of 'v1', both calculated in the lines prior. This new matrix is a rank-x approximation of the matrix A. Low-rank approximation is useful for understanding and analysing the system of ODEs because it reduces the computational complexity, especially if A is large. It also filters out noise that may be present, while retaining the important behaviour of the system. This makes analysing the system as a whole simpler.

The fourth line sums the square of the differences between the elements of A2 and the elements of A. This provides a measure of the approximation error between the solution matrix A and the rank-x matrix A2.

The four lines should be considered to be very efficient since SVD-based rank-x approximation is the most efficient low-rank approximation. However, this code takes the lowest singular values which means that the matrix may be reconstructed poorly, as these values may carry information that is important for accurate representation.
