Scientific Computation Project 3

Jonathan Cheung CID:01063446

March 27, 2020

Part 1

1.1)

The code in the function 'repair2' will factorise an input matrix R into an A and B matrix, which minimises some cost function defined in the assignment. 'repair2' is more efficient than 'repair1' for a few reasons. The most significant improvement is that 'repair2' uses the np.dot function to vectorise the calculation of the sums that form the A and B entries. This is much quicker than the two nested for loops used in 'repair1'. So 'repair2' will have a time complexity O(n) and 'repair1' will have $O(n^2)$. The other improvement made is that a dictionary has been used instead of a list. To calculate the sums for some A_{mn} we need to find indexes j such that (m,j) refers to a non empty (non -1000) value in R. With this dictionary, we can search the dictionary for a key value of m to receive the list of j indexes we want. This is faster than searching through a list of lists.

The missing values in data1 can be seen as the purple patch (fig1). Repair2 produces fig2. The function has been fairly successful as the purple patch has been mostly filled in and it has retained the features of data1; you can still see the horizontal waves.

The parameters used for this reconstruction were p=36, l=1, niter=51. p was chosen to be this value since this is the rank of the original data1 matrix. By choosing this rank, we are not introducing any new rank or trends into our repaired data. l was chosen to be this so that the height values are roughly similar and this can be seen as the shade/colour of the plots are similar. niter was chosen to be this value to give a reasonable runtime. Perhaps this result could be improved by increasing iterations and optimising l so that the colour of the plot of the repaired data is also the same.

1.2i))

1.2ii)

We can look at the waves along the radius for the different angles for a fixed time. The function fix_t_plot will plot the waves at the three theta values (pi/4, 3pi/4, 5pi/4) for a single time. We call the waves formed by these three angles wave1 wave3 and wave5 (the number denotes the constant that multiplies pi/4). We produce fig3a by creating this plot at time 0 and notice that it is hard to compare the waves to each other. fix_t_plot also returns means and variances of the separate waves. So at time 0 we see that wave3 has the lowest mean height along the whole radial length (-0.76), whereas wave3 and wave5 had means around 0. The variance of wave5 was the highest. These values do not tell us the full story, since each wave here is a result of multiple waves. To examine these height dynamics further we use the function fft_tplot. This will calculate the Fourier tranform coefficients for a wave at a certain time. Looking at the Fourier transform coefficients at time 0 for all three waves (fig3b-3c), we see that all waves have sin waves with coefficients between 0 and 0.7, but wave3 has one significantly more large coefficient around 1.5 and wave5 has a few coefficients around 1.2.

If we look at time point t=60, we can form a new set of graphs. 'fix_t_plot' will display all three waves and point out that wave3 has the largest mean and wave5 has the largest variance (fig4a). The main difference between the waves at t=60 and t=0 is that wave3 has a much larger mean, it has changed from -0.76 to 1.78. We will plot the Fourier coefficients to explore this further (fig4b-4c). The Fourier coefficients at t=60 for wave1 is very similar to the ones at t=60. However, wave3 now has a coefficient of 3.5. wave5 has a similar range of amplitude coefficients, but these are distributed differently (in both amplitude and frequency).

In fact, if we are to apply a Fourier transform to the waves at time points between t=0 and t=118 we will continue to see that wave1 has coefficients consistently between 0 and 0.7, wave5 has a large variance that changes with time and coefficients randomly between 0 and 1.3, and wave3 will have many coefficients less than 0.5 and one significantly large coefficient that varies between 0.7 and 3.5.

In the context of waves in the ocean what could this mean? Perhaps the waves along the pi/4 line are fairly steady and small throughout the full time period. The waves along 5pi/4 can be interpreted as slightly taller than in pi/4. The much larger variance in wave5 compared to other waves (approximately 4 compared to 1) and the fact that the Fourier coefficients change as time changes shows that wave5 is chaotic; it has different periods and Fourier coefficients for each time point. So this wave5 could be turbulent waves. Lastly, the large coefficient from wave3 could suggest there is a large wave along the 3pi/4 line.

While there are time points where the coefficients from wave3 are smaller and more reasonable, this could be seen as a time where the wave has not formed yet, such as low tide or a calm before the storm.

1.3)

The function 'reduce' will take as its input a 3d array and return a tuple of arrays that contain the key features of the original array, but is much smaller in dimension. This function works by converting the 3d array into a 2d array, and then applying PCA on this 2d matrix and returns the transformation matrix U, the reduced size matrix G, and an array of dimensions. The function 'reconstruct' will take this output as its input and reconstruct an approximation of the original dataset with the same shape. This reconstruction works by applying the transformation U back to G. This works because U is orthogonal, so U is the inverse of U^t. Lastly, the array is reshaped back to the original dimensions.

PCA works by finding axes that maximise variance and then only saving information along these axes since they are seen as the most important axes. Each axes is a combination of the original variables in the dataset. In the context of waves, perhaps key features could be interpreted as variables that define the individual waves such as Fourier coefficients. Figure 5 shows an hfield plot of data3 at time 0 and we can compare this to figure 6 which is a plot of a reconstructed data3. This has been quite successful as the plots look very similar; we can see the same wave shapes in both. Fig 6 has a grainier resolution than fig 5, but this is to be expected since there is less information used to create fig 6.

The size reduction can be modified by changing the input 'inputs'. By default this is set to 12, increasing this would reduce the memory savings, but increase the accuracy of the final image. The file data3 is of dimension (300,289,119) and this is reduced to matrices of size (12,300*119) and (289,12). The first matrix is the significant one. So using our function onto data3 with parameter 12, we have achieved a memory reduction of almost 95% (by reducing 289 into 12). See below for an hfield plot of data3 at time 0 and and an hfield plot of the reconstructed data3 at time 0.

Part 2

2.1)

'newdiff' takes as its input, an array of function points. It then calculates the second order derivative using a compact finite difference scheme. The function works by forming Ax=b, where A is the matrix of coefficients that multiplies x, a vector of second derivatives to form the LHS of the given compact scheme. The RHS of the scheme relates to b, a vector of values formed by taking linear combinations of the input array values. This is a banded system and we use a built in function to solve this to return x, our array of second derivatives.

This function has time complexity O(n), where n is the length of the input array. This can be verified with fig. The implementation is efficient because the creation of the b vector has been vectorised and the banded system solver is very efficient. The banded system solver does not waste time with calculations involving the zero entries in A. I believe this function might use a lower and upper decomposition method to break down the matrix A and then forward and back substitution to solve the equations.

For an example function sin(x) where x ranges between 0 and 2pi, we can plot the errors of our function by plotting the output of newdiff against -sin(x), which is what we know the second derivative to be. In fig we can see that the error is fairly large at the boundaries, but effectively non existent for the values in the middle. This compact scheme is a higher order than a second order centred scheme, and thus more accurate and stable. The error from this newdiff scheme at the boundaries can then be expected to be lower than the errors for a centred second order scheme. Newdiff in general is a good scheme to use for large multiscale problems due its runtime and relatively high accuracy as long as the function is periodic. Though, to consider when it is best to use which scheme, we can plot wavenumber against second derivative for different schemes and look at which lines lie closest the true second derivative plot (http://acoustics.ae.illinois.edu/pdfs/lele-1992.pdf?fbclid=IwAR1O-acGOkb7EE7nBP8CgS317RDqOHAQvMCArmr1amxQvHNAV3KPKz9HwO0, page24).

2.2)

Using the 'microbes' function we can plot the simulation results for kappa = (1.5,1.7,2). The patterns of these contour plots look like fractals, but it is hard to tell if the behaviour is chaotic. From fig9,10,11, we can estimate the period by looking at horizontal lines (ie fixed time) and seeing how the colour changes. Perhaps we can deduce that there is no fixed period since the contour change is different for each time point.

To investigate further, we will use correlation sum plots for each of these

kappa values. Using pdist we create Figure 12 where we can see a correlation sum plot for κ=2. We can then hone in on the middle section (epsilon =[2,6]) and do a new plot and calculate the gradient using np.polyfit (fig13). Fig 14 and Fig 15 correspond to the correlation sum plots for κ =1.7 and 1.5 respectively. The gradients for κ=2, 1.7, 1.5 are 2.53,2.61, 2.66. These are all greater than 2. We conclude that the behaviour for $\psi = 0.3$, $L = 1024$, $Nx = 1024$, $\mu/\kappa = 0.4$ and $\kappa = 2, 1.7, 1.5$ is chaotic. So there is no period or structure to how the concentration of the 'f' microorganism changes when satisfies these parameters

2.3)

With the given solutions to the model, we can treat these as initial conditions. Using odeint, we can integrate forward using these solutions as initial conditions. Then we run the model simulation for large time T¿¿t and save this solution. We repeat this process but by taking perturbed versions of the original initial condition solution (add a small value to the space initial condition). These perturbations should be sinusoidal as well (ie x + ϵsin(n), where n is grid of of points). After this we will have a series of solutions and we can compute the distances between the adjacent solutions. And then we can use a least squares method to calculate the slope of the distance plot. This gradient is two times the Lyapunov exponent which tells us how sensitive the solutions are to the perturbations for finite time. For infinite time we expect that the distance between the solutions will have some upper bound.

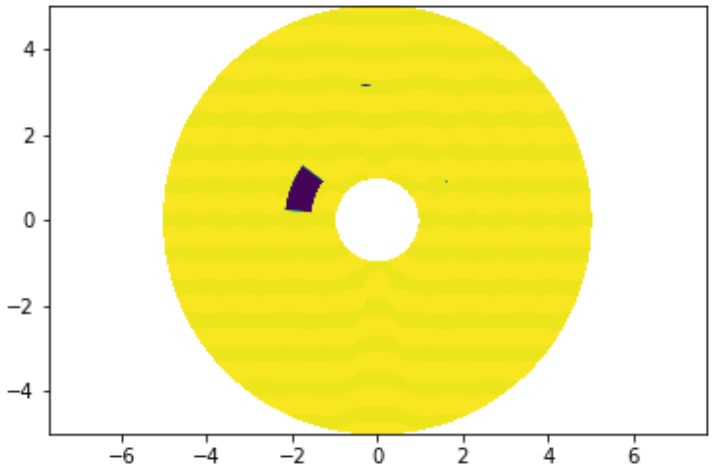


Figure 1: hfield plot of data1

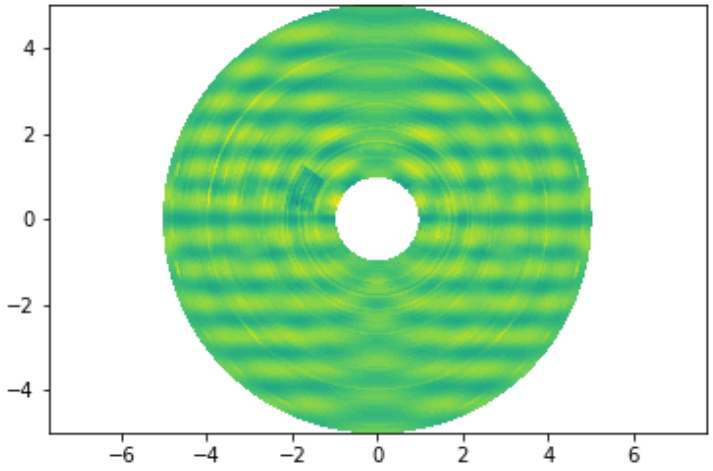
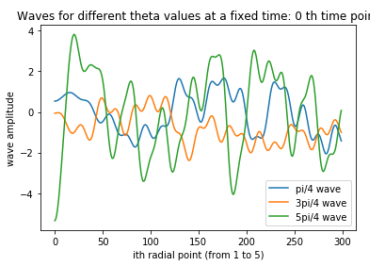
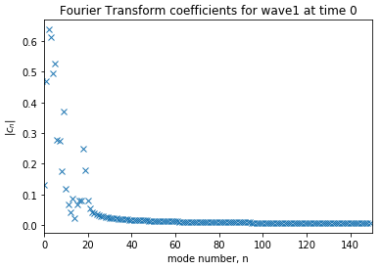


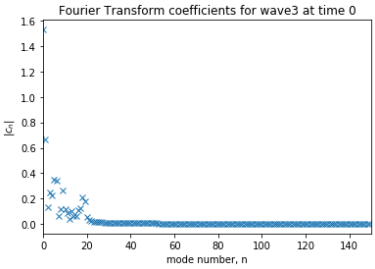
Figure 2: hfield plot of a repaired data1



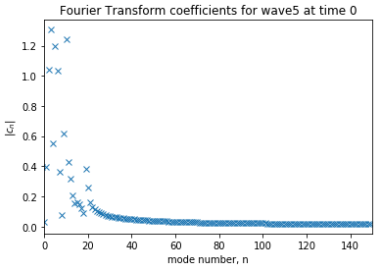
(a) plot of all waves



(b) fft of wave1

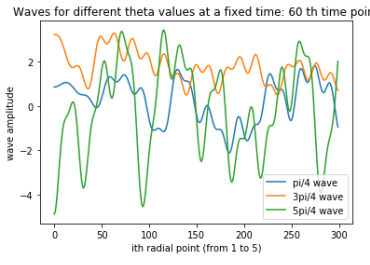


(c) fft of wave3

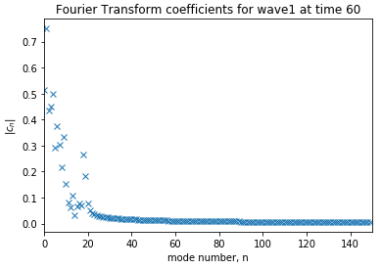


(d) fft of wave5

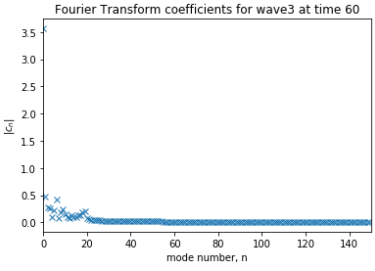
Figure 3: plots of waves and fourier coefficients at time t=0



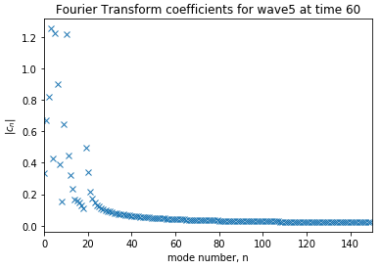
(a) plot of all waves



(b) fft of wave1



(c) fft of wave3



(d) fft of wave5

Figure 4: plots of waves and fourier coefficients at time t=60

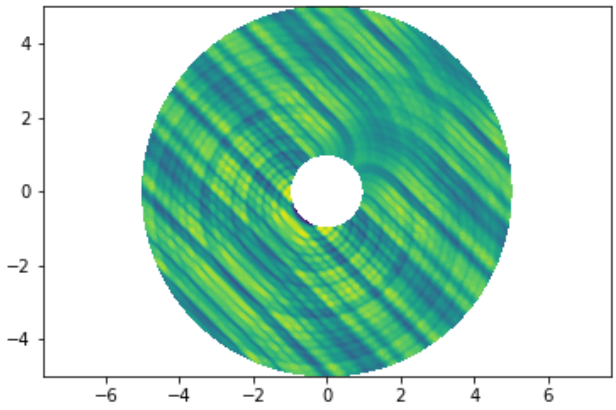


Figure 5: hfield plot of data3 at time 0

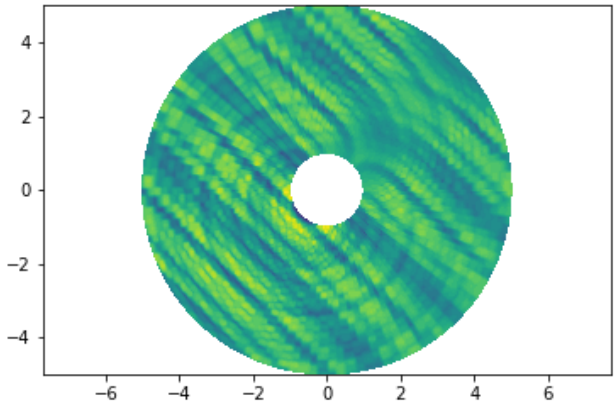


Figure 6: hfield plot of a reconstructed data3 at time 0

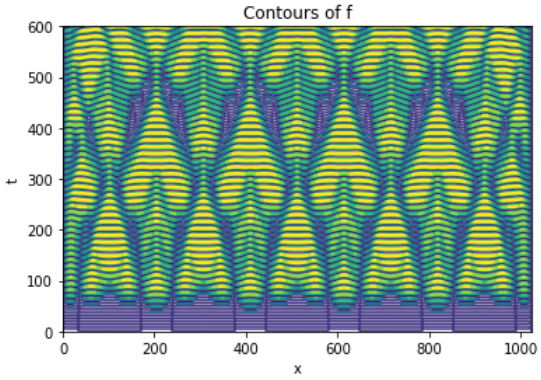


Figure 7: microbes contour plot for k=2

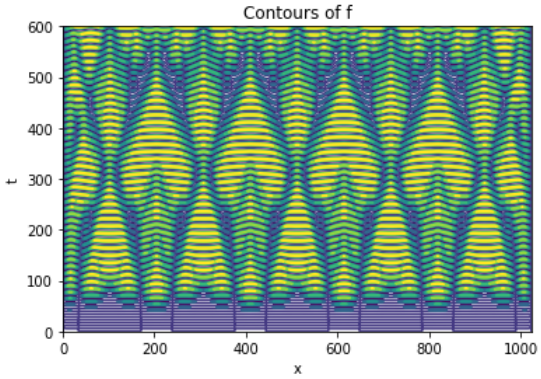


Figure 8: microbes contour plot for k=1.7

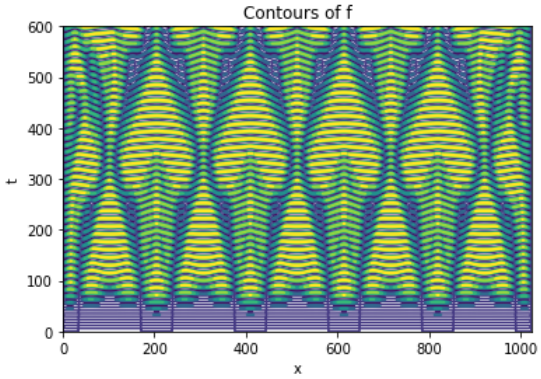


Figure 9: microbes contour plot for k=1.5

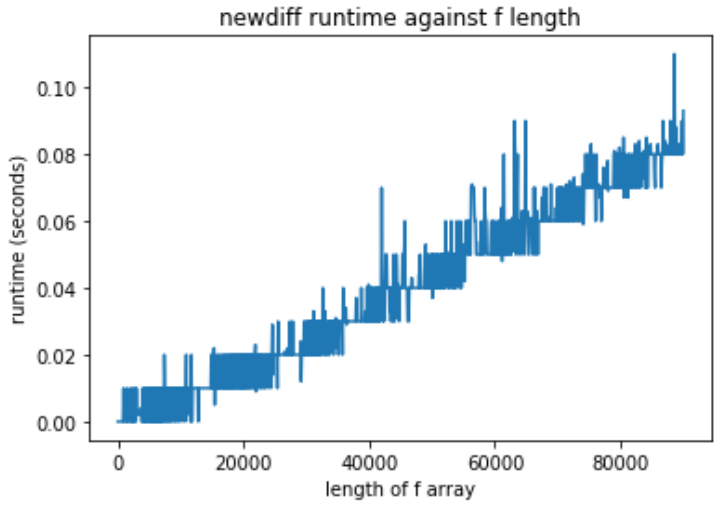


Figure 10: runtime for newdiff applied to a sine function with varying number of points

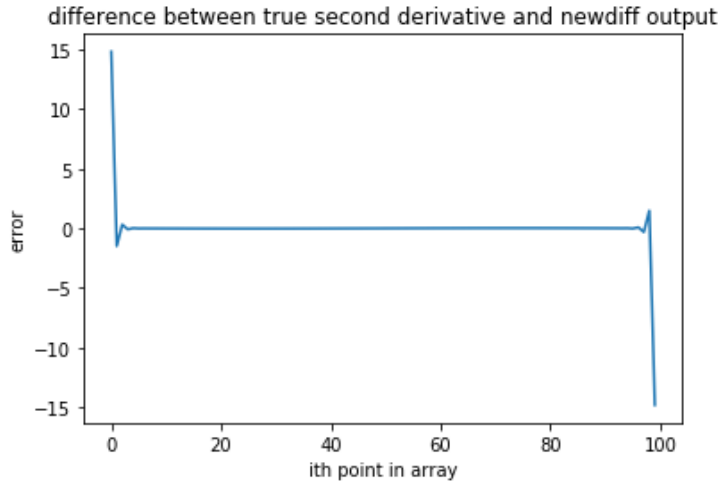


Figure 11: The difference between the newdiff derivative and the theoretical derivative

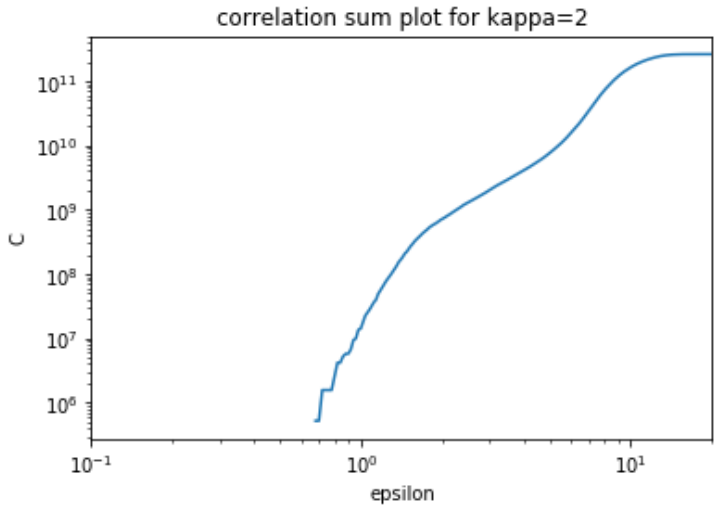


Figure 12: Correlation sum plot for k=2, with the large range of epsilon

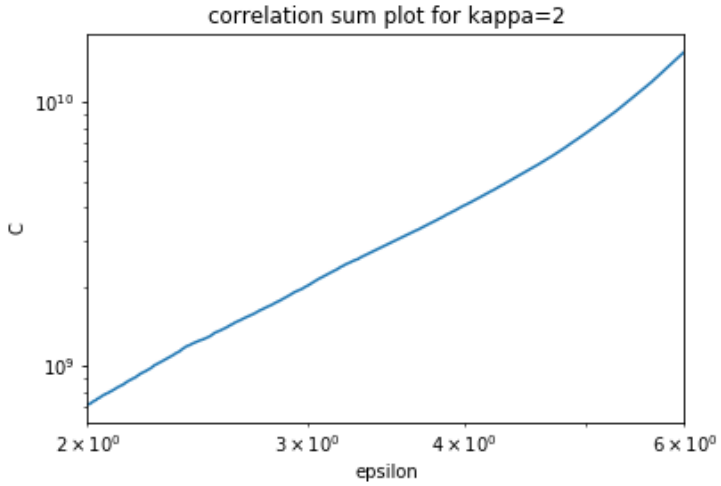


Figure 13: Correlation sum plot for k=2, with the narrowed down range of epsilon

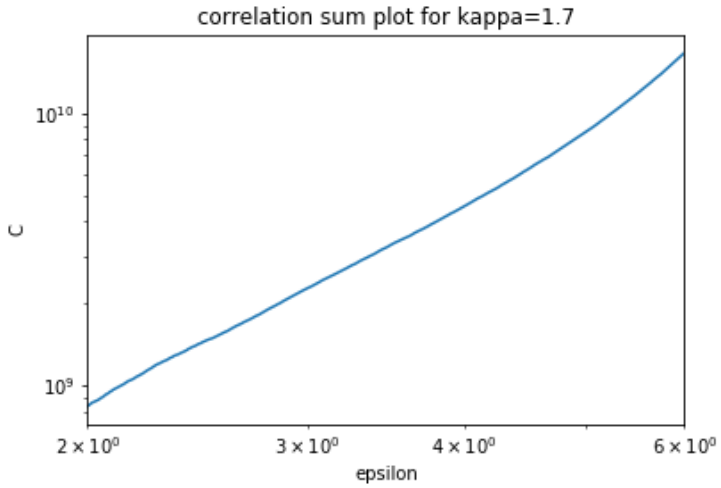


Figure 14: Correlation sum plot for k=1.7, with the narror range of epsilon

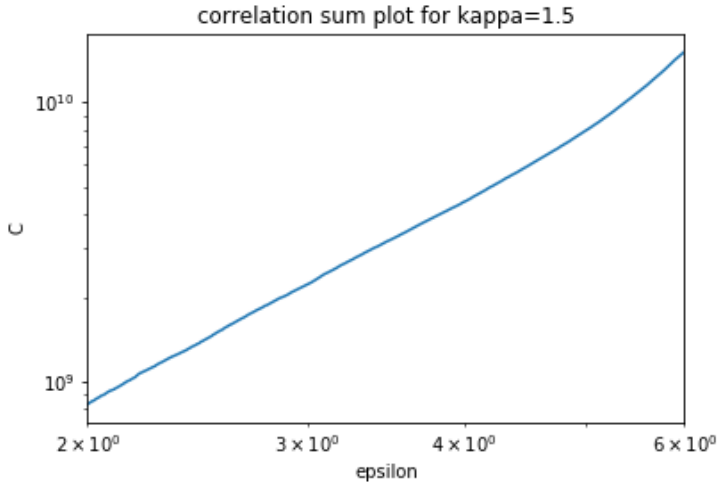


Figure 15: Correlation sum plot for k=1.5, with the narror range of epsilon