For the first exercise I created a function called *Laplacian()* which took in two parameters, a matrix, *U*, of *m* rows and *n* columns, and a special step size, *k*. The function first extracts the size of the matrix and sets the size of the rows and columns to separate variables. After that function handles are created to deal with the boundary conditions. In this exercise, and the following exercises, periodic boundary conditions were used. The conditions state that the last and first index of an array are equal and that the set repeats. For example if I wanted the value at a position x-1, where x is the first index, I would get the value at the second last index. Then I reserved a matrix with a size of *m* rows and *n* columns to be used for the laplacian matrix. Then I used the equation given in the powerpoint, the discretised laplacian operator, in order to calculate the Laplacian of the matrix *U.* The output is a matrix of size *m* x *n.*

In the second task I created a function called *Diffusion\_Eq()*  which took a multi variable function, *f(x,y),* a special step size *k*, the maximum values for the *x* and *y* coordinates, the maximum time, *T,* over which to perform the operation, the constant of diffusion *kappa*, and a difference in time, *h.* The first process was to discretise the function. To do this I calculated *mx* the amount of divisions we will split the *x* coordinates into by dividing the maximum value of *x* by the special step size, similarly for the *y* coordinates. Then I calculated the number of steps of time. I used a nested for loop to calculate the initial values of the output matrix. Each for loop loops an amount equal to the *mx* variable and *my* variable. The matrix that is obtained from a multi variable function evaluated at different combinations of *x’s* and *y’s.*  Then I used the Euler approximation iterated over *T/h* times to calculate the upcoming matrices. When creating a movie of the output of this function, higher values are on the outer edges of the graph and they slowly decrease over every iteration and I don’t think this is correct. I used the *del2()* built in function to calculate the Laplacian and the movie created looks more correct, the outer edges slowly increase in magnitude, the colour changes from blue to orange/yellow.

The second and third task were all very similar to the procedure of the second task, where I had to discretise a function and use the Euler approximation to calculate the next approximation of data. For the third task when I create a movie the highest magnitude part of the graph at the initial conditions is at the center that I have declared in the *solve\_wave\_eq.m* file. Over time the center decreases in magnitude and the edges increase slightly in magnitude then decrease, the center becomes blue and the edges become yellow-ish then fade to blue. If I decrease the special step size the graph does not change that much. Changing the step size for the time intervals doesn’t change the process but how quick the movie plays. I did not attempt the frame rate part of the second task as I was not sure what to do.

For the third task I create a movie which only lasts for the first couple of time intervals. This is because when I compute the matrices *U* and *V* only the first couple of pages are filled with numbers then the matrices become filled with NaN values. I am not sure why this happens.