Assessment 1

Debugging, Coupled Harmonic Oscillator

Ben Sheppard

The main purpose of the code is to calculate eigenvalues using the QU factorisation with the Gram-Schmidt process.

Within the code there are 7 functions. The first two calculate the eigenvalues for 2x2 and 3x3 matrices, following QU factorisation. They are named “two_by_two” and three_by_three”. Both follow the same process of: 1) Accept a matrix. 2) Create a comparison array containing arbitrary numbers. 3) Enter a for loop (set for an arbitrary large number of loops) that runs the calculation. 4) Assign the accepted matrix to be the C matrix. 5) Calculate the f matrix. 6) Use the f matrix to calculate the q values and subsequently create a Q matrix. 7) Calculate a U matrix from the c and q values. 8) Cross multiply the U and Q matrices, producing the new A matrix. 9) Check the leading diagonal against the comparison array, if the values are the same (rounded to 7 decimal places) the for loop is broken. The leading diagonal will be returned from function as these are the eigenvalues. If they are different the leading diagonal is assigned as the new comparison matrix. 10) Assign the new A matrix as the C matrix. 11) Repeat steps 5-10. Unless the for loop ends, which means the values are not converging. An error will be raised informing the user of this, and the code will stop.

Before a matrix is assigned as the C matrix it is transposed. And all the calculations occur using transposed matrices. The Q and U matrices are transposed back to normal before they are cross multiplied. Steps up to the cross multiplication are using transposed matrices because I was struggling to make most of the calculations work, when the matrix was orientated regularly.

The next function, “two_by_two_check”, was created to check the eigenvalues of QU factorisation against those found using the characteristic equation. This was done by following the matrix elements of the initial matrix through to the quadratic equation. It is shown in more detail in the markdown cell above the function.

The next function, “det_check”, is for checking the determinant of the matrix, as if it is zero then the eigenvalues cannot be calculated using QU factorisation. If the determinant is 0 an error is raised stating so. This function gives the option to return the string “skip”, this is so that if you are calculating many matrices sequentially it doesn’t stop the whole code it just skips that matrix. It also checks to see of the matrix is a 2x2 or 3x3, and thus solvable using this code. Since the main point of the function is to stop the code if the determinant is 0 and stops the entire code in this situation. It only needs to return anything in the situation where you wish to skip the matrix.

The three remaining functions all calculate frequencies when supplied with masses and spring constants. One accepts two masses (equal or different) which increase by 1 each time; one accepts a single mass and make the second mass double it; the third accepts two masses but only changes (increases by 1 each time) one of the. All three call to check the determinant and then call to calculate the eigenvalues using the previously mentioned functions. They then return all then masses used in the calculations and frequencies calculated in arrays.

All the matrices in the code are made using np.array as this is what I know better, and I was struggling to us np.matrix. Pythons default data types are kept throughout the code, as the checks are to 7 decimal places. This should by well within the processing capacity of modern computers. Eigenvalues are returned from functions at 7 decimal places; however, they can gain more when square rooting them to get the frequency.

There is a “code check” section which calculates the eigenvalues for an arbitrary mass and spring constant. It is here the characteristic equation check is used. If the eigenvalues from QU factorisation doesn’t match those from the characteristic equation, an error is raised informing you so and the code stops. Currently the test matrix is $\begin{pmatrix} -\frac{2k}{m} & \frac{k}{m} \\ \frac{k}{m} & -\frac{2k}{m} \end{pmatrix}$, with k =5 and m=3. This produces eigenvalues of -5 and -5/3 using QU factorisation and the characteristic equation, both in code and checked manually as shown in fig 1.

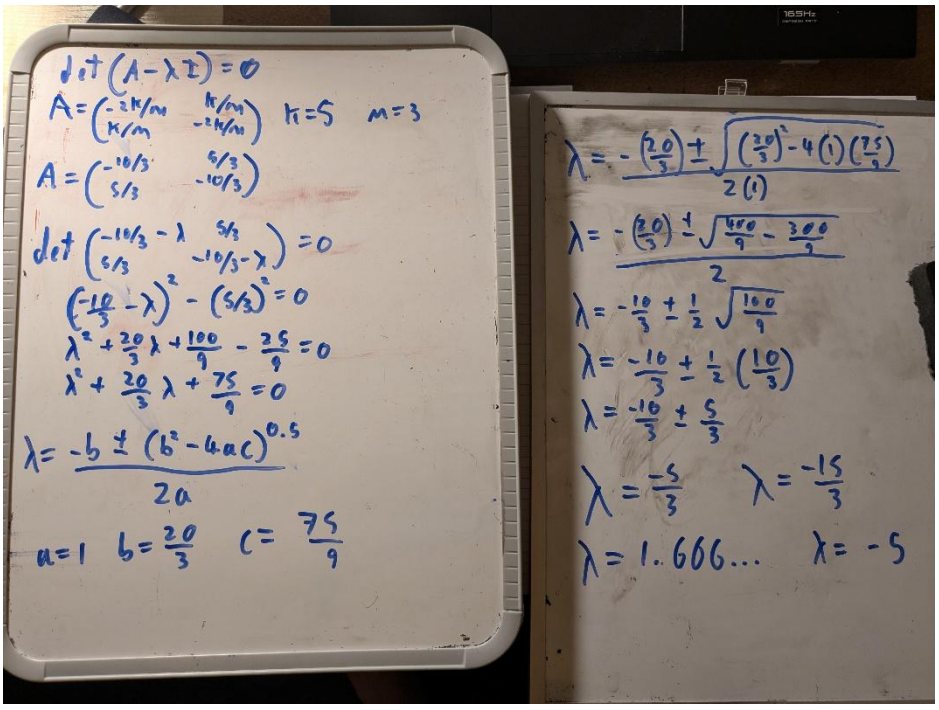

Figure 1) Solving of the test matrix by hand, using the characteristic equation.

There are then several sets of calculations and plots for frequencies dependent on the masses changing. There are both masses changing linearly, the code allows for the masses to be set separately so they can be different if it is desired. The calculation of frequency when the second mass is double the first mass. And when one mass is constant.

Figure 2 is the plot for the frequencies when both masses increase. They both increase at an equal rate and in this plot from the same initial mass of 1kg. Two different sets of frequency are produced by the system, one is where the masses are moving in phase, and the other is where they are out of phase. Frequency 1 is out of phase; this can be seen by the greater frequency. The greater frequency would require a greater restoring force, which is achievable by all three springs being extended and for this to happen the two masses must be moving separately. The lower frequency of in phase is

because of a weaker restoring force as only the outer two springs are affecting the system, the one connecting the masses doesn't extend as they are moving together. Thus, the system acts a single heavy mass. The decrease in frequency with the increase in mass is expected since $-m\omega^2 A_1 = -2kA_1 + kA_2$ has $\omega \propto \frac{1}{\sqrt{m}}$.

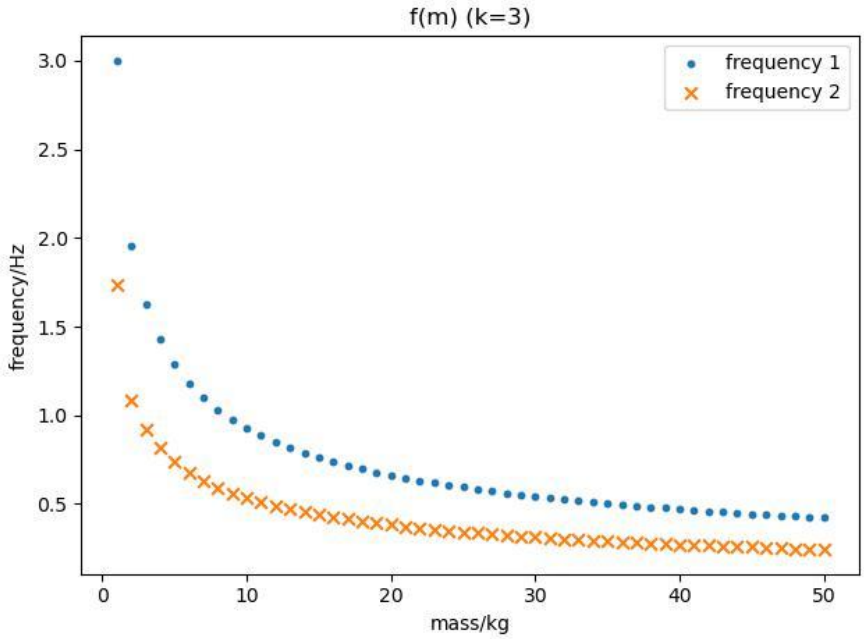

Figure 2) A plot of mass against frequency for a coupled oscillator system. The spring constant was 3 for its calculation. Both masses in the system increased linearly from a starting point of 1kg.

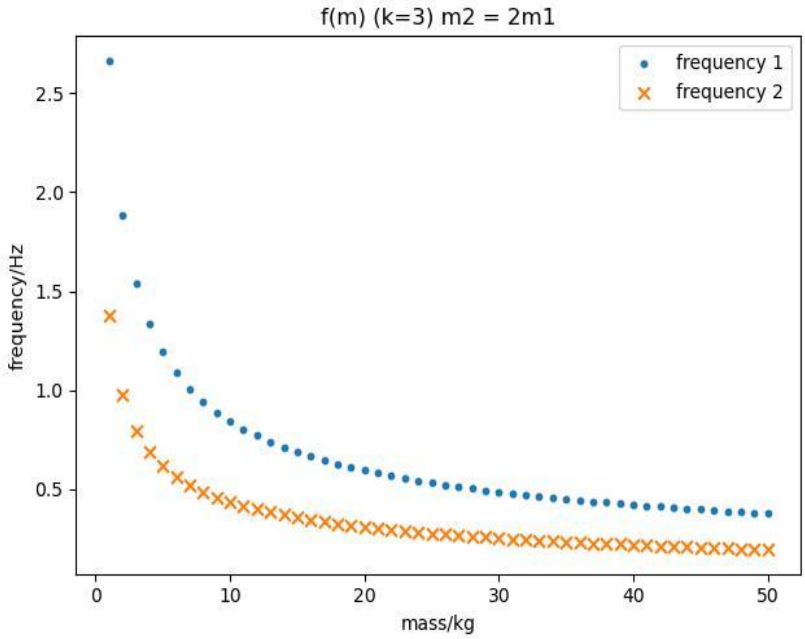

Figure 3) Mass against frequency, when one mass is twice as great as the other. The displayed mass is that of the lower mass.

Figure 3 shows the frequencies of the system against mass, when one mass is twice as great. The initial mass of the smaller was 1kg. The effect on the frequencies is the same as when they both increase by the same amount.

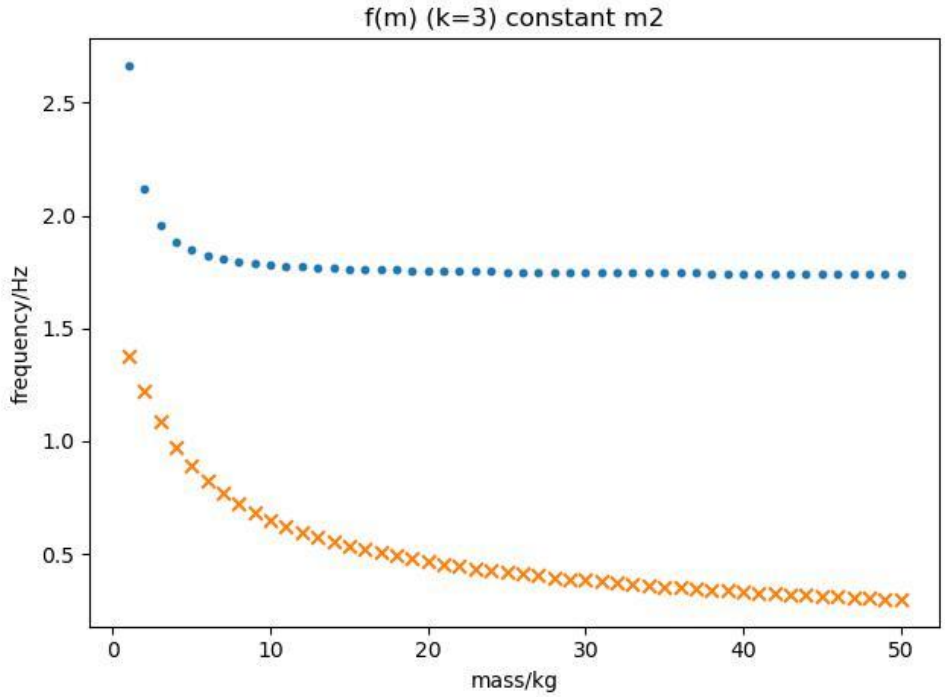

Figure 4) The oscillation frequency of a two-mass system where one mass is increasing, and the other constant. The initial masses were 1kg and 2kg, with the 2kg constant.

When one mass is constant the out of phase frequency flattens to near constant, this is very different to either of the other two systems which subtended towards zero. This is shown in figure 4. The near constant frequencies will be a result of the dominance of the greater mass, effectively causing the two masses to act as one mass and a stationary object. When this occurs the increase in the greater mass will make no difference as, making a "stationary" object heavier won't affect its movement. The in-phase frequency decreases as the other increasing mass systems have, in that it obeys, $\omega \propto \frac{1}{\sqrt{m}}$.