Coupled Oscillators - a worked example of the Eigenvalue and Eigenvector analysis.

Introduction

In the first three lectures of the Classical I: Mechanics and Thermal Physics course we considered the case of coupled oscillators. As you will have realised from the lectures, the full solution to these problems often entails tedious matrix calculations (finding determinants etc.) that are very prone to simple errors in calculation. In practise computers are veryy good at handling these linear algebra problems. The idea of this work book is that you should begin to appreciate the physics of the solutions to the problem and what they mean, without getting bogged down in calculation. This said, you should make sure that you are able to solve eigenvalue/eigenvector problems for simple 2 or 3 particle coupled systems.

Before we move on let's consider the general scheme for approaching these problems.

Firstly, from the description of the problem you should be able to recognise it as a question of coupled oscillators. Questions will almost always state that you should consider small oscillations, in which case you can assume the potential will be parabolic. You should be able to demonstrate this by consider a Taylor expansion of any sensible 'bowl' like potential. Typical examples might be springs, coupled pendulums, coupled LCR circuits or even the interatomic potentials between atoms.

For our typical mechanics questions it is the solution for the position of the masses as a function of time that we seek.(Coupled LCR circuits will be discussed separately).

We will first concentrate on the steps needed to solve the problem using the example given the lectures.

Step 1 - The formulation of the coupled oscillator equations

In order to solve this problem we need to apply Newtons second law of motion to each of the masses in turn. We should remember that we will specify the coordinate of each mass m_1, m_2 in terms of their coordinates x_1 and x_2 that are defined as zero when the masses are in their equilibrium positions. Hence we should obtain,

$$m_1\ddot{x}_1 = -k_1x_1 + k_2(x_2 - x_1)$$
$$m_2\ddot{x}_2 = -k_3x_2 + k_2(x_1 - x_2)$$

The trickiest part here is getting the signs correct. For the coupling terms (for example the term including k_2) setting either $x_1 = 0$ or $x_2 = 0$ is often sufficient to check the signs are correct. You should also note by Newton's third law that the force acting between the masses should be equal and opposite for each one. We can write these equations in matrix form as:

$$\begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix} \begin{bmatrix} \ddot{x}_1 \\ \ddot{x}_2 \end{bmatrix} = \begin{bmatrix} -(k_1 + k_2) & k_2 \\ k_2 & -(k_2 + k_3) \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

This is the equation we need to solve.

At this stage, as we did in lectures we will set $m_1 = m_2 = m$. We'll come back to what this means and how we deal with the case $m_1 \neq m_2$ later. In this case we get,

$$\begin{bmatrix} \ddot{x}_1 \\ \ddot{x}_2 \end{bmatrix} = \frac{1}{m}\begin{bmatrix} -(k_1 + k_2) & k_2 \\ k_2 & -(k_2 + k_3) \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

.

In contrast to the lectures, where we set $k_1 = k_2 = k_3$, let's make the middle spring stronger so that $k_2 = 2k_1 = 2k_3 = 2k$. In this case we have the equation,

$$\begin{bmatrix} \ddot{x}_1 \\ \ddot{x}_2 \end{bmatrix} = -\frac{k}{m}\begin{bmatrix} 3 & -2 \\ -2 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad \dots (1)$$

.

Step 2 - The trial solution

Formally we need to solve these coupled differential equations. However, the solutions are well known and closely related to the (uncoupled) simple harmonic oscillator equation where used trials solutions of the form $x = C\exp(i\omega t)$. In a similar way we will choose a trial solution for this problem of the form,

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} C_1 \\ C_2 \end{bmatrix} \exp(i\omega t) \quad \ldots (2)$$

There are a few points to note here.

We are no longer imagining the individual masses as moving with simple harmonic motion but we asscociate the oscillatory motion with a vector $\vec{x} = \vec{x}_0 \exp(i\omega t) = (C_1\hat{x}_1 + C_2\hat{x}_2)\exp(i\omega t)$ where $\hat{x}_1$ and $\hat{x}_2$ are unit vectors in our vector space. The idea of the vector space can be a difficult concept to understand especially when you realise that for N masses we have an N dimensional vector space (and N may be considerably larger than 3!).

We will also see that we are not looking for a single frequency but N different frequencies (although they may be degerate, i.e. two independent solutions have the same frequency.)

Hence, using the trial solution on (2) and subsituting into (1) we find

$$\omega^2 \begin{bmatrix} C_1 \\ C_2 \end{bmatrix} = -\frac{k}{m} \begin{bmatrix} 3 & -2 \\ -2 & 3 \end{bmatrix} \begin{bmatrix} C_1 \\ C_2 \end{bmatrix} \quad \ldots (3)$$

.

As we know k and m we need to solve this equation to find the values of ω and $\begin{bmatrix} C_1 \\ C_2 \end{bmatrix}$.

At this point we may also note that we multiply both sides of this equation by an arbitary constant, so we will not find unique values for C_1 or C_2 although the ratio between them must always be the same. Ultimately the value of these constants is determined when we substitute the initial conditions into our general solution. We can re-arrange (3) to find,

$$\left(-\frac{k}{m} \begin{bmatrix} 3 & -2 \\ -2 & 3 \end{bmatrix} - \omega^2 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) \begin{bmatrix} C_1 \\ C_2 \end{bmatrix} = 0 \quad \ldots (4)$$

.

that we see is in the form

$$(A - \lambda I)\,\vec{v} = 0$$

for which the solutions λ and $\vec{v}$ are the eigenvalues and eigenvectors of this equation.

Step 3 - Finding the Eigenvalues and Eigenvectors.

We may find, either the Eigenvalues in terms of k and m (or sometimes we might write $\sqrt{\frac{k}{m}} = \omega_0^2$) or we can enter their values. You should make sure that you can do this yourself for 2×2 matrices (where you will need to solve a quadratic equation to find the eigenvalues. It is also possible to find analytic (closed form) solutions for 3×3 and 4×4 problems but the solutions are difficult to remember and apply, so in this course you will only be asked to solve 2×2 problems or 3×3 problems where it is straightforward to determine the roots of the cubic equation you obtain. In general you cannot find analytic solutions for $N \geq 5$ although solutions do exist for special cases.

In this sheet we will use Python and the Linear Algebra package available with it, to determine the eigenvalues and eigenvectors for this problem. For convenience I will set $\sqrt{\frac{k}{m}} = 1$ but you should be able to modify the code to take into account different values of k and m if you wish

Hence, we now wish to find the eigenvalues and eigenvectors of the matrix,

$$\begin{bmatrix} 3 & -2 \\ -2 & 3 \end{bmatrix}$$

Here is the code we need to do the calculation,

In [1]:

```
### The first part imports the linear algebra library so it recognises the function nee
ded to calculate the eigenvalues and vectors
import numpy as NP
from scipy import linalg as LA
###
### Now we'll enter the array
###
A = NP.array([[3,-2],[-2,3]])
print (A)
e_vals,e_vecs = LA.eig(A)
print ("Eigenvalues")
print(e_vals)
print ("Eigenvectors")
print(e_vecs)
```

```
[[ 3 -2]
 [-2  3]]
Eigenvalues
[5.+0.j 1.+0.j]
Eigenvectors
[[ 0.70710678  0.70710678]
 [-0.70710678  0.70710678]]
```

The python language is not the easiest language to use when creating an manipluating matrices but hopefully you can see how this works. Also note that the eigenvalues are returned as complex numbers with the imaginary part as zero as we expect.

From the result we see from this case (slightly different to the lecture) we have two eigenvalues. $\omega^2 = 5$ and $\omega^2 = 1$.

The eigenvectors are return as a matrix of normalised column vectors so the first eigenvector is $\begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix}$ and the second eigenvector is $\begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$. With these results we can see the effect of the stronger spring in the middle. We note that the second eigenvector (eigenvalue = 1) is unchanged from that obtained in the lecture when the spring constants were all the same. In this mode the central spring is never extended or compressed so it should its strength should have no bearing on the result. The first eigenvector is the same as previously but the eigenvalue is higher which again is perhaps unsurprising as we have just increased the net strength of the springs pulling on the mass in this mode.

Hence, we'd expect our general solution to be a linear cominbation of these two modes i.e.

$$\begin{bmatrix} \ddot{x}_1 \\ \ddot{x}_2 \end{bmatrix} = A \begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix} \exp(i(5t)) + B \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} \exp(i(1t))$$

where A and B are constants to be determined from the initial conditions.

Note. As you might expect, the eigenvectors are unchanged from the case studied in lectures, but the eigenfrequencies are.

Exercise

The python code from the example above is copied below. Go back to Step 1 above and find the matrix when you enter different values of k_1, k_2 and k_3. Use the code to calculate the eigenvalues and eigenvectors and try to understand how the nature of the coupled oscillations has changed. Look in particular when $k_1 \neq k_2 \neq k_3$ for example, $k_1 = 1, k_2 = 2, k_3 = 3$. Try to interpret physically what is going on.

In [2]:

```
### The first part imports the linear algebra library so it recognises the function nee
ded to calculate the eigenvalues and vectors
import numpy as NP
from scipy import linalg as LA
###
### Enter values into the array below for the equations of motion you have determined.
###
A = NP.array([[3,-2],[-2,5]])
print (A)
e_vals,e_vecs = LA.eig(A)
print ("Eigenvalues")
print(e_vals)
print ("Eigenvectors")
print(e_vecs)
```

```
[[ 3 -2]
 [-2  5]]
Eigenvalues
[1.76393202+0.j 6.23606798+0.j]
Eigenvectors
[[-0.85065081  0.52573111]
 [-0.52573111 -0.85065081]]
```

Beyond 2 x 2 problems

Beyond 2×2 matrices it becomes timeconsuming and tedious to calculate eigenvalues and eigenvectors. However, the techniques for calculating them using computers are well established using 'Linear Algebra' packages. The code above using python is one example but you can also find libraries for e.g. fortran, C/C++, for computational software packages such as Matlab, Scilab and symbolic algebra packages such as Maple, Mathematica (Wolfram Alpha) and macsyma. Here we give an example of a more complex problem using python.

A note of caution. These programmes do not always return what you expect. I have found this an issue especially for the case where we have degeberate eigenvalues (two eigenvalues that have the same value) and their associated eigenvectors.

Let's extend our code to image a coupled spring system consistenting of 11 equal masses m and 12 springs, all with spring constant k. Demonstrate that you can obtain the equation

$$\begin{bmatrix} \ddot{x}_1 \\ \ddot{x}_2 \\ \ddot{x}_3 \\ \ddot{x}_4 \\ \ddot{x}_5 \\ \ddot{x}_6 \\ \ddot{x}_7 \\ \ddot{x}_8 \\ \ddot{x}_9 \\ \ddot{x}_{10} \\ \ddot{x}_{11} \end{bmatrix} = \frac{k}{m} \begin{bmatrix} 2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & -2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \\ x_{11} \end{bmatrix}$$

and use the code below to find the eigenvalues and eigenvectors.

You can also find the eigenvectors for this equation analytically (you can find an example in the set problems). It is an example of a sparse matrix (contains lots of zeros). Note that the eigenvectors are returned the columns in an 11×11 matrix. You should extract these and plot them out (x_i vs i). You may like to use this code and decrease or increase N to explore how the solutions change.

In [3]:

```
### The first part imports the linear algebra library so it recognises the function nee
ded to calculate the eigenvalues and vectors
import numpy as NP
from scipy import linalg as LA
A = NP.array([
              [2,-1,0,0,0,0,0,0,0,0,0],
              [-1,2,-1,0,0,0,0,0,0,0,0],
              [0,-1,2,-1,0,0,0,0,0,0,0],
              [0,0,-1,2,-1,0,0,0,0,0,0],
              [0,0,0,-1,2,-1,0,0,0,0,0],
              [0,0,0,0,-1,2,-1,0,0,0,0],
              [0,0,0,0,0,-1,2,-1,0,0,0],
              [0,0,0,0,0,0,-1,2,-1,0,0],
              [0,0,0,0,0,0,0,-1,2,-1,0],
              [0,0,0,0,0,0,0,0,-1,2,-1],
              [0,0,0,0,0,0,0,0,0,-1,2],
             ])
print (A)
e_vals,e_vecs = LA.eig(A)
print ("Eigenvalues")
print(e_vals)
print ("Eigenvectors")
print(e_vecs)
```

```
[[ 2 -1  0  0  0  0  0  0  0  0]
 [-1  2 -1  0  0  0  0  0  0  0]
 [ 0 -1  2 -1  0  0  0  0  0  0]
 [ 0  0 -1  2 -1  0  0  0  0  0]
 [ 0  0  0 -1  2 -1  0  0  0  0]
 [ 0  0  0  0 -1  2 -1  0  0  0]
 [ 0  0  0  0  0 -1  2 -1  0  0]
 [ 0  0  0  0  0  0 -1  2 -1  0]
 [ 0  0  0  0  0  0  0 -1  2 -1]
 [ 0  0  0  0  0  0  0  0 -1  2]]
Eigenvalues
[3.93185165+0.j 3.73205081+0.j 3.41421356+0.j 3.        +0.j
 2.51763809+0.j 2.        +0.j 1.48236191+0.j 0.06814835+0.j
 0.26794919+0.j 0.58578644+0.j 1.        +0.j]
Eigenvectors
[[-1.05662433e-01 -2.04124145e-01 -2.88675135e-01  3.53553391e-01
   3.94337567e-01 -4.08248290e-01 -3.94337567e-01  1.05662433e-01
  -2.04124145e-01 -2.88675135e-01 -3.53553391e-01]
 [ 2.04124145e-01  3.53553391e-01  4.08248290e-01 -3.53553391e-01
  -2.04124145e-01  6.80885866e-16 -2.04124145e-01  2.04124145e-01
  -3.53553391e-01 -4.08248290e-01 -3.53553391e-01]
 [-2.88675135e-01 -4.08248290e-01 -2.88675135e-01  2.27812574e-16
  -2.88675135e-01  4.08248290e-01  2.88675135e-01  2.88675135e-01
  -4.08248290e-01 -2.88675135e-01 -1.16264558e-16]
 [ 3.53553391e-01  3.53553391e-01 -1.15842804e-15  3.53553391e-01
   3.53553391e-01 -7.62137297e-16  3.53553391e-01  3.53553391e-01
  -3.53553391e-01  3.59976775e-16  3.53553391e-01]
 [-3.94337567e-01 -2.04124145e-01  2.88675135e-01 -3.53553391e-01
   1.05662433e-01 -4.08248290e-01 -1.05662433e-01  3.94337567e-01
  -2.04124145e-01  2.88675135e-01  3.53553391e-01]
 [ 4.08248290e-01  3.47517159e-15 -4.08248290e-01 -8.76752927e-16
  -4.08248290e-01  1.20152971e-15 -4.08248290e-01  4.08248290e-01
  -1.68123499e-16  4.08248290e-01  3.71041776e-16]
 [-3.94337567e-01  2.04124145e-01  2.88675135e-01  3.53553391e-01
   1.05662433e-01  4.08248290e-01 -1.05662433e-01  3.94337567e-01
   2.04124145e-01  2.88675135e-01 -3.53553391e-01]
 [ 3.53553391e-01 -3.53553391e-01 -6.96321835e-16 -3.53553391e-01
   3.53553391e-01 -6.41082192e-16  3.53553391e-01  3.53553391e-01
   3.53553391e-01  2.95845970e-16 -3.53553391e-01]
 [-2.88675135e-01  4.08248290e-01 -2.88675135e-01 -1.65782792e-15
  -2.88675135e-01 -4.08248290e-01  2.88675135e-01  2.88675135e-01
   4.08248290e-01 -2.88675135e-01 -5.66917868e-16]
 [ 2.04124145e-01 -3.53553391e-01  4.08248290e-01  3.53553391e-01
  -2.04124145e-01  3.01619793e-16 -2.04124145e-01  2.04124145e-01
   3.53553391e-01 -4.08248290e-01  3.53553391e-01]
 [-1.05662433e-01  2.04124145e-01 -2.88675135e-01 -3.53553391e-01
   3.94337567e-01  4.08248290e-01 -3.94337567e-01  1.05662433e-01
   2.04124145e-01 -2.88675135e-01  3.53553391e-01]]
```

What happens when we remove the walls?

In the previous examples the masses were attached to fixed walls by springs (Dirichlet boundary conditions). Modify the code above (reproduced below) to consider the case when the masses at the ends are not connected to a wall but are allowed to float freely. To do this you need to redetermine the equations of motion for the two masses at the end of the chain and modify the first and last rows of the matrix accordingly. Again plot out the eigenvectors. One of the eigenvectors is distinct from the others. Can you explain why it has this form?

In [4]:

```
### The first part imports the linear algebra library so it recognises the function nee
ded to calculate the eigenvalues and vectors
import numpy as NP
from scipy import linalg as LA
A = NP.array([
              [2,-1,0,0,0,0,0,0,0,0,0],
              [-1,2,-1,0,0,0,0,0,0,0,0],
              [0,-1,2,-1,0,0,0,0,0,0,0],
              [0,0,-1,2,-1,0,0,0,0,0,0],
              [0,0,0,-1,2,-1,0,0,0,0,0],
              [0,0,0,0,-1,2,-1,0,0,0,0],
              [0,0,0,0,0,-1,2,-1,0,0,0],
              [0,0,0,0,0,0,-1,2,-1,0,0],
              [0,0,0,0,0,0,0,-1,2,-1,0],
              [0,0,0,0,0,0,0,0,-1,2,-1],
              [0,0,0,0,0,0,0,0,0,-1,2],
             ])
print (A)
e_vals,e_vecs = LA.eig(A)
print ("Eigenvalues")
print(e_vals)
print ("Eigenvectors")
print(e_vecs)
```

```
[[ 2 -1  0  0  0  0  0  0  0  0]
 [-1  2 -1  0  0  0  0  0  0  0]
 [ 0 -1  2 -1  0  0  0  0  0  0]
 [ 0  0 -1  2 -1  0  0  0  0  0]
 [ 0  0  0 -1  2 -1  0  0  0  0]
 [ 0  0  0  0 -1  2 -1  0  0  0]
 [ 0  0  0  0  0 -1  2 -1  0  0]
 [ 0  0  0  0  0  0 -1  2 -1  0]
 [ 0  0  0  0  0  0  0 -1  2 -1]
 [ 0  0  0  0  0  0  0  0 -1  2]]
Eigenvalues
[3.93185165+0.j 3.73205081+0.j 3.41421356+0.j 3.        +0.j
 2.51763809+0.j 2.        +0.j 1.48236191+0.j 0.06814835+0.j
 0.26794919+0.j 0.58578644+0.j 1.        +0.j]
Eigenvectors
[[-1.05662433e-01 -2.04124145e-01 -2.88675135e-01  3.53553391e-01
   3.94337567e-01 -4.08248290e-01 -3.94337567e-01  1.05662433e-01
  -2.04124145e-01 -2.88675135e-01 -3.53553391e-01]
 [ 2.04124145e-01  3.53553391e-01  4.08248290e-01 -3.53553391e-01
  -2.04124145e-01  6.80885866e-16 -2.04124145e-01  2.04124145e-01
  -3.53553391e-01 -4.08248290e-01 -3.53553391e-01]
 [-2.88675135e-01 -4.08248290e-01 -2.88675135e-01  2.27812574e-16
  -2.88675135e-01  4.08248290e-01  2.88675135e-01  2.88675135e-01
  -4.08248290e-01 -2.88675135e-01 -1.16264558e-16]
 [ 3.53553391e-01  3.53553391e-01 -1.15842804e-15  3.53553391e-01
   3.53553391e-01 -7.62137297e-16  3.53553391e-01  3.53553391e-01
  -3.53553391e-01  3.59976775e-16  3.53553391e-01]
 [-3.94337567e-01 -2.04124145e-01  2.88675135e-01 -3.53553391e-01
   1.05662433e-01 -4.08248290e-01 -1.05662433e-01  3.94337567e-01
  -2.04124145e-01  2.88675135e-01  3.53553391e-01]
 [ 4.08248290e-01  3.47517159e-15 -4.08248290e-01 -8.76752927e-16
  -4.08248290e-01  1.20152971e-15 -4.08248290e-01  4.08248290e-01
  -1.68123499e-16  4.08248290e-01  3.71041776e-16]
 [-3.94337567e-01  2.04124145e-01  2.88675135e-01  3.53553391e-01
   1.05662433e-01  4.08248290e-01 -1.05662433e-01  3.94337567e-01
   2.04124145e-01  2.88675135e-01 -3.53553391e-01]
 [ 3.53553391e-01 -3.53553391e-01 -6.96321835e-16 -3.53553391e-01
   3.53553391e-01 -6.41082192e-16  3.53553391e-01  3.53553391e-01
   3.53553391e-01  2.95845970e-16 -3.53553391e-01]
 [-2.88675135e-01  4.08248290e-01 -2.88675135e-01 -1.65782792e-15
  -2.88675135e-01 -4.08248290e-01  2.88675135e-01  2.88675135e-01
   4.08248290e-01 -2.88675135e-01 -5.66917868e-16]
 [ 2.04124145e-01 -3.53553391e-01  4.08248290e-01  3.53553391e-01
  -2.04124145e-01  3.01619793e-16 -2.04124145e-01  2.04124145e-01
   3.53553391e-01 -4.08248290e-01  3.53553391e-01]
 [-1.05662433e-01  2.04124145e-01 -2.88675135e-01 -3.53553391e-01
   3.94337567e-01  4.08248290e-01 -3.94337567e-01  1.05662433e-01
   2.04124145e-01 -2.88675135e-01  3.53553391e-01]]
```

Conclusion for this notebook

In this notebook we have explored the two mass coupled oscillator system covered in lectures and extended the solution to cases where the spring constants are unequal. The case where the masses are different is covered in a different notebook. We have shown how we can treat this as an eigenvalue/eigenvector problem and shown how the soluitions can be calculated using python. We have extended the analysis to chains with $N >> 2$ and shown how the solutions may still be calculated with simple function calls in python.