Lab 3

The Wave Equation: Steady State and Resonance

To see why we did so much work in Lab 2 on ordinary differential equations when this is a course on partial differential equations, let's look at the wave equation in one dimension. For a string of length L fixed at both ends with a force applied to it that varies sinusoidally in time, the wave equation can be written as

$$\mu \frac{\partial^2 y}{\partial t^2} = T \frac{\partial^2 y}{\partial x^2} + f(x)\cos\omega t \quad ; \quad y(0,t) = 0, \quad y(L,t) = 0 \tag{3.1}$$

where $y(x,t)$ is the (small) sideways displacement of the string as a function of position and time, assuming that $y(x,t) \ll L$. [1] This equation may look a little unfamiliar to you, so let's discuss each term. We have written it in the form of Newton's second law, $F = ma$. The "ma" part is on the left of Eq. (3.1), except that μ is not the mass, but rather the linear mass density (mass/length). This means that the right side should have units of force/length, and it does because T is the tension (force) in the string and $\partial^2 y/\partial x^2$ has units of 1/length. (Take a minute and verify that this is true.) Finally, $f(x)$ is the amplitude of the driving force (in units of force/length) applied to the string as a function of position (so we are not necessarily just wiggling the end of the string) and ω is the frequency of the driving force.

Before we start calculating, let's train our intuition to guess how the solutions of this equation behave. If we suddenly started to push and pull on a string under tension with force density $f(x)\cos(\omega t)$, we would launch waves, which would reflect back and forth on the string as the driving force continued to launch more waves. The string motion would rapidly become very messy. Now suppose that there was a little bit of damping in the system (not included in the equation above, but in Lab 5 we will add it). Then what would happen is that all of the transient waves due to the initial launch and subsequent reflections would die away and we would be left with a steady-state oscillation of the string at the driving frequency ω. (This behavior is the wave equation analog of damped transients and the steady final state of a driven harmonic oscillator.)

Steady state solution

Let's look for this steady-state solution by guessing that the solution has the form

$$y(x,t) = g(x)\cos(\omega t) \tag{3.2}$$

[1] N. Asmar, *Partial Differential Equations and Boundary Value Problems* (Prentice Hall, New Jersey, 2000), p. 87-110.

This function has the expected form of a spatially dependent amplitude which oscillates at the frequency of the driving force. Substituting this "guess" into the wave equation to see if it works yields (after some rearrangement)

$$Tg''(x) + \mu\omega^2 g(x) = -f(x) \quad ; \quad g(0) = 0, \quad g(L) = 0 \tag{3.3}$$

This is just a two-point boundary value problem of the kind we studied in Lab 2, so we can solve it using our matrix technique.

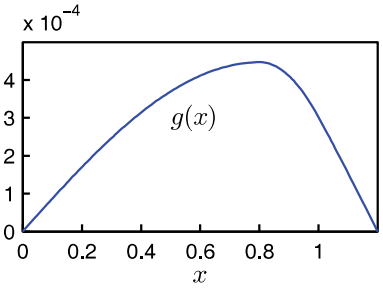

Figure 3.1 Solution to 3.1(a)

P3.1 (a) Modify one of your Python scripts from Lab 2 to solve Eq. (3.3) with $\mu = 0.003$, $T = 127$, $L = 1.2$, and $\omega = 400$. (All quantities are in SI units.) Find the steady-state amplitude associated with the driving force density:

$$f(x) = \begin{cases} 0.73 & \text{if } 0.8 \le x \le 1 \\ \\ 0 & \text{if } x < 0.8 \text{ or } x > 1 \end{cases} \tag{3.4}$$

(b) Repeat the calculation in part (a) for 100 different frequencies between $\omega = 400$ and $\omega = 1200$ by putting a loop around your calculation in (a) that varies ω. Use this loop to load the maximum amplitude as a function of ω and plot it to see the resonance behavior of this system. Can you account qualitatively for the changes you see in $g(x)$ as ω varies? (Use a pause command after the plots of $g(x)$ and watch what happens as ω changes. Using `pause(.3)` will make an animation and using `ylim([0 0.03])` will prevent Python's autoscaling of plots from making all of the string responses from looking like they have the same amplitude.)

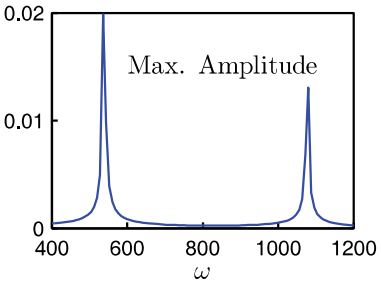

Figure 3.2 Solution to problem 3.1(b).

In problem 3.1(b) you should have noticed an apparent resonance behavior, with resonant frequencies near $\omega = 550$ and $\omega = 1100$ (see Fig. 3.2). Now we will learn how to use Python to find these resonant frequencies directly (i.e. without solving the differential equation over and over again).

Resonance and the eigenvalue problem

The essence of resonance is that at certain frequencies a large steady-state amplitude is obtained with a very small driving force. To find these resonant frequencies we seek solutions of Eq. (3.3) for which the driving force is zero. With $f(x) = 0$, Eq. (3.3) takes on the form

$$-\mu\omega^2 g(x) = Tg''(x) \quad ; \quad g(0) = 0, \quad g(L) = 0 \tag{3.5}$$

If we rewrite this equation in the form

$$g''(x) = -\left(\frac{\mu\omega^2}{T}\right) g(x) \tag{3.6}$$

then we see that it is in the form of a classic eigenvalue problem:

$$Ag = \lambda g \tag{3.7}$$

where A is a linear operator (the second derivative on the left side of Eq. (3.6)) and λ is the eigenvalue ($-\mu\omega^2/T$ in Eq. (3.6).)

Equation (3.6) is easily solved analytically, and its solutions are just the familiar sine and cosine functions. The condition $g(0) = 0$ tells us to try a sine function form, $g(x) = g_0 \sin(kx)$. To see if this form works we substitute it into Eq. (3.6 and find that it does indeed work, provided that the constant k is $k = \omega\sqrt{\mu/T}$. We have, then,

$$g(x) = g_0 \sin\left(\omega\sqrt{\frac{\mu}{T}}x\right) \tag{3.8}$$

where g_0 is the arbitrary amplitude. But we still have one more condition to satisfy: $g(L) = 0$. This boundary condition tells us the values that resonance frequency ω can take on. When we apply the boundary condition, we find that the resonant frequencies of the string are given by

$$\omega = n\frac{\pi}{L}\sqrt{\frac{T}{\mu}} \tag{3.9}$$

where n is an integer. Each value of n gives a specific resonance frequency from Eq. (3.9) and a corresponding spatial amplitude $g(x)$ given by Eq. (3.8). Figure 3.3 shows photographs of a string vibrating for $n = 1, 2, 3$.

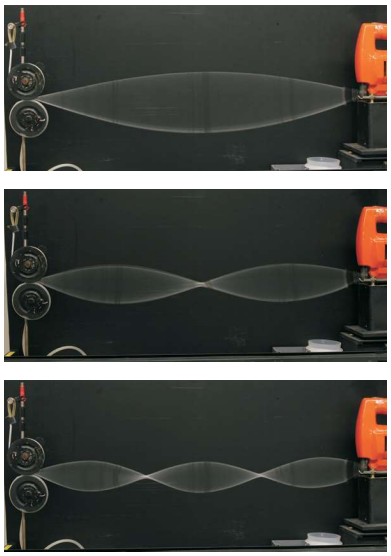

Figure 3.3 Photographs of the first three resonant modes for a string fixed at both ends.

For this simple example we were able to do the eigenvalue problem analytically without much trouble. However, when the differential equation is not so simple we will need to do the eigenvalue calculation numerically, so let's see how it works in this simple case. Rewriting Eq. (3.5) in matrix form, as we learned to do by finite differencing the second derivative, yields

$$\mathbf{Ag} = \lambda\mathbf{g} \tag{3.10}$$

which is written out as

$$\begin{bmatrix} ? & ? & ? & ? & \dots & ? & ? & ? \\ \frac{1}{h^2} & -\frac{2}{h^2} & \frac{1}{h^2} & 0 & \dots & 0 & 0 & 0 \\ 0 & \frac{1}{h^2} & -\frac{2}{h^2} & \frac{1}{h^2} & \dots & 0 & 0 & 0 \\ . & . & . & . & \dots & . & . & . \\ . & . & . & . & \dots & . & . & . \\ . & . & . & . & \dots & . & . & . \\ 0 & 0 & 0 & 0 & \dots & \frac{1}{h^2} & -\frac{2}{h^2} & \frac{1}{h^2} \\ ? & ? & ? & ? & \dots & ? & ? & ? \end{bmatrix} \begin{bmatrix} g_1 \\ g_2 \\ g_3 \\ . \\ . \\ . \\ g_{N-1} \\ g_N \end{bmatrix} = \lambda \begin{bmatrix} ? \\ g_2 \\ g_3 \\ . \\ . \\ . \\ g_{N-1} \\ ? \end{bmatrix} \tag{3.11}$$

where $\lambda = -\omega^2\frac{\mu}{T}$. The question marks in the first and last rows remind us that we have to invent something to put in these rows that will implement the correct

boundary conditions. Note that having question marks in the g-vector on the right is a real problem because without g_1 and g_N in the top and bottom positions, we don't have an eigenvalue problem (i.e. the vector **g** on left side of Eq. (3.11) is not the same as the vector **g** on the right side).

The simplest way to deal with this question-mark problem and to also handle the boundary conditions is to change the form of Eq. (3.7) to the slightly more complicated form of a *generalized eigenvalue problem*, like this:

$$\mathbf{Ag} = \lambda \mathbf{Bg} \tag{3.12}$$

where **B** is another matrix, whose elements we will choose to make the boundary conditions come out right. To see how this is done, here is the generalized modification of Eq. (3.11) with **B** and the top and bottom rows of **A** chosen to apply the boundary conditions $g(0) = 0$ and $g(L) = 0$.

$$\begin{array}{ccccccccccc} & & & \mathbf{A} & & & & \mathbf{g} & = \lambda & \mathbf{B} & & & \mathbf{g} \\ \begin{bmatrix} 1 & 0 & 0 & \dots & 0 & 0 \\ \frac{1}{h^2} & -\frac{2}{h^2} & \frac{1}{h^2} & \dots & 0 & 0 \\ 0 & \frac{1}{h^2} & -\frac{2}{h^2} & \dots & 0 & 0 \\ \cdot & \cdot & \cdot & \dots & \cdot & \cdot \\ \cdot & \cdot & \cdot & \dots & \cdot & \cdot \\ \cdot & \cdot & \cdot & \dots & \cdot & \cdot \\ 0 & 0 & 0 & \dots & -\frac{2}{h^2} & \frac{1}{h^2} \\ 0 & 0 & 0 & \dots & 0 & 1 \end{bmatrix} & \begin{bmatrix} g_1 \\ g_2 \\ g_3 \\ \cdot \\ \cdot \\ \cdot \\ g_{N-1} \\ g_N \end{bmatrix} & = \lambda & \begin{bmatrix} 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & \dots & 0 & 0 \\ \cdot & \cdot & \cdot & \dots & \cdot & \cdot \\ \cdot & \cdot & \cdot & \dots & \cdot & \cdot \\ \cdot & \cdot & \cdot & \dots & \cdot & \cdot \\ 0 & 0 & 0 & \dots & 1 & 0 \\ 0 & 0 & 0 & \dots & 0 & 0 \end{bmatrix} & \begin{bmatrix} g_1 \\ g_2 \\ g_3 \\ \cdot \\ \cdot \\ \cdot \\ g_{N-1} \\ g_N \end{bmatrix} \end{array} \tag{3.13}$$

Notice that the matrix **B** is very simple: it is just the identity matrix (made in Python with `eye(N,N)`) except that the first and last rows are completely filled with zeros. Take a minute now and do the matrix multiplications corresponding the first and last rows and verify that they correctly give $g_1 = 0$ and $g_N = 0$, no matter what the eigenvalue λ turns out to be.

To numerically solve this eigenvalue problem you simply do the following in Python:

(i) Load the matrix **A** with the matrix on the left side of Eq. (3.13) and the matrix **B** with the matrix on the right side.

(ii) Use Python's generalized eigenvalue and eigenvector command:

```
[V,D]=eig(A,B);
```

which returns the eigenvalues as the diagonal entries of the square matrix `D` and the eigenvectors as the columns of the square matrix `V` (these column arrays are the amplitude functions $g_j = g(x_j)$ associated with each eigenvalue on the grid x_j.)

(iii) Convert eigenvalues to frequencies via $\omega^2 = -\frac{T}{\mu}\lambda$, sort the squared frequencies in ascending order, and plot each eigenvector with its associated frequency displayed in the plot title.

This is such a common calculation that we will give you a section of a Matlab script below that does steps (ii) and (iii). You can get this code snippet on the Physics 430 web site so you don't have to retype it.

Listing 3.1 (eigen.m)

```
[V,D]=eig(A,B);  % find the eigenvectors and eigenvalues

w2raw=-(T/mu)*diag(D);  % convert lambda to omega^2

[w2,k]=sort(w2raw);  % sort omega^2 into ascending along with a
                     % sort key k(n) that remembers where each
                     % omega^2 came from so we can plot the proper
                     % eigenvector in V

for n=1:N  % run through the sorted list and plot each eigenvector
   % load the plot title into t
   t=sprintf(' w^2 = %g w = %g ',w2(n),sqrt(abs(w2(n))) );
   gn=V(:,k(n)); % extract the eigenvector
   plot(x,gn,'b-');  % plot the eigenvector that goes with omega^2
   title(t);xlabel('x');ylabel('g(n,x)');  % label the graph
   pause
end
```

P3.2 (a) Use Matlab to numerically find the eigenvalues and eigenvectors of Eq. (3.5) using the procedure outlined above. Use $\mu = 0.003$, $T = 127$, and $L = 1.2$. Note that there is a pause command in the code, so you'll need to hit a key to step to the next eigenvector. When you plot the eigenvectors, you will see that two infinite eigenvalues appear together with odd-looking eigenvectors that don't satisfy the boundary conditions. These two show up because of the two rows of the **B** matrix that are filled with zeros. They are numerical artifacts with no physical meaning, so just ignore them. You will also see that the eigenvectors of the higher modes start looking jagged. These must also be ignored because they are poor approximations to the continuous differential equation in Eq. (3.5).

(b) A few of the smooth eigenfunctions are very good approximations. Plot the eigenfunctions corresponding to $n = 1, 2, 3$ and compare them with the exact solutions in Eq. (3.8). Calculate the exact values for ω using Eq. (3.9) and compare them with the numerical eigenvalues. Now compare your numerical eigenvalues for the $n = 20$ mode with the exact solution. What is the trend in the accuracy of the eigenvalue method?

(c) The first two values for ω should match the resonances that you found in 3.1(b). Go back to your calculation in 3.1(b) and make two plots

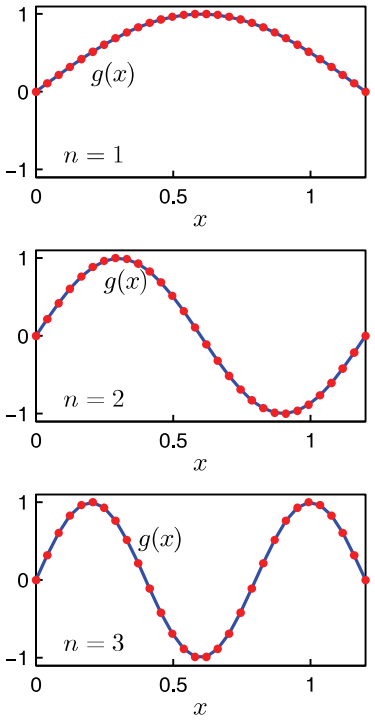

Figure 3.4 The first three eigenfunctions found in 3.2. The points are the numerical eigenfunctions and the line is the exact solution.

of the steady state amplitude for driving frequencies near these two resonant values of ω. (For each plot, choose a small range of frequencies that brackets the resonance frequency above and below). You should find very large amplitudes, indicating that you are right on the resonances.

Finally let's explore what happens to the eigenmode shapes when we change the boundary conditions.

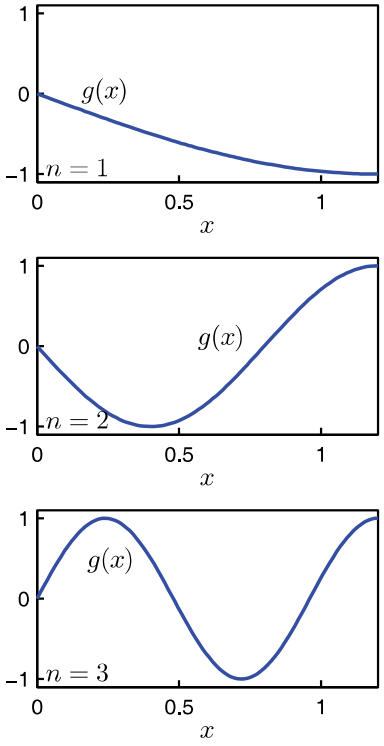

Figure 3.5 The first three eigenfunctions for 3.3(a).

P3.3 (a) Change your program from problem 3.2 to implement the boundary condition

$$g'(L) = 0$$

Use the approximation you derived in problem 2.4(b) for the derivative $g'(L)$ to implement this boundary condition, i.e.

$$g'(L) \approx \frac{1}{2h} g_{N-2} - \frac{2}{h} g_{N-1} + \frac{3}{2h} g_N$$

Explain physically why the resonant frequencies change as they do.

(b) In some problems mixed boundary conditions are encountered, for example

$$g'(L) = 2g(L)$$

Find the first few resonant frequencies and eigenfunctions for this case. Look at your eigenfunctions and verify that the boundary condition is satisfied. Also notice that one of your eigenvalues corresponds to ω^2 being negative. This means that this nice smooth eigenfunction is actually unphysical in our wave resonance problem with this boundary condition. The `sqrt(abs(w2(n)))` command in the code snippet we gave you misleads you in this case–be careful.