Eigenvalue Analysis

Daniel Underwood

July 21, 2018

Vibrating String

Many physical system can be represented as a set of masses on a string. This type of system can easily be modeled by differential equations using Newton's Second Law, resulting in the following system of differential equations:

$$m_i \frac{d^2 x_i}{dt^2} = \frac{F}{h}\left(x_{i-1} + x_{i+1} - 2x_i\right) \tag{1}$$

where $i = 1, ..., n$ for n masses, F is the constant horizontal tension on the string, h is the horizontal separation of each of the masses, and x_i is the vertical displacement for the i^{th} mass. In addition to the given values of i, eq. (1) also indicates the displacements x_0 and x_{n+1}. These terms would represent the masses attached on the outsides of masses $i = 1$ and $i = n + 1$. These cannot have a displacement, as they are not masses in the system; therefore $x_0 = x_{n+1} = 0$.

This system of differential equations can be represented by the matrix differential equation

$$\frac{d^2 \boldsymbol{x}}{dt^2} = -DA\boldsymbol{x} \tag{2}$$

where $\boldsymbol{x}$ is the column vector consisting of the vertical diaplacements, $[x_1, ..., x_n]^T$, and A and D are $n \times n$ matrices defined as

$$A := \begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & -1 & 2 & -1 & \\ & & \ddots & \ddots & \ddots \\ & & & -1 & 2 \end{bmatrix}$$

$$D := \frac{F}{h}\begin{bmatrix} m_1^{-1} & & & \\ & m_2^{-1} & & \\ & & \ddots & \\ & & & m_n^{-1} \end{bmatrix}$$

Equation (2) can easily be solved by assuming a solution of the form

$$\boldsymbol{x}(t) = e^{\lambda t}\boldsymbol{v} \tag{3}$$

Differentiating the assumption in eq. (3) results in

$$\ddot{\boldsymbol{x}} = \lambda^2 e^{\lambda t}\boldsymbol{v} = \lambda^2 \boldsymbol{x} \tag{4}$$

Combining eq. (2) with the derivative relation in eq. (4) results in

$$\lambda^2 \boldsymbol{x} = -DA\boldsymbol{x} \tag{5}$$

Equation (5) is the same as

$$DA\boldsymbol{v} = -\lambda^2 \boldsymbol{v} \tag{6}$$

by noticing that the $e^{\lambda t}$ terms cancel and the negative can be moved to λ in order to avoid possibly iterating through a matrix and multiplying every element by -1.

Solving the eigenvalue in eq. (6) results in eigenvalues $\mu = -\lambda^2$ of DA. Relating this with eqs. (4)–(5) results in fundamental frequencies $\lambda = i\sqrt{\mu}$ with fundamental modes $\boldsymbol{v}$ being the eigenvectors of DA.

Code to set up this calculation with a given set of masses, force, and positioning between masses is listed in Appendix A

Superposition Principle with Matrix Differential Equations

The general solution for a linear differential system can be given by

$$\boldsymbol{x}(t) = \sum_{i=1}^{n} \gamma_i e^{\lambda_i t}\boldsymbol{v}_i \tag{7}$$

Equation (7) can be expanded in terms of sines and cosines using the Euler formula, resulting in

$$\boldsymbol{x}(t) = \sum_{i=1}^{n}\left(\alpha_i \cos\left(\sqrt{\mu_i}t\right) + \beta_i \sin\left(\sqrt{\mu_i}t\right)\right)\boldsymbol{v}_i \tag{8}$$

where $\lambda = -i\sqrt{\mu}$ and the coefficients α_i and β_i are determinted by initial conditions

$$\boldsymbol{x}(0) = \boldsymbol{x}_0 \tag{9a}$$
$$\dot{\boldsymbol{x}}(0) = \boldsymbol{z}_0 \tag{9b}$$

Luckily, cosine and sine are easily evaluated at 0, resulting in the following relations for eqs. (9a)–(9b)

$$\boldsymbol{x}(0) = \sum_{i=1}^{n} \alpha_i \boldsymbol{v}_i \tag{10a}$$
$$\dot{\boldsymbol{x}}(0) = \sum_{i=1}^{n} \beta_i \sqrt{\mu_i} \boldsymbol{v}_i \tag{10b}$$

By taking the eigenvectors $\{\boldsymbol{v}_i\}$ to be column vectors in a matrix V, we have

$$V := \begin{bmatrix} v_1^1 & v_2^1 & \cdots & v_n^1 \\ \vdots & \vdots & & \vdots \\ v_1^m & v_2^m & \cdots & v_n^m \end{bmatrix}$$

i.e., $V_i^j = v_i^j$ where v_i^j is the j^{th} component of the i^{th} eigenvector or $V := [\boldsymbol{v}_1 \ldots \boldsymbol{v}_n]$. Taking $\boldsymbol{\alpha} = [\alpha_1 \ldots \alpha_n]^T$ and $\boldsymbol{b} = \left[\beta_1\sqrt{\mu_1} \ldots \beta_n\sqrt{\mu_n}\right]^T$, we are able to rewrite eqs. (10a)–(10b) as the following matrix equations

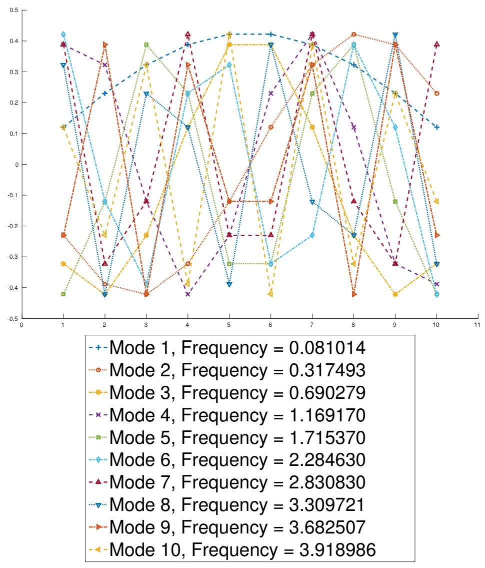
$$V\boldsymbol{\alpha} = \boldsymbol{x}_0 \tag{11a}$$
$$V\boldsymbol{b} = \boldsymbol{z}_0 \tag{11b}$$

Using matrix techniques, the solution vectors α and $\boldsymbol{b}$ may be obtained. Using the definitions of these solution vectors, we hhave the coefficients α_i as the i^{th} component of $\boldsymbol{\alpha}$ and $\beta_i = \frac{b_i}{\sqrt{\mu_i}}$.

Inverse Eigenvalue Problem

In addition to the regular eigenvalue problem discussed in the first section, there is also the inverse eigenvalue problem. The general inverse eigenvalue problem is constructing the matrix that will have a desired set of eigenvalues; in the context of the string eigenvalue problem, it is finding the masses that the string should be made out of to have a certain set of frequencies. While the inverse eigenvalue problem is difficult to solve and is an active area of mathematical research, we will explore the idea by analyzing the effect of making a string out of different patterns of masses. In addition to examining the configuration of masses on the string, one may examing the effects of the values of F or h; however, upon simple experimentation, it is found that they are not very interesting. Changing the values of h leaves the modes the same while reducing or enlarging the entire length of the string due to the length of each segment being changed. This could be useful if one wanted to make a string of a certain length, but is trivial. Changing the value of F leaves the pattern of the modes the same, but scales the frequency by s^{-1} where s is the scaling factor of the force, i.e., $F \to sF$. Having the control over the force is more interesting as it can allow one to tune the frequencies to a certain range, which can also be done by scaling the masses, as will be discussed later and is shown in fig. 1.

We will first examine placing the same masses across the string.



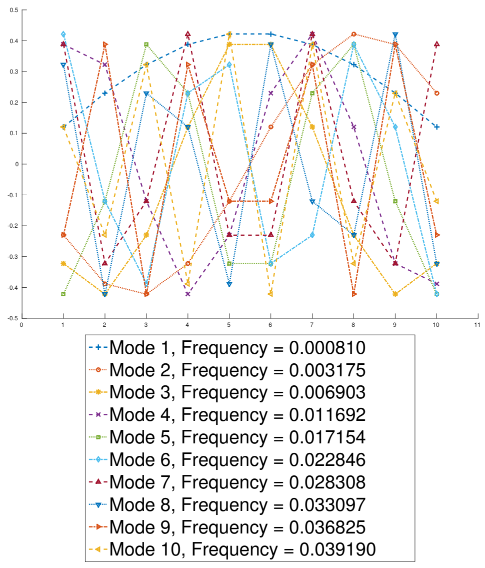
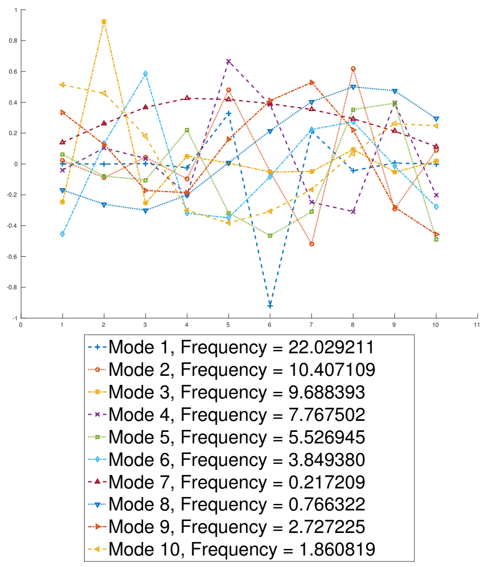


Figure 1: Normal Modes of All Masses M_{Low} (Top) and M_{High} (Bottom)

Figure 1 shows how the string responds when masses are places evenly across the string. The natural modes are exactly the same, but the frequency is divided by the mmultiplier of the masses. This indicates that we could achieve a certain set of frequencies within a given error by scaling the masses by a certain amount. There is also a good amount of variation in the eigenmodes and it would be suspected that using more masses could provide access to a broader range of frequencies. This case could be used to systematically place masses to achieve a set of desired frequencies.

We will now look at placing different masses around the string. To achieve this, the an array of masses was created with the MATLAB command rand(10,1), creating 10 random masses – this command could be used with another number replacing 10 to create any number of masses.



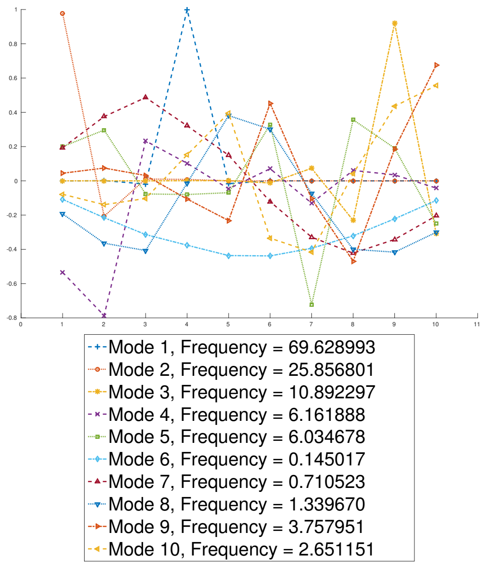


Figure 2: Normal Modes of 2 Sets of 10 Random Masses

As can be seen in fig. 2, the normal (natural) modes of masses placed randomly on a string is very chaotic. Due to the randomness of the normal modes and variations of frequencies of the modes, it is likely that a very broad range of motion may be formed by these normal modes; however, it would be very unwise to randomly generate masses until a desired set of frequencies is reached.

We can also explore the effect of placing the same masses in a variety of configurations. In this exploration, we will consider two sets of equal masses, $M_{\text{Low}} = 1$ and $M_{\text{High}} = 100$ respectively representing small and large masses. We will explore the low and high masses being placed on opposite sides of the string, as well as the low and high masses being places on one side of the string with random masses on the other side, using M_{Low} and M_{High} as multipliers for the random masses where $M_{\text{Random}} \in [0, 1]$ before being multiplied by M_{Low} or M_{High}. Once again, 10 masses will be used.

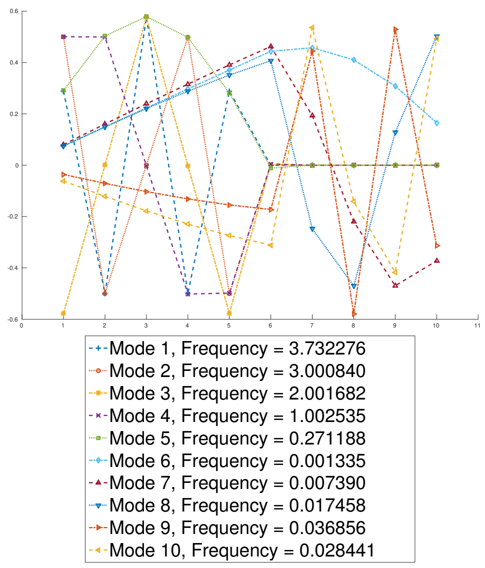


Figure 3: Normal Modes of M_{Low} on Left and M_{High} on Right

In fig. 3, it is seen that in all normal modes except one, primarily the smaller masses are moving, which is to be expected. There are also eigenmodes in which the smaller masses all move in a certain way, producing more movement of the larger masses; however, these are the less frequent eigenmodes.

We will now look at the effects of using constant masses with either high or low random masses on the other sides.

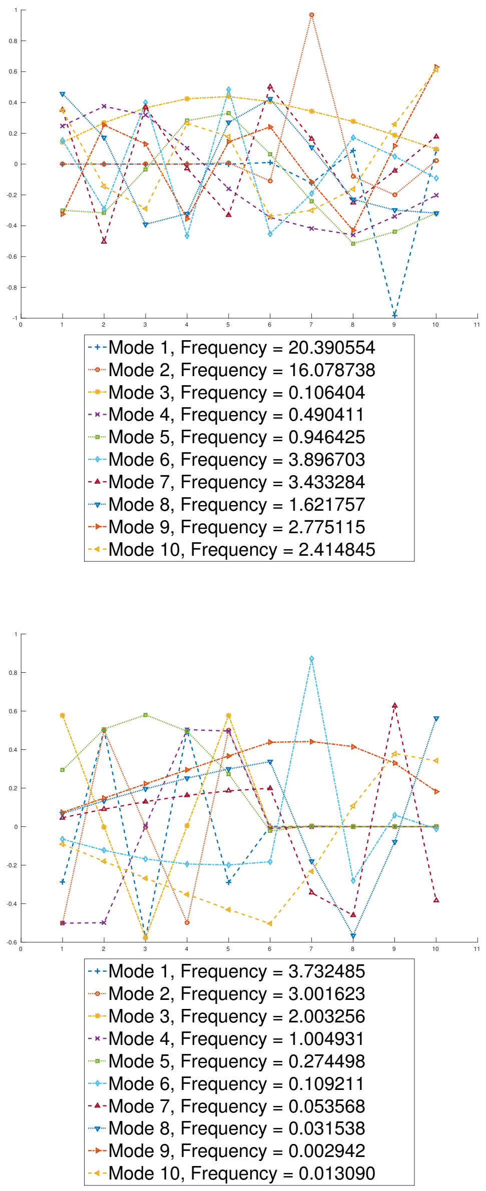
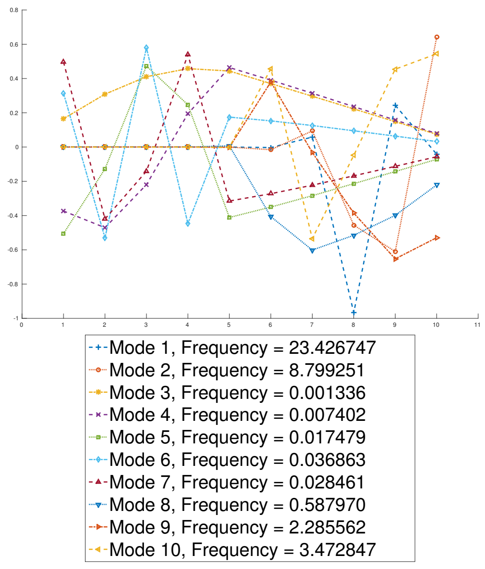


Figure 4: Low Masses on Left with Low (Top) and High (Bottom) Random Masses on Right

Figure 4 shows the effects of random masses on the oppopsite side of a string from low masses. These plots can be compared with the plots in fig. 5. It can be seen that the modes in all cases are quite chaotic. Having a chaotic system such as this would allow for constructing arbitrary frequencies, although it would once again be unwise to generate random masses until the desired set of frequencies is achieved.



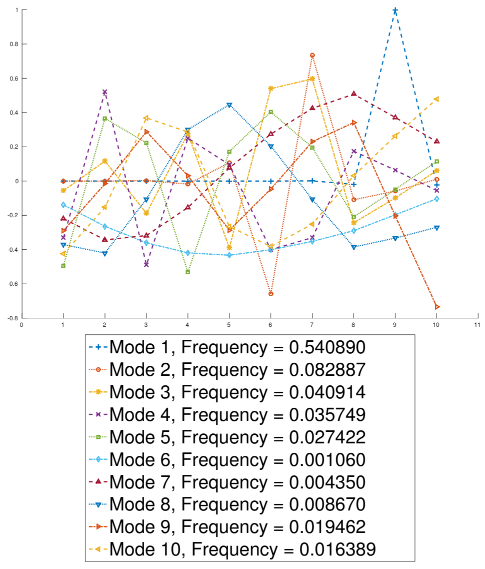


Figure 5: High Masses on Left with Low (Top) and High (Bottom) Random Masses on Right

From this exploration of the inverse eigenvalue problem, it seems that frequencies in general are quite arbitrary when the parameters of a system are able to be changed. However, for a given physical system, such as a violin, the modes and frequencies are already determined. Furthermore, in a system such as a violin, the masses on the string would be represented by infinitesimal mass dm with width between the masses $h \to 0$. The system represented here is similar to a set of discrete masses connected by springs with the displacements of the masses X_i representing the distance of the mass from the equilibrium position of the spring. It should also be noted that while there are a given number of natural modes with their respective natural frequencies, superposition of these modes also happens as a system evolves in time.

Appendices

A Code Listing

Below is a listing of the MATLAB code used in this project. The most up to date code with the code for this report and the produced figures can be accessed at `https://github.com/danielunderwood/string-eigenvalue-analysis` and a compiled version of the latest report can be found at `https://www.sharelatex.com/github/repos/danielunderwood/string-eigenvalue-analysis/builds/latest/output.pdf`.

Listing 1: string_eigenvalue_solver.m

```
function [ naturalFrequencies, naturalModes ] = ...
string_eigenvalue_solver( masses, F, h )
% string_eigenvalue_solver Solves simple eigenvalue problem of a string
%   Solves simple eigenvalue problem of a string with user-given masses in
%   a column or row vector, constant tension force F on the string and
%   separation of masses h

% Get dimension of arrays by size of masses vector
% Getting the maximum of the dimensions will allow the mass vector
% to be a column vector or row vector
n = max(size(masses,1), size(masses,2));

% --- Generate A ---
% 2s on main diagonal
A = 2*eye(n);

% -1s on second diagonals
diagOnes = -1 * ones(n-1,1);
A = A + diag(diagOnes,1) + diag(diagOnes,-1);

% --- Generate D ---
% Diagonal elements
diags = F ./ (h * masses);
D = diag(diags);

% Get eigenvalues and eigenvectors (natrual frequencies) of D * A
[naturalModes, eigenVals] = eig(D * A);

% Gather naturalModes in case a gpuArray was used
naturalModes = gather(naturalModes);

% Natural frequencies are square roots of eigenvalues times -i
naturalFrequencies = -1i * sqrt(eigenVals);

% --- Plotting ---
% Range of numbers for plotting
maxVal = h * n;
x = h:h:maxVal;

% Create new figure
plotFig = figure;
plotAx = axes;
hold(plotAx);

```

```
% Markers for plotting
markers = {'--+',':o','-.*','--x',':s','-.d', ...
    '--^',':v','-.>','--<',':p','-.h'};

% Plot eigenmodes with frequencies
for i=1:size(naturalModes, 2)
    plot(plotAx, x, naturalModes(:,i), markers{mod(i,numel(markers)) + 1}, ...
        'DisplayName', sprintf('Mode %d, Frequency = %f', ...
        i, eigenVals(i,i)), 'Linewidth', 2);
end

% Set limits of plot
xlim(plotAx, [min(x) - 1, max(x) + 1]);

% Show legend
plotLegend = legend('-DynamicLegend');
set(plotLegend, 'FontSize', 30);
set(plotLegend, 'Location', 'southoutside');

% Toggle off hold on axes
hold(plotAx);
end
```