Lab 3: Springs I

Introduction

The next two labs investigate different aspects of the Strang Quartet model of two-dimensional spring networks. In this first lab, we shall verify the ability of our model to predict displacements given a known load. Next week, we shall physically realize Exercise [2] in Chapter 2 of the course notes: detecting a stiff spring in a network.

The present lab consists of two parts. In part one, we will determine spring constants for two types of springs. In part two, we will arrange these springs into a network, load the net, capture the ensuing displacement, and compare the results with those predicted by the Strang Quartet. The data collected in both these sections will be reused for the next lab.

Part 1: Measuring Hooke’s Constants

Our primary hardware for this lab is a force table, shown in Figure 3.1, on which we can mount a network of springs that we load by suspending masses. These masses are attached to the network via strings that run atop pulleys, then drop over the edge of the table. For this lab we will be working with springs of three different stiffnesses: identical horizontal and vertical springs of rest length 4.5 centimeters, one different horizontal spring of rest length 4.5 centimeters, and diagonal springs of length 7 centimeters.

Before we can intelligently predict the behavior of our network, we must first estimate the Hooke’s constants for these two types of springs. Toward this end, pick one example of each. For each spring, secure one end to an immobile support (e.g., the metal bar protruding onto the table) and the other to string that runs over a pulley and attaches to a mass hook hung over the side of the table. You will then place brass masses on the hook and measure

Figure 3.1: Force table and spring network.
the associated elongation, in units of meters (m).

(We will describe later how to accurately measure this distance.) As you do this for, say, four distinct masses, \( m_1, m_2, m_3 \) and \( m_4 \), in units of kilograms (kg), you will have applied forces \( f_j = m_j g \), where \( g = 9.81 \text{ m/s}^2 \), in units of Newtons (N = \( \text{kg m/s}^2 \)). (We recommend that you begin with a single 0.1 kg mass and increase at most 0.2 kg from there; the hook has mass 0.05 kg.)

Hooke’s law informs us that \( f_j = k e_j \), where the spring constant \( k \) is in units of N/m. In order to reconcile these four distinct empirical estimates, \( k = f_j/e_j \), we wish to find the value of \( k \) that minimizes the misfit between the theoretical force \( k e_j \) and the experimental measurement \( f_j \).

There are various ways to compute this misfit; one appealing approach, to be justified in Chapter 5 of the course notes, minimizes the sum of the squares of the errors. In particular, we define the misfit function

\[
M(k) = \frac{1}{2} \sum_{j=1}^{4} (f_j - k e_j)^2
\]

and identify its minimum by solving \( M'(k) = 0 \). When you do this you will arrive at the estimate of the best

\[
k = \frac{e^T f}{e^T e}.
\]

For example, if the masses, forces, and elongations are

\[
m = \begin{bmatrix}
0.55 \\
0.75 \\
0.95 \\
1.15 \\
\end{bmatrix} \text{ kg}, \quad
f = \begin{bmatrix}
5.3955 \\
7.3575 \\
9.3195 \\
10.3005 \\
\end{bmatrix} \text{ N}, \quad
e = \begin{bmatrix}
0.011 \\
0.0175 \\
0.022 \\
0.028 \\
\end{bmatrix} \text{ m},
\]

then \( k \approx 418 \text{ N/m} \). It may help to visualize this data, as in Figure 3.2.

**Part 2: Predicting Displacements of a Small Network**

Now we wish to experiment with the spring network shown in Figure 3.3, made up of sixteen springs and nine nodes, two of which are fixed in place to remove four degrees of freedom from the problem. We are left with seven free nodes (hence fourteen degrees of freedom), and the resulting adjacency matrix \( A \) and Strang Quartet matrix \( A^T K A \) are nonsingular. (It can be quite difficult to spot instabilities by simply looking at the network; see the example in the Epilogue to this lab.)

We build this network using the 4.5 centimeter springs for the horizontal and vertical components, and the 7 centimeter springs for the diagonal components. Springs are connected together using the pennies as the nodes. The pennies are also connected to strings that go over the pulleys and are connected to suspended weights. The pulleys themselves are free to slide up and down along the steel rod; ensure they are positioned so that the forces are aligned in the horizontal and vertical directions, rather than being askew.

Unlike the scenario described in the course notes, we do not have the leisure of accurately measuring the node locations for a completely unloaded network. (Even with two nodes fixed, the
\[ k = (f^T e) / (e^T e) \approx 418 \text{ N/m} \]

Figure 3.2: Computing a spring constant.

Figure 3.3: The spring network for this lab. The dashed lines indicate wires from which masses are to be attached; the gray nodes are tied in a fixed position to anchor the network. Two types of springs should be used in this lab: one for all horizontal and vertical members, another for all those on the diagonals.

other springs and nodes will fall slack without any load.) Instead, we shall first load the network with a control load and measure the node locations (using the webcam, as described below). Then we shall adjust the forces, make predictions for the displacements from the control using the Strang
Quartet, and check the veracity of the model using measurements of the new node locations.

To construct the control configuration, suspend weights from each of the nodes such that the spring network is as square as possible and each spring is slightly elongated. If the springs are not elongated in the control configuration, small displacements will not experience the force applied by the spring. This effect breaks the linearity key to studying this problem. When the spring network is not square, then the adjacency matrix built assuming the angles are square will not be a good model for the physical system. Both of these must be satisfied to obtain good results.

Experience has shown that suspending approximately .150 kg from node 2 and node 6 in both directions approximately satisfies these constraints. Applying additional weights to further straighten the network in the reference configuration will improve the accuracy of your measurements.

Observe that this load \( f^{(1)} \), the pennies will displace by \( x^{(1)} \), according to the Strang Quartet:

\[
A^T K A x^{(1)} = f^{(1)}. \tag{3.3}
\]

You can (and should) measure \( f^{(1)} \), but it would be difficult to measure the displacements \( x^{(1)} \), because we do not know the positions of the nodes in the absence of any load. Use the webcam to record the positions of the nodes under the load \( f^{(1)} \); let us call this vector of positions \( p^{(1)} \).

Having recorded \( p^{(1)} \), add further masses to obtain a load \( f^{(2)} \). Now the masses displace according to

\[
A^T K A x^{(2)} = f^{(2)}. \tag{3.4}
\]

As before, we cannot measure the displacement \( x^{(2)} \) directly, but with the webcam we can measure the new positions, say \( p^{(2)} \). (We assume that \( p^{(2)} \) is measured in the same coordinate frame as \( p^{(1)} \), i.e., that you have not bumped the table or webcam between experiments!) Now the difference of the two webcam measurements gives us the difference of displacements:

\[
p^{(2)} - p^{(1)} = x^{(2)} - x^{(1)}. \tag{3.5}
\]

We can obtain a Strangian equation for this difference by subtracting (3.3) from (3.4):

\[
A^T K A (x^{(2)} - x^{(1)}) = f^{(2)} - f^{(1)}. \tag{3.6}
\]

With this equation in hand, we can predict the difference \( x^{(2)} - x^{(1)} \) knowing only \( f^{(1)} \) and \( f^{(2)} \). For this particular network, the matrix \( A^T K A \) will be invertible, and so we can compute the difference using

\[
x^{(2)} - x^{(1)} = (A^T K A)^{-1} (f^{(2)} - f^{(1)}). \tag{3.7}
\]

Now we can predict the location of the perturbed state via

\[
\tilde{p}^{(2)} = p^{(1)} + x^{(2)} - x^{(1)}. \tag{3.8}
\]

Our goal is to compare this prediction \( \tilde{p}^{(2)} \) with the measurements of the perturbed location \( p^{(2)} \) obtained via the lab webcam.

Lab Hardware and Software
When you enter the lab, the webcam should already be mounted on a stand and approximately centered above the force table. The following command will open up a preview window in MATLAB of the table surface:

```matlab
vid = videoinput('winvideo',1,'RGB24_640x480');
preview(vid);
```

Center the camera, and rotate the table such that the network appears square and flat on the preview screen. You may need to focus or adjust lighting. To focus, turn the white ring on the camera’s front. To change lighting, click on the white webcam item in the toolbar. Under the tab camera controls, adjust shutter speed until a reasonable image is obtained. Do not change the color balance.

A script has been provided that will locate red pennies and return a vector of their locations. The script is located under `C:\CAAMlab\Scripts` and is called `getloc`. (You may wish to `addpath C:\CAAMlab\Scripts` so that MATLAB finds `getloc` whatever your working directory.) To locate `N` pennies, call

```matlab
p = getloc(N);
```

This function requires you to click on the `N` different pennies in the image, and returns a vector `p` with `2N` components denoting the centers of the pennies in pixels, listed in the order in which you clicked on them. Thus the first two entries of `p` denote the horizontal and vertical positions of the first penny; the next two entries give the position of the second penny, and so forth.

To begin, you will need to calibrate the experiment so that you can convert the pixel measurements from `getloc` into metric distances. (This is necessary because of the adjustable height of the arm on which the webcam is mounted.) To calibrate, measure the distance between two pennies in the network; we recommend the pennies attached to the aluminum bars as these have their centers clearly marked. Now run `getloc` with `N=2` to locate these two pennies. The pixel distance between the pennies is thus

$$\sqrt{(p(3) - p(1))^2 + (p(4) - p(2))^2},$$

giving the conversion factor from pixels to meters of

$$\text{pix2m} = \frac{0.1}{\sqrt{(p(3) - p(1))^2 + (p(4) - p(2))^2}}.$$

With this conversion factor at hand, we can proceed to the main experiment described above in Figure 3.3. Set up the loaded network and run

```matlab
p = getloc(6); % obtain the position of 6 pennies
p = p*pix2m; % convert pixels to meters
```

**Tasks**
1. Derive the equation (3.2) by setting the derivative of (3.1) to zero and solving for $k$.

2. Estimate spring constants for each of the two types of spring used. Use `getloc` to obtain displacement measurements. Include plots of force versus elongation, as in Figure 3.2. Label and title each plot as above. If so inclined, repeat the experiment for several of the other springs you will use in the sixteen-spring network to investigate their uniformity. In addition to using the formula given above, compute both the spring constant and rest length simultaneously. The Matlab command `polyfit` may be used for this purpose.

3. Derive the adjacency matrix $A$ for the network in Figure 3.3, and use the `null` command in MATLAB to verify that it possesses a trivial null space. You may wish to modify the code `truss.m` from the course materials for this purpose.

4. Conduct the control experiment and variable experiment with the sixteen-spring network. Before taking the measurements for the control experiment, attempt to load the network such that it is as square as possible. For the variable experiment, apply a 0.1 kg load to each of the eight mass hangers.

How do your predictions of the displacement $x_2 - x_1$ match the measured quantities? List some potential sources of error in this procedure. (We expect that your data may be fairly messy for this lab.)

For two experiments, include an image of the perturbed state taken by the webcam, overlaid with squares marking the unperturbed location $p^{(1)}$, circles showing the perturbed location $p^{(2)}$ measured from the webcam, and dots at the predicted perturbed location $\tilde{p}^{(2)}$, as measured by the Strang Quartet. Use large, colorful markers that are visible against the background of the force table image; for example, to plot large, white squares, use

```matlab
plot(p(1:2:end-1), p(2:2:end), 'ws', 'markersize', 10, 'markerfacecolor', 'w')
```

Finally, use the measured change in positions $p^{(1)}$ and $p^{(2)}$ to predict the change in forces, i.e., compare the expected value $f^{(2)} - f^{(1)}$ to the prediction

$$A^T K A (p^{(2)} - p^{(1)}) = A^T K A (x^{(2)} - x^{(1)}).$$

Save the masses and displacements for each experiment for use in the next lab.

5. (Optional) In constructing the adjacency matrix, we assumed that all the angles were either right angles or $45^\circ$ angles, however the physical network is not as simple. Using the node positions you measured in the lab using image capture, compute the actual adjacency matrix. Now recompute the displacements of your network from the applied load. Is your answer more accurate? How much so?
6. (Optional, very challenging) A critical assumption in computing network displacement is that the motion is small. Otherwise as the load is applied, the springs skew changing the angles between them and hence the adjacency matrix. Hence this problem is *nonlinear*. Construct an algorithm to solve this nonlinear problem.