CAAM 335 · MATRIX ANALYSIS

Physical Laboratory

Steven J. Cox, Mark Embree, Jeffrey M. Hokanson

Computational and Applied Mathematics

Rice University

draft version, 23 August 2008

Contents

1. Circuits I 1

2. Circuits II 7

3. Springs I 17

4. Springs II 24

5. Frequency Identification via Fourier Transformation 27

6. Discovering Frequencies in a Time Series 38

7. RLC Circuit Dynamics 42

8. Dynamics of Compound Pendula 50

9. Dynamics of a Beaded String 61

10. Hearing the Beads on a String 70

Preface

The seventeenth and eighteenth centuries witnessed momentous progress in our understanding of the world. The savants responsible for this progress were not physicists, nor astronomers, nor mathematicians, but rather *natural philosophers*, masters who drew upon a range of skills to attack the fundamental questions that surrounded them. Since their time these tools have been perfected, extended, specialized. For example, the development of matrix techniques and spectral theory during the nineteenth century provided a mechanism for organizing and understanding ever-larger physical models.

Too often in our classrooms, such mathematical techniques are separated from the applications that motivate them. Our goal is to supplement undergraduate linear algebra education with a series of physical laboratories that make tangible – and extend – the material typically covered in such a course. *Inverse problems* form a prevailing theme, arising in the context of electrical impedance tomography, structures, and vibrating systems.

These notes were developed to accompany the CAAM 335 Matrix Analysis course offered each semester at Rice University. Students may participate in the physical laboratory for one extra course credit. We are very grateful for the helpful comments from students who have participated in these labs, who have impressed us with their persistence and insight.

Caveat lector: These notes are a work-in-progress, and may still contain errors, typographical or more substantial. We are grateful for the assistance of Kenneth Davis, who drafted the early text for several of these labs, to Brian Leake for his extensive comments and creative suggestions, and to support from a Brown Teaching Grant during the summer of 2007.

Lab 1: Circuits I

▶ Introduction

The course notes develop a model of a nerve fiber as an electrical network, which in turn reduces to a system of linear algebraic equations ($A^TGAx = f$) using a four-step methodology, the 'Strang Quartet'. In this lab, you shall investigate the accuracy of such a mathematical description of a circuit by comparing your predictions to measurements from several physical circuits.

▶ Using the Breadboard

The central component of our testbed is a *breadboard*, a rectangular plastic card lined with holes arranged in a specific pattern. This pattern governs how these holes are connected to one another,

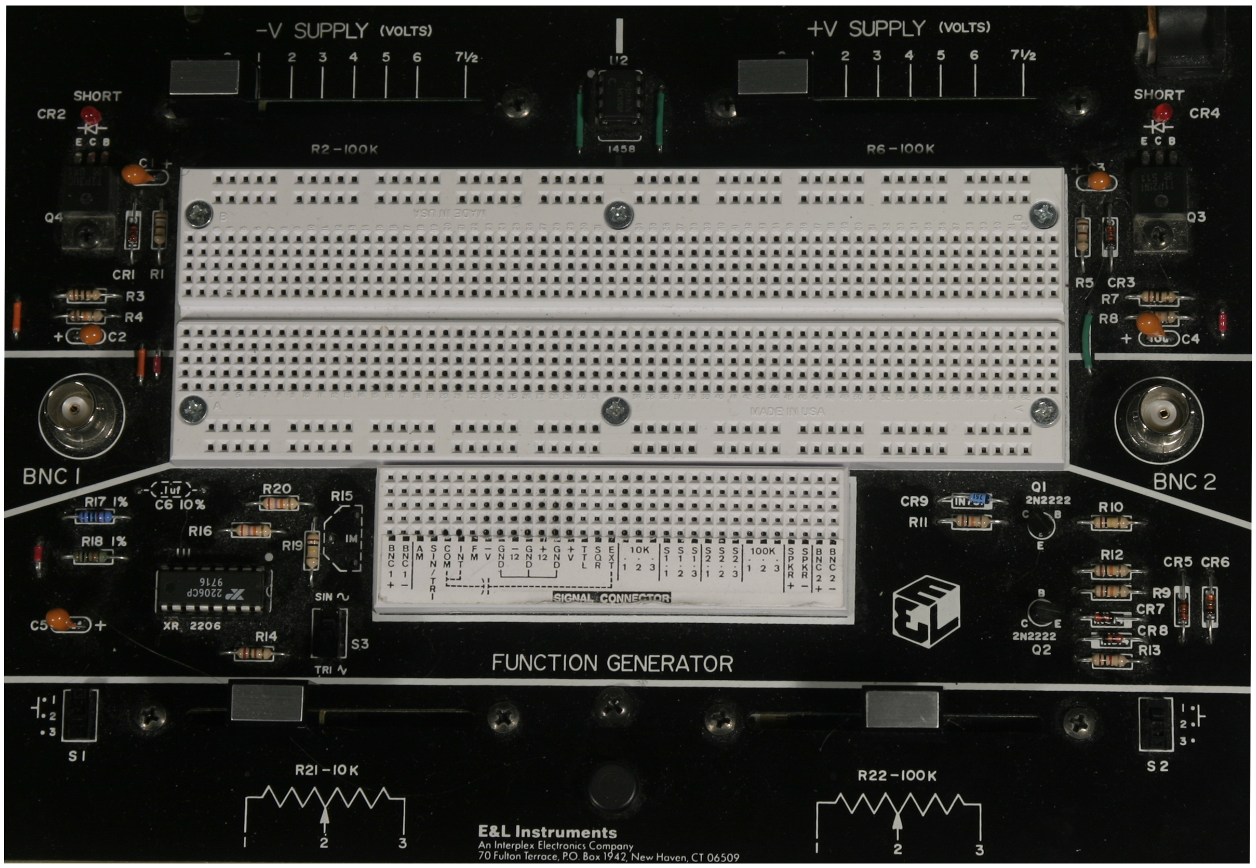


Figure 1.1: The lab breadboard (large white rectangle in center), with supporting elements.

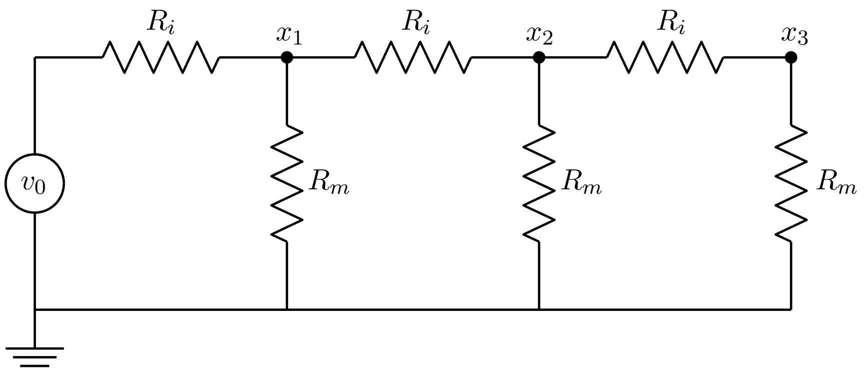


Figure 1.2: The three-compartment nerve fiber circuit from the CAAM 335 class notes, with a potential v_0 replacing the current source i_0 given in the notes.

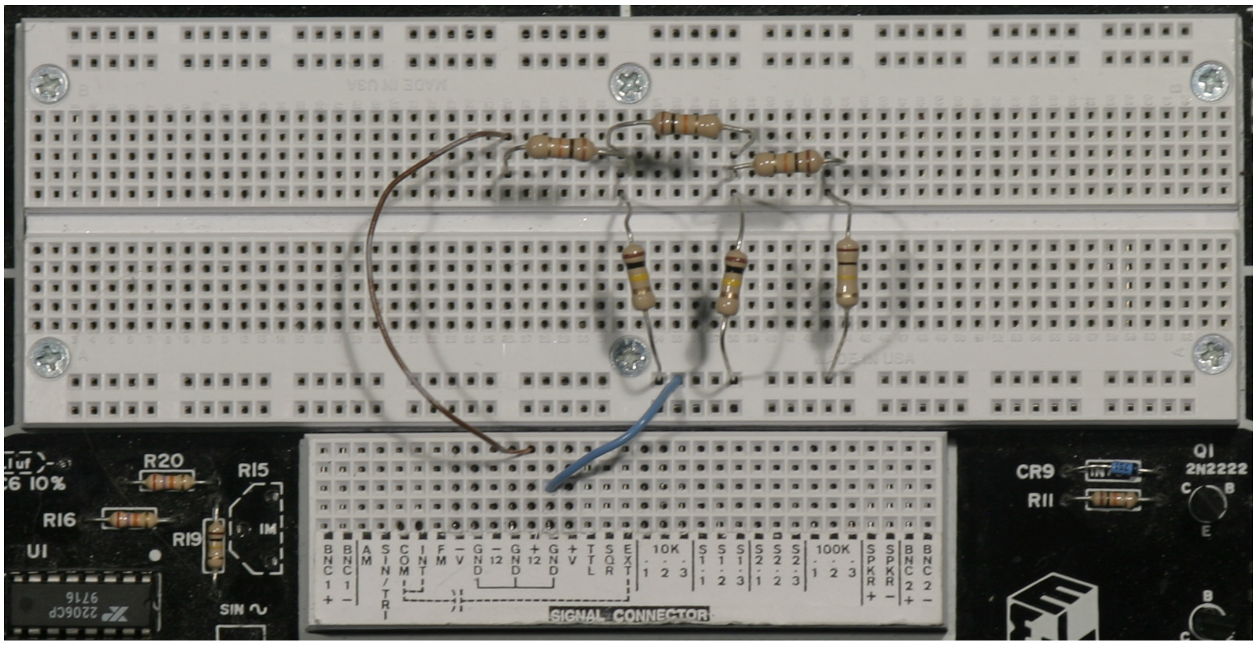


Figure 1.3: An implementation of the circuit from Figure 1.2 using an input voltage of +12 V.

underneath the board. You will use these connections to configure simple circuits: By plugging the (stripped) end of a wire into one of these holes, you connect that wire to any other device plugged into an associated hole. Understanding the pattern of connections is clearly fundamental.

The breadboard, which forms the largest panel in Figure 1.1, has four major sets of holes. The two middle sets each consist of 64 columns of five holes each; each of these five holes is connected to all other holes in that column. This means that each column can act as a separate node in our circuit. To illustrate this, consider the three-compartment nerve model shown in Figure 1.2. (Here we presume we know a *potential* v_0 (in volts), rather than the *current* i_0 as in the class notes; as a result, there is no unknown potential between the voltage source and the first resistor.) Figure 1.3

shows one way to wire this circuit up on the breadboard. Notice how three resistors are connected to the node at x_1. Also notice that the holes are *not* connected across the divide that separates the two 64 column sets. This makes it simple to wire up the membrane resistors (R_m) in parallel.

Now we turn to the two remaining sets of holes on the breadboard, those at the top and bottom of the breadboard each consisting of two rows broken into ten sections of ten holes each. Each row is connected horizontally, independently of the sections but disconnected in the middle (near the screw). Thus, each row gives two connected components. There are no vertical connections, so between the top and bottom sets of holes, we have a total of eight independent components. Although we will use these upper and lower sets as nodes, they are generally used in applying a current to the circuit. We, however, possess alternate means of supplying power.

Just below the breadboard is a much smaller strip with labeled columns that will be our means of supplying power and variable resistance to the circuit. Just as before, the holes of this section are connected vertically. We are most interested in the columns labeled GND (there are three of them), −V, +V, −12, and +12. GND denotes the ground, −12 and +12 refer to the voltage (in volts) of constant potential sources, and −V and +V are variable input voltages that can be adjusted by levers just above the breadboard. As shown in the Figure 1.3, we can simply use a wire to connect the circuit to the power supply.

There are two sliders below the breadboard identical to the voltage sliders above the breadboard. These lower sliders, along with the corresponding sections on the smaller strip marked 10K and 100K, can be used to apply variable resistance. If we choose to utilize a variable resistor, or potentiometer, of resistance no greater than 10 kΩ (actually, it is approximately 9.5 kΩ) in our circuit, we must connect two wires to the 10K section – one in the slot labeled 1 and the other in slot 2. The opposite ends of these two wires should be inserted into the circuit in the same manner that an actual resistor would be connected. If we know the resistance value we wish to use beforehand, we should be sure to set the resistance before connecting it to the rest of the circuit because of our method of measurement.

▶ Measurements

During this lab it will be necessary to acquire data from the circuit we have constructed. More specifically, we will need to measure the resistance of and the potential drop across each resistor. To do so, we will use a multimeter.

A multimeter is a device that can measure current, potential difference, and resistance; we shall only be concerned with the latter two in this lab. The multimeter is divided into five sections labeled with V, Ω, and A for measuring voltage, resistance, and current, respectively. There are two sections labeled A and two labeled V, one pair of A and V accounting for direct current (DC), the other for alternating current (AC). In this lab, we are interested in a constant potential source (i.e., DC), and hence will use the A and V marked $\overline{---}$. The multimeter has a central dial that can be set to any of several numerically labeled positions. This value corresponds to the maximum value that the multimeter can record. For example, if the dial is set to '200 V', we will be unable to measure any voltage greater than 200 V. You may have to adjust the dial while measuring in order to get a reading.

When using the multimeter, one must be careful of where to plug in and place the leads. The negative (black) lead should always be plugged in to the slot marked COM while the positive (red) lead should be plugged in to one of the other three. In our case only the slot marked V Ω will be needed. When both leads are plugged in correctly, we are ready to take a measurement. When measuring a voltage drop across a resistor, we place the negative lead on one side of the resistor and the positive lead on the other so that current is flowing from the positive to the negative. Reversing these positions will result in a voltage of opposite sign. In this lab however, we will be measuring the potential at each node with respect to ground so the negative lead will remain in GND while we place the positive lead on the appropriate node.

To measure the resistance, place the leads on opposite sides of the resistor. Resistance is always a positive value so the direction of measurement is unimportant. For variable resistors, place the leads at the same nodes to which the wires of the variable resistor are attached. The inner workings of the multimeter allow us to measure the correct resistance only when the variable resistor is not connected to the rest of the circuit.

▶ Measuring Resistance

One can quickly approximate resistance using the color-coded bands on the side of the resistor. Our

Table 1.1: A resistor color code chart (adapted from `http://www.elexp.com/t_resist.htm`).

COLOR	1st BAND	2nd BAND	MULTIPLIER	TOLERANCE
black	0	0	1 Ω	
brown	1	1	10 Ω	±1%
red	2	2	100 Ω	±2%
orange	3	3	1 KΩ	
yellow	4	4	10 KΩ	
green	5	5	100 KΩ	±0.5%
blue	6	6	1 MΩ	±0.25%
violet	7	7	1 MΩ	±0.10%
gray	8	8		±0.05%
white	9	9		
gold			0.1 Ω	±5%
silver			0.01 Ω	±10%

resistors have four bands. To read the resistance, place the gold or silver colored band to the right. Then reading left to right, the first two bands will be the first two digits of the resistance. The third band gives the value of a multiplier. The final band indicates the tolerance of the resistor, referring the difference between labeled resistance and measured resistance. A table of the color codes is shown in Table 1.1.

For example, a resistor with brown, black, red, and gold bands, in that order, has a value of $10 \times 10^2\ \Omega$, or 1 kΩ, with a tolerance of 5%, meaning that we can expect a resistance value in the range .95 − 1.05 kΩ. To convince yourself of this, try using Table 1.1 to predict a resistance value, which you can then check with the multimeter.

▶ Wheatstone Bridge

One circuit you will encounter in this lab is the famous Wheatstone bridge, named after Sir Charles Wheatstone who greatly improved upon Samuel Hunter Christie's original invention. First introduced in 1833, the Wheatstone bridge is a circuit that is commonly used to capture unknown resistances. As shown in Figure 1.4, the Wheatstone bridge consists of four resistors: two with known resistances (R_1 and R_2), one with variable resistance (R_3), and one with unknown resistance (R_4). Whenever a voltmeter placed between nodes x_1 and x_2 reads 0 V, the bridge is balanced and the following equality holds:

$$R_4 = \frac{R_1 R_3}{R_2} \tag{1.1}$$

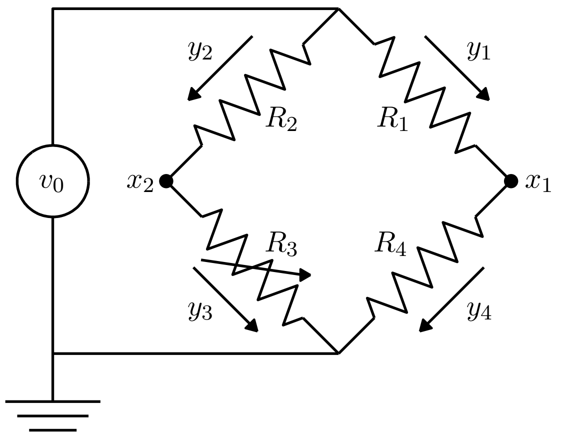


Figure 1.4: The Wheatstone bridge circuit diagram.

▶ Lab Instructions

1. Build and analyze five- and ten-compartment models of the nerve cell per the diagram in Figure 1.5. To be precise, we learned in Section 1.1 of the course notes that the axial resistance of an N-compartment fiber obeys $R_i = (\rho_i \ell/(\pi a^2))/N$, while the membrane resistance is $R_m =$

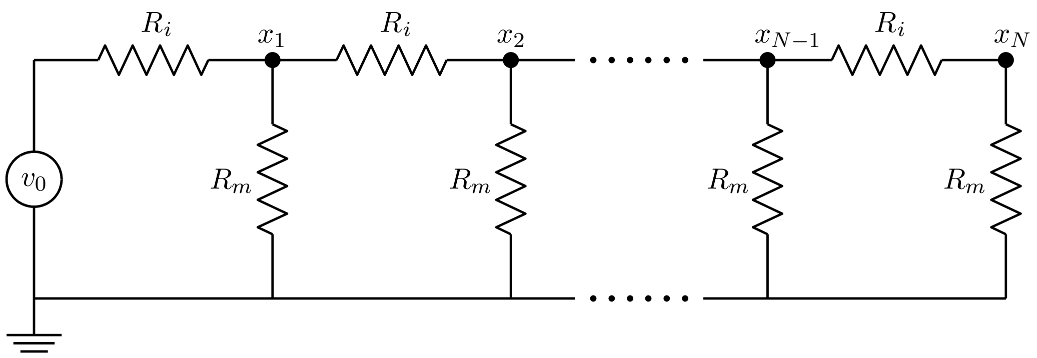


Figure 1.5: The N-compartment model of the neuron.

$\rho_m N/(2\pi a\ell)$. We suppose that $\ell = 1\,\text{cm}$ and $(\rho_i\ell/(\pi a^2)) = 10\,\text{k}\Omega$ and $\rho_m/(2\pi a\ell) = 5\,\text{k}\Omega$. We also suppose that $v_0 = 10\,\text{V}$.

(a) With $N = 5$ build the 5-compartment circuit by proper placement of $R_i = 2\,\text{k}\Omega$ and $R_m = 25\,\text{k}\Omega$ resistors. Use the multimeter to measure the 5 potentials, with respect to ground, and graph them in MATLAB in the manner used in Figure 1.4 of the course notes, with line type `'o-'`.

(b) With $N = 10$ build the 10-compartment circuit by proper placement of $R_i = 1\,\text{k}\Omega$ and $R_m = 50\,\text{k}\Omega$ resistors. Use the multimeter to measure the 10 potentials, with respect to ground, and graph them in the same figure created in (a), this time with line type `'x-'`.

(c) Read exercise [1.2] in the course notes and compare (in prose) your two curves with one another and with Figure 1.4 of the course notes. What do you expect to happen as N increases? Submit your graph and prose.

2. Regarding the Wheatstone bridge:

(a) On paper and by hand, follow the four steps of the Strang Quartet. Carefully show that if $x_1 = x_2$ then R_4 may indeed be determined via equation (1.1).

(b) Construct a Wheatstone bridge on the breadboard, with $v_0 = 10\,\text{V}$, $R_1 = R_2 = 10\,\text{k}\Omega$, $R_4 = 50\,\text{k}\Omega$, and R_3 determined via the built-in R22 potentiometer. Use the multimeter to (i) measure the true values of R_1, R_2 and R_4, and (ii) display the potential difference $x_1 - x_2$. Now vary R_3 until $x_1 - x_2$ is as close as possible to zero. Use the multimeter to determine the value of R_3 and assess (in complete sentences) the accuracy of (1.1). Submit a table of all measured values and your written assessment.

Lab 2: Circuits II

▶ Introduction

It is rare today to design, let alone fabricate, a complicated electrical device prior to simulating its behavior on a computer. The simulation requires the *construction* of a mathematical model. Matrix algebra is the language for expressing such models.

As an example, let us consider the medical application of EIT, Electrical Impedance Tomography. The premise behind this technology is that healthy and diseased tissue present significantly different resistance to electrical current. With regard to a brain tumor, one exploits this difference by applying a (safe) current across two electrodes placed on the skull. By measuring the resulting potential difference across a companion pair of electrodes, and repeating the test at a variety of sites around the skull one can determine the size, location, and conductivity of the tumor. In order to *simulate* such a procedure we must construct a mathematical model of a resistor network and be able to *compute* the potential difference between two points on the 'boundary' from knowledge of the current applied to the network.

▶ Layer-Stripping, Theory

The network portrayed in Figure 2.1 is a crude conduction model of a cross-section of the human head. Assuming that diseased tissue offers a different resistance to current than healthy tissue, one asks whether providing stimulus and measuring response at the 'boundary' of the network might produce information about the R_j values in the 'interior.'

In the interest of actually putting this into practice, the rules of the game state that, *at the boundary*, one may

(i) ground any one node,

(ii) specify every current, and

(iii) measure every potential.

The difficulty is to determine which current patterns coax from the network the most information about its conductance. The simplest approach to this problem is the so-called layer-stripping technique of Curtis and Morrow ("Determining the Resistors in a Network," *SIAM J. Appl. Math.* 50(3), pp. 918–930, 1990).

The phrase 'layer-stripping' refers to starting at one corner of the network and sequentially determining diagonal layers of resistors.

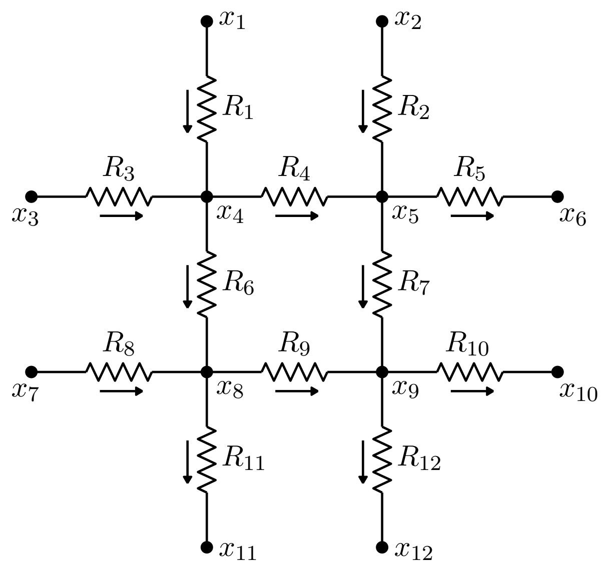


Figure 2.1: A network of resistors, forming a crude model of a cross-section of a human head.

To begin, we note that rules (i)–(iii), together with the laws of Ohm and Kirchhoff, permit the easy determination of the resistors at every corner. Let us start at the upper right. In particular, let us inject f_2 amperes of current at x_2 and remove f_2 amperes of current at x_6. Recalling

> **Ohm's Law:** The potential difference (tail minus head) across a resistor equals the product of its resistance and current,

we find

$$x_2 - x_5 = R_2 f_2 \quad \text{and} \quad x_5 - x_6 = R_5 f_2.$$

As both x_2 and x_6 may be measured, we find that both R_2 and R_5 are hence readily determined once we know x_5. We extract this interior potential through a clever use of rule (i). Namely, we ground node 3, i.e., set $x_3 = 0$. Let 'edge j' denote the edge with resistor R_j. As current enters and leaves through edges 2 and 5, it follows that all the other boundary currents are zero. Denoting the current on edge j by y_j we find, in particular, that $y_3 = 0$ and hence Ohm's law on edge 3 produces

$$x_3 - x_4 = R_3 y_3, \quad \text{i.e.,} \quad -x_4 = 0.$$

This implies that Ohm's Law on edge 4 takes the form

$$-x_5 = R_4 y_4.$$

Applying

Kirchhoff's Current Law: The net current (in minus out) at each node must vanish,

at node 5 we conclude that $y_4 = 0$ and $x_5 = 0$ as well. As a result,

$$R_2 = x_2/f_2 \quad \text{and} \quad R_5 = -x_6/f_2.$$

The next step is to determine the resistors in the second layer, i.e., R_1, R_4, R_7 and R_{10}. In the first layer we had the luxury of injecting and removing current in a path that contained the resistors of interest. To emulate that at the second layer we should inject in node 1 and remove at node 10 in such a way that y_6 and y_9 both vanish. Unfortunately, the easy fix, i.e., ground nodes 3 and 7, violates rule (i). The way out is to *combine* two experiments in such a way that both x_3 and x_7 vanish.

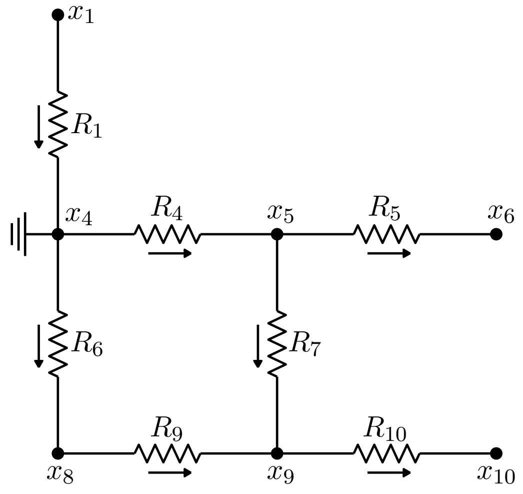


Figure 2.2: A sub-network of Figure 2.1.

In each experiment we inject current at node 1 while grounding node 3. The experiments differ in that current is removed at node 6 in the former and node 10 in the latter. As a result, the current through edges 2, 3, 8, 11, and 12 is zero and so we need only examine the subnet in Figure 2.2. Note also that as the current through edge 2 is zero it follows from Ohm's Law that $x_5 = x_2$. Hence x_5, and similarly x_8 and x_9 are *visible* from the boundary. We gather the vector of relevant potentials in

$$x = [x_1\ x_5\ x_6\ x_8\ x_9\ x_{10}]^T,$$

and their associated differences in

$$e = [e_1\ e_4\ e_5\ e_6\ e_7\ e_9\ e_{10}]^T$$

where

$$e_1 = x_1$$

$$\begin{aligned}
e_4 &= -x_5 \\
e_5 &= x_5 - x_6 \\
e_6 &= -x_8 \\
e_7 &= x_5 - x_9 \\
e_9 &= x_8 - x_9 \\
e_{10} &= x_9 - x_{10}.
\end{aligned}$$

This is more conveniently expressed in the matrix notation

$$e = -Ax \qquad \text{where} \qquad A = \begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 \end{bmatrix}.$$

Next comes Ohm's Law,

$$y_j = e_j/R_j, \quad j = 1, 4, 5, 6, 7, 9, 10$$

or, in matrix notation,

$$y = Ge$$

where

$$G = \begin{bmatrix} 1/R_1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/R_4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/R_5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/R_6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/R_7 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1/R_9 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1/R_{10} \end{bmatrix}$$

Denoting by f_k the current *into* node k apply Kirchhoff's Current Law,

$$\begin{aligned}
f_1 - y_1 &= 0 \\
y_4 - y_5 - y_7 &= 0 \\
y_5 + f_6 &= 0 \\
y_6 - y_9 &= 0 \\
y_7 + y_9 - y_{10} &= 0 \\
y_{10} + f_{10} &= 0
\end{aligned}$$

at each of the ungrounded nodes. In matrix notation this reads

$$By = -f$$

where

$$B = \begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{and} \quad f = \begin{bmatrix} f_1 \\ 0 \\ f_6 \\ 0 \\ 0 \\ f_{10} \end{bmatrix}.$$

Turning back the page we recognize in B the *transpose* of A. Calling it such, we recall our main steps

$$e = -Ax, \quad y = Ge, \quad \text{and} \quad A^T y = -f.$$

On substitution of the first two into the third we arrive at

$$A^T G A x = f. \tag{2.1}$$

This is our desired matrix model for the subnetwork of Figure 2.2. The A matrix encodes the geometry of the network, the G matrix encodes the physical characteristics of the network components, the f vector captures the stimulus and the x measures the network's response. In circuit analysis one meets both (1) problems of analysis: given G and f find x, as well as (2) problems of design or synthesis: what must G be in order that a prescribed f elicit a particular x? In this jargon, our EIT problem is one of synthesis. We agreed at the outset of this lab that we would carry out two distinct experiments on the subnetwork. In both cases we inject 1 ampere at node 1. In the first (second) case we remove it from node 6 (10) respectively. This gives rise to respective f vectors

$$f^{(1)} = \begin{bmatrix} 1 \\ 0 \\ -1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \text{and} \quad f^{(2)} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ -1 \end{bmatrix}$$

and their associated responses, $x^{(1)}$ and $x^{(2)}$. In light of (2.1) we note that they satisfy

$$A^T G A x^{(k)} = f^{(k)}, \quad k = 1, 2. \tag{2.2}$$

With (2.2) we are now close to our hope of grounding node 8. More precisely, for any two scalars a_1 and a_2 it follows directly from (2.2) that

$$A^T G A(a_1 x^{(1)} + a_2 x^{(2)}) = a_1 f^{(1)} + a_2 f^{(2)}.$$

Now one simply chooses a_1 and a_2 so that

$$a_1 x_8^{(1)} + a_2 x_8^{(2)} = 0. \tag{2.3}$$

As there are many possible solutions to (2.3) we append to it another equation that renders the solution unique. Namely, in the interest of keeping the total injected current at 1 ampere, we ask that

$$a_1 + a_2 = 1. \tag{2.4}$$

Combining (2.3) and (2.4) we arrive at

$$\begin{bmatrix} x_8^{(1)} & x_8^{(2)} \\ 1 & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

This system indeed has a unique solution so long as $x_8^{(1)} \neq x_8^{(2)}$. Can you argue on physical grounds why these two potentials should be distinct?

With a_1 and a_2 under our belts, let us now solve for the desired resistances. For ease of reference let us now write x for $a_1 x^{(1)} + a_2 x^{(2)}$ and note that by construction this x corresponds to the subnet below.

Can you explain why grounding node 8 has effectively grounded node 9? From Figure 2.3, R_1 and R_{10} are easily recovered, namely

$$R_1 = x_1 \quad \text{and} \quad R_{10} = -x_{10}/a_2. \tag{2.5}$$

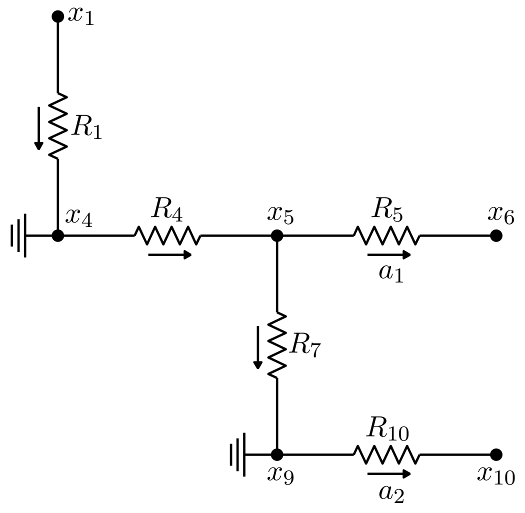


Figure 2.3: A sub-network of Figure 2.2.

As $y_4 = y_1 = 1$ and $y_7 = a_2$, it follows that

$$R_4 = -x_5 \quad \text{and} \quad R_7 = x_5/a_2. \tag{2.6}$$

The remaining resistors may be determined precisely as above by instead starting in the bottom left corner and stripping up.

▶ Layer-Stripping: A Numerical Example

As a simple example let us suppose that the value of each resistance is 1, except R_7, which has the value 10. As the retrieval of R_2 and R_5 is relatively straightforward, let us proceed directly to the second layer. The relevant matrix is

$$A^T G A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2.1 & -1 & 0 & -0.1 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & -1 & 0 \\ 0 & -0.1 & 0 & -1 & 2.1 & -1 \\ 0 & 0 & 0 & 0 & -1 & 1 \end{bmatrix}$$

Solving (2.2) we arrive at

$$x^{(1)} = \begin{bmatrix} 1.0000 \\ -0.9231 \\ -1.9231 \\ -0.0769 \\ -0.1538 \\ -0.1538 \end{bmatrix} \quad \text{and} \quad x^{(2)} = \begin{bmatrix} 1.0000 \\ -0.1538 \\ -0.1538 \\ -0.8462 \\ -1.6923 \\ -2.6923 \end{bmatrix}$$

and so the system for a_1 and a_2 reads

$$\begin{bmatrix} -0.0769 & -0.8462 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

Hence,

$$a_1 = 1.1, \quad a_2 = -0.1$$

and

$$x = \begin{bmatrix} 1 \\ -1 \\ -2.1 \\ 0 \\ 0 \\ 0.1 \end{bmatrix}$$

and so indeed

$$R_4 = -x_5 = 1, \quad R_7 = x_5/a_2 = -1/(-0.1) = 10.$$

▶ Layer-Stripping: Practical Implementation

We present in this section one means by which the main steps of layer-stripping may be implemented. The central problem is to supply/draw equal current to/from two specified nodes while grounding a third. As voltage is typically more easy to regulate than current we shall rely on two voltage sources to get the job done. More precisely, the equipment we require, in addition to the resistor network, is

- One positive voltage supply, call it V_+;
- One negative voltage supply, call it V_-;
- One digital multimeter, measuring ohms, volts and amps.

Setup: Insert 3 wires into holes associated with GND, V_+ and V_-. Insert the black connector into the COM port of the multimeter. Insert the red connector into the red $V\ \Omega$ port of the multimeter. Connect the alligator clip of the red wire to the V_+ wire and the other alligator clip to the GND wire and slide V_+ until the multimeter reads 1.77 V. V_+ will remain at this level throughout the remainder of the experiment.
With reference to Figure 2.1 we proceed as follows.

Layer 1: Connect V_+ to node 2 with the red double-ended jumper. Connect V_- to node 6 with the yellow double-ended jumper. Ground node 3 through the multimeter. This will permit you to measure y_3. Vary V_- until $y_3 = 0$. Remove the multimeter and connect the GND wire to node 3. As $y_3 = 0$ it follows that $x_4 = x_3 = 0$. As y_4 is also 0 then so too is x_5. Now insert the multimeter between V_+ and node 2 and measure y_2. Finally,

$$R_2 = V_+/y_2, \quad R_5 = -V_-/y_2.$$

Layer 2$^{(1)}$: Connect V_+ to node 1. Connect V_- to node 6. Ground node 3 through the multimeter. Vary V_- until $y_3 = 0$. Use the multimeter to measure

$$x^{(1)} = [x_1^{(1)} \ \ x_5^{(1)} \ \ x_6^{(1)} \ \ x_8^{(1)} \ \ x_9^{(1)} \ \ x_{10}^{(1)}]^T,$$

noting that the potential at nodes 5, 8, and 9 may be measured at nodes 2, 7 and 12 due to the absence of current through resistors 2, 8 and 12, respectively. Use the multimeter to measure $y_1^{(1)}$. Recover the edge resistance

$$R_1 = V_+/y_1^{(1)}.$$

Layer 2$^{(2)}$: Connect V_+ to node 1. Connect V_- to node 10. Ground node 3 through the multimeter. Vary V_- until $y_3 = 0$. Use the multimeter to measure

$$x^{(2)} = [x_1^{(2)} \ \ x_5^{(2)} \ \ x_6^{(2)} \ \ x_8^{(2)} \ \ x_9^{(2)} \ \ x_{10}^{(2)}]^T,$$

noting, as before, that the potential at nodes 5, 8, and 9 may be measured at nodes 2, 7 and 12 due to the absence of current through resistors 2, 8 and 12, respectively. Use the multimeter to measure $y_1^{(2)}$. Recover the edge resistance

$$R_{10} = \frac{x_9^{(2)} - x_{10}^{(2)}}{y_1^{(2)}}.$$

Layer 2: In order to recover the interior resistors set up and solve the linear system

$$\begin{bmatrix} x_8^{(1)} & x_8^{(2)} \\ 1 & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

for a_1 and a_2 and calculate

$$R_4 = -\frac{a_1 x_5^{(1)} + a_2 x_5^{(2)}}{a_1 y_1^{(1)} + a_2 y_1^{(2)}} \quad \text{and} \quad R_7 = \frac{a_1 x_5^{(1)} + a_2 x_5^{(2)}}{a_2 y_1^{(2)}}.$$

▶ Lab Instructions

Follow the steps laid down above and record your findings on the next page.

Name: __________________ **Date:** __________________

Circuits II – Results

Record all currents in amps and all potentials in volts.

Layer 1: $y_2 = 17.03 \times 10^{-3}$

$V_+ = 1.77$ $\qquad$ $V_- =$ __________

$R_2 = V_+/y_2 =$ __________ $\qquad$ true $R_2 =$ __________

$R_5 = -V_-/y_2 =$ __________ $\qquad$ true $R_5 =$ __________

Layer 2$^{(1)}$: $x_1^{(1)} =$ __________ $\qquad$ $x_5^{(1)} =$ __________

$x_6^{(1)} =$ __________ $\qquad$ $x_8^{(1)} =$ __________

$x_9^{(1)} =$ __________ $\qquad$ $x_{10}^{(1)} =$ __________

$y_1^{(1)} =$ __________

$R_1 = x_1^{(1)}/y_1^{(1)} =$ __________ $\qquad$ true $R_1 =$ __________

Layer 2$^{(2)}$: $x_1^{(2)} =$ __________ $\qquad$ $x_5^{(2)} =$ __________

$x_6^{(2)} =$ __________ $\qquad$ $x_8^{(2)} =$ __________

$x_9^{(2)} =$ __________ $\qquad$ $x_{10}^{(2)} =$ __________

$y_1^{(2)} =$ __________

$R_{10} = (x_9^{(2)} - x_{10}^{(2)})/y_1^{(2)} =$ __________ $\qquad$ true $R_{10} =$ __________

Layer 2:

$$A = \begin{bmatrix} x_8^{(1)} & x_8^{(2)} \\ 1 & 1 \end{bmatrix}, \qquad b = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \qquad a = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = A\backslash b = \begin{bmatrix} \underline{\qquad\qquad} \\ \underline{\qquad\qquad} \end{bmatrix}$$

$$R_7 = \frac{a_1 x_5^{(1)} + a_2 x_5^{(2)}}{a_2 y_1^{(2)}} = \underline{\qquad\qquad} \qquad \text{true } R_7 = \underline{\qquad\qquad}$$

$$R_4 = -\frac{a_1 x_5^{(1)} + a_2 x_5^{(2)}}{a_1 y_1^{(1)} + a_2 y_1^{(2)}} = \underline{\qquad\qquad} \qquad \text{true } R_4 = \underline{\qquad\qquad}$$

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.

▶ 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 two different stiffnesses: identical horizontal and vertical springs of rest length 1.25 inches, and diagonal springs of length 1.75 inches (notice that $1.75 \approx 1.25\sqrt{2} = 1.7677\ldots$).

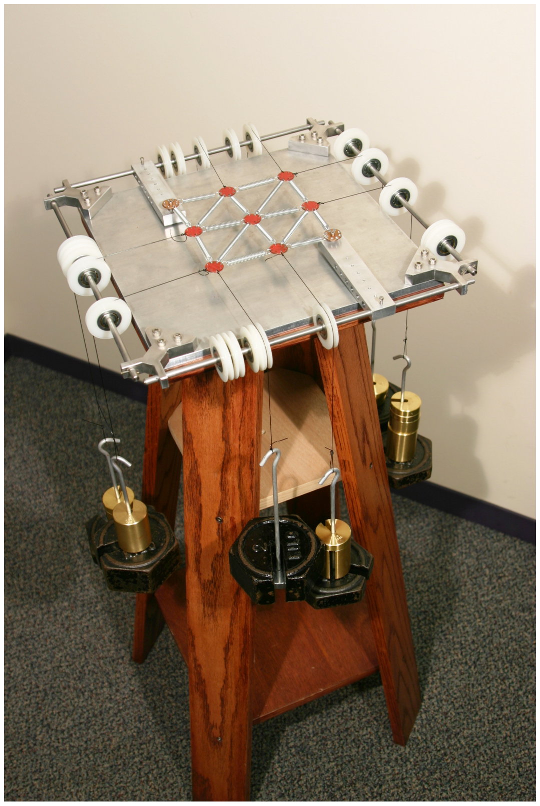


Figure 3.1: Force table and spring network.

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., a large paper clip anchored at the opposite end) and the other to string that runs over a pulley and attaches to a mass hook hung over the side of the table. Fix a metric ruler to the table near this latter end of the spring. You will then place brass masses on

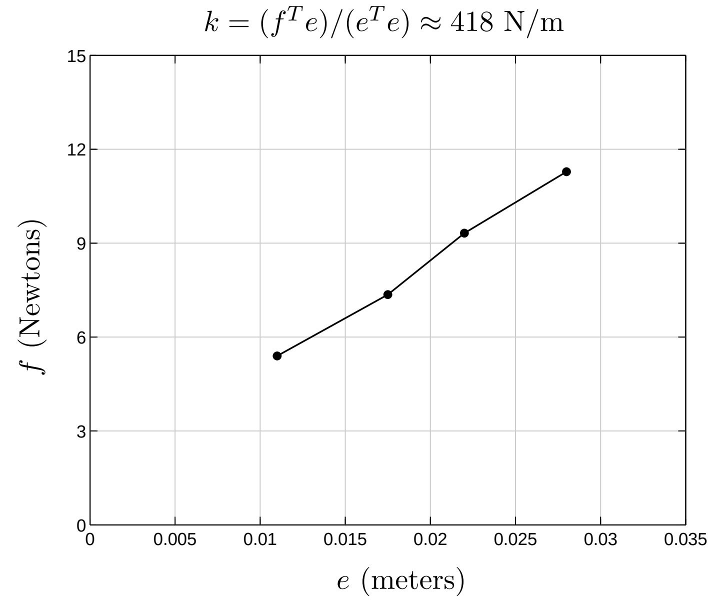


Figure 3.2: Computing a spring constant.

the hook and measure the associated elongation, in units of meters (m). 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 $g = 9.81\,\mathrm{m/s}^2$, in units of Newtons ($\mathrm{N} = \mathrm{kg\,m/s}^2$). (We recommend that you begin with a single 0.5 kg mass and increase from there; the hook has mass 0.05 kg.) Hooke's law informs us that $f_j = ke_j$, 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 (ke_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

$$M(k) \equiv \frac{1}{2}\sum_{j=1}^{4}(f_j - ke_j)^2 \tag{3.1}$$

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

$$k = \frac{e^T f}{e^T e}. \tag{3.2}$$

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

$$m = \begin{bmatrix} 0.55 \\ 0.75 \\ 0.95 \\ 1.15 \end{bmatrix} \mathrm{kg}, \qquad f = \begin{bmatrix} 5.3955 \\ 7.3575 \\ 9.3195 \\ 10.3005 \end{bmatrix} \mathrm{N}, \qquad e = \begin{bmatrix} 0.011 \\ 0.0175 \\ 0.022 \\ 0.028 \end{bmatrix} \mathrm{m},$$

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

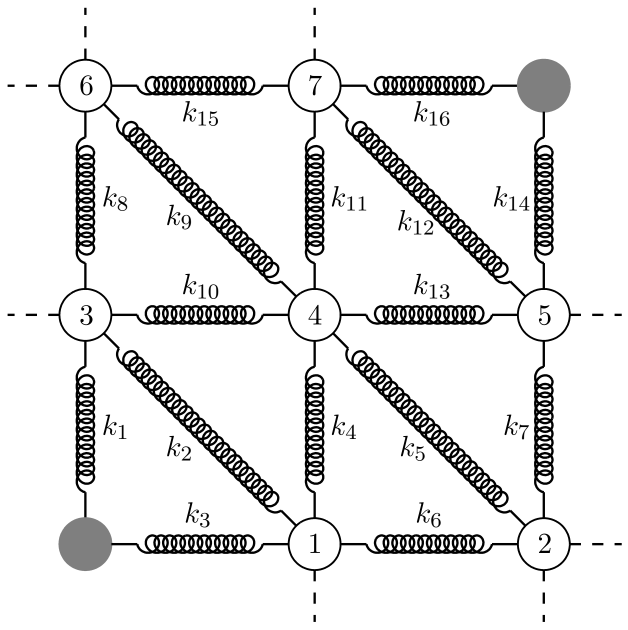


Figure 3.3: The spring network for this lab. The dashed lines indicate wires from which masses are to be attached; the gray nodes are tied in a fixed position to anchor the network. Two types of springs should be used in this lab: one for all horizontal and vertical members, another for all those on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

▶ 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^TKA 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 1.25 inch springs for the horizontal and vertical components, and the 1.75 inch 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 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.

For the control configuration, suspend 0.5 kg masses from each 0.05 kg mass hook. Under this load $f^{(1)}$, the masses 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

$$\widetilde{p}^{(2)} = p^{(1)} + x^{(2)} - x^{(1)}. \tag{3.8}$$

Our goal is to compare this prediction $\widetilde{p}^{(2)}$ 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 centered above the force table. The following command will open up a preview window in MATLAB of the table surface:

```
vid = videoinput('winvideo',1,'IYUV_640x480');
preview(vid);
```

You may need to focus or adjust lighting. To focus, turn the orange cone on the camera. 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

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

This function requires you to click on the `N` different pennies in the image, and returns a vector `p` with 2`N` 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, clear the force table except for two red pennies placed directly on the surface, separated by a known distance (say, 10 cm = 0.1 m apart). Now run `getloc` with `N=2` to locate these two pennies. The pixel distance between the pennies is thus

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

giving the conversion factor from pixels to meters of

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

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

```
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. 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.

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.

4. Conduct the control and variable experiment with the sixteen-spring network. You may wish to perform a handful of experiments with different loads. 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 each experiment, 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 $\widetilde{p}^{(2)}$, 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

   ```
   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

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

▶ Epilogue: The Subtlety of the Null Space

Suppose we now allow the lower-left node in Figure 3.3 to move freely in both directions, leading to

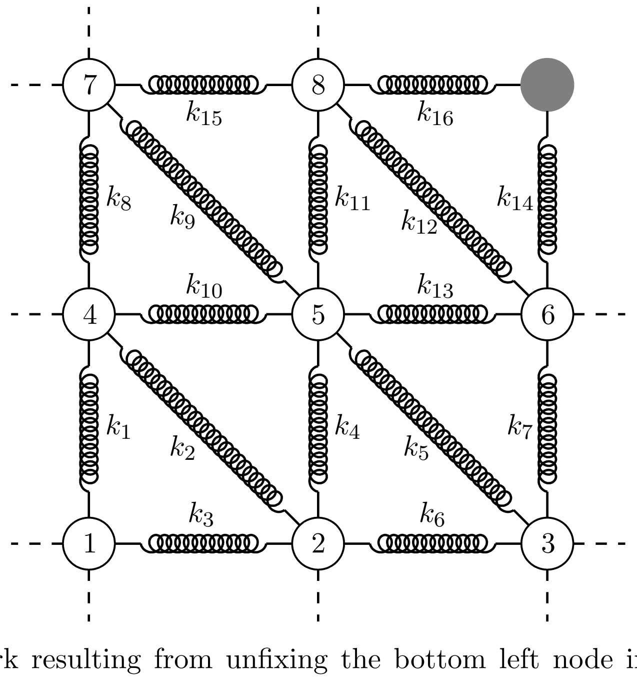


Figure 3.4: The network resulting from unfixing the bottom left node in Figure 3.3. This new network has 16 springs and 16 degrees of freedom, but $\mathcal{N}(A)$ is nontrivial: it corresponds to infinitesimal rotations about the fixed node, shown in gray.

the configuration in Figure 3.4. Since this new network has eight free nodes, there are now sixteen degrees of freedom. As there are also sixteen springs, $A \in \mathbb{R}^{16\times 16}$, and we might expect this square matrix to be invertible, and hence have a trivial null space. Perhaps you might initially struggle to spot any instabilities – ways of displacing the eight nodes without stretching any springs. For example, the fixed upper-right node prevents such a stretch-free shifting of all nodes uniformly in either the horizontal or vertical directions. However, we can still rotate the entire structure about that one fixed node, and indeed such displacements characterize the null space:

$$\mathcal{N}(A) = \text{span}\{[2,2,2,1,2,0,1,2,1,1,1,0,0,2,0,1]^T\}.$$

Lab 4: Springs II

▶ Introduction

The first spring network laboratory focused on the *forward problem* of predicting the displacements that will result from a given load, presuming we know the material properties of the springs. Now we turn our attention to an *inverse problem.* Suppose we apply a variety of loads to the network, and measure the displacements induced by these loads. What can such experiments reveal about the material properties of the springs? We might imagine that we are conducting a biopsy, looking for some hidden flaw in a material we are not permitted to dissect or otherwise mangle.

▶ The Biaxial Truss, Revisited

We shall focus on a modest alteration to the network proposed in Figure 3.3 of the last lab. In that lab, all the springs were rather elastic; now replace one of the diagonal springs, say the ninth one, with a very stiff spring of the same unstretched length, as reflected in Figure 4.1. Can we identify the stiff spring using only force and displacement data?

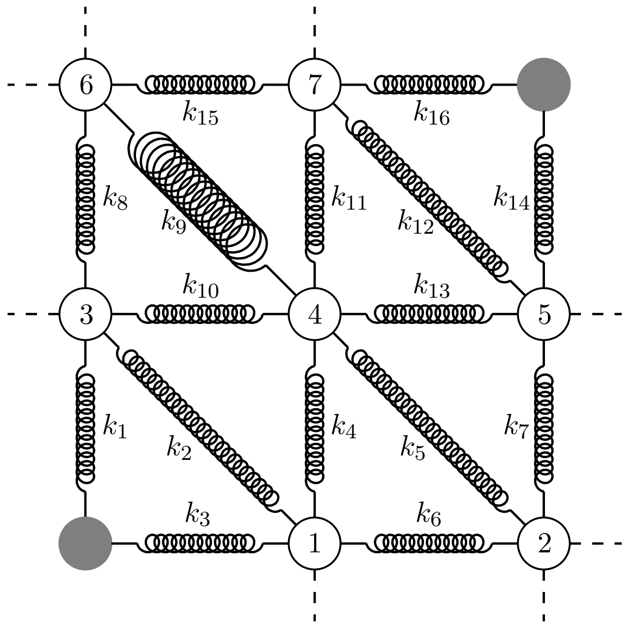


Figure 4.1: A repetition of the network in Figure 3.3, but here with the ninth spring replaced by one of significantly greater stiffness.

▶ The Inverse Problem

To extract the spring constants from the data we are permitted to measure, we shall follow the methodology described in Section 5.3 of the course notes. Need to sort out displacements attributable to the net force.

There is one small alteration to the setting in the notes, caused by our inability to reliably measure the position of the unloaded network. As in the last lab, we shall impose an initial control force, which we here call $f^{(0)}$, that leads to the positions recorded in the vector $p^{(0)}$. We shall then conduct a series of experiments with loads

$$f^{(1)}, \quad f^{(2)}, \quad \ldots, \quad f^{(s)}$$

and corresponding positions

$$p^{(1)}, \quad p^{(2)}, \quad \ldots, \quad p^{(s)}$$

for some modest value of $s \geq 2$. For these positions we can determine the net displacements from the control, $x^{(j)} - x^{(0)}$, which satisfy

$$x^{(j)} - x^{(0)} = p^{(j)} - p^{(0)}, \quad j = 1, \ldots, s. \tag{4.1}$$

As we saw in the last lab, these displacements should also satisfy the mathematical model

$$A^T K A (x^{(j)} - x^{(0)}) = f^{(j)} - f^{(0)}, \quad j = 1, \ldots, s, \tag{4.2}$$

where the adjacency matrix $A \in \mathbb{R}^{m \times n}$ with $m = n = 16$ for the network in Figure 4.1. In this present lab, we treat the matrix $K \in \mathbb{R}^{m \times m}$ containing the spring constants as unknown. Notice that, for a generic x, we can write

$$KAx = \begin{bmatrix} k_1 & & \\ & \ddots & \\ & & k_m \end{bmatrix} \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} k_1 \sum_{j=1}^n a_{1j} x_j \\ \vdots \\ k_m \sum_{j=1}^n a_{mj} x_j \end{bmatrix} = \texttt{diag}(Ax) \begin{bmatrix} k_1 \\ \vdots \\ k_m \end{bmatrix},$$

where $\texttt{diag}(Ax)$ is the $m \times m$ matrix obtained by putting the entries of Ax along the diagonal:

$$\texttt{diag}(Ax) = \begin{bmatrix} \sum_{j=1}^n a_{1j} x_j & & \\ & \ddots & \\ & & \sum_{j=1}^n a_{mj} x_j \end{bmatrix},$$

with all unspecified entries equal to zero.

With this observation, we can rewrite (4.2) as

$$A^T \texttt{diag}(A(x^{(j)} - x^{(0)}))k = f^{(j)} - f^{(0)}. \tag{4.3}$$

For the present network, the matrix $A^T\texttt{diag}(A(x^{(j)} - x^{(0)}))$ is a square 16×16 matrix that will almost always be invertible. Still, we expect the *measured* values of $x^{(j)} - x^{(0)}$ from (4.1) to be polluted with experimental error (not to mention approximations inherent in the mathematical model). For this reason, we choose not to solve (4.3) directly, but rather to accumulate data from numerous experiments and then solve a least squares problem. In particular, form the matrix and vector

$$B = \begin{bmatrix} A^T\text{diag}(A(x^{(1)} - x^{(0)})) \\ A^T\text{diag}(A(x^{(2)} - x^{(0)})) \\ \vdots \\ A^T\text{diag}(A(x^{(s)} - x^{(0)})) \end{bmatrix} \in \mathbb{R}^{sn\times m}, \quad f = \begin{bmatrix} f^{(1)} - f^{(0)} \\ f^{(2)} - f^{(0)} \\ \vdots \\ f^{(s)} - f^{(0)} \end{bmatrix} \in \mathbb{R}^{sn}, \tag{4.4}$$

and then determine the k by solving the least squares problem

$$\min_{k\in\mathbb{R}^m} \|Bk - f\|,$$

which you can solve via the normal equations

$$k = (B^TB)^{-1}B^Tf.$$

▶ Hardware and Software Notes

This lab asks you conduct the experiment we have just described using the spring network from Lab 3; the software tools are identical to those you used in the last lab.

- Use the `getloc` routine from the `C:\CAAMlab\Scripts` directory (which can be added to your MATLAB path via `addpath C:\CAAMlab\Scripts`).
- You will need to use `getloc` to calibrate webcam, i.e., to compute the pixel-to-meter conversion ratio.

▶ Tasks

- Calibrate the camera, compute the displacement $p^{(0)}$ for a control load $f^{(0)}$ (e.g., 0.55 kg loaded onto each mass hook). Now conduct $s \geq 3$ experiments as described above to obtain positions $p^{(1)}, \ldots, p^{(s)}$.
- Set up the matrix B and vector f in (4.4), and solve the least squares problem to obtain an estimate to the vector of spring constants, k.
- Display your results as a bar chart using MATLAB's `bar` command. Ideally, you will see three prominent spring constants: one for the identical horizontal and vertical springs, one for the three identical diagonal springs, and one for the stiff diagonal spring. Given the noise in our measurements and approximations in our model, can you at least identify the stiff spring from the others?
- How do your results differ if you use data from only one experiment ($s = 1$)?

Lab 5: Frequency Identification via Fourier Transformation

▶ Introduction

Thus far all our labs have studied systems in *static equilibrium*; for the rest of the semester we shall study the rather more exciting world of *dynamics*, in which case we investigate the way a system evolves in time. Suppose we examine that system at a single point in space, such as a node in a circuit network or a point on a plucked guitar string, and measure a physical quantity like potential or displacement as a function of time,

$$f(t), \quad t \geq 0.$$

The forces and constraints controlling the systems we study shall typically induce *vibrations*, and thus we might expect $f(t)$ to be periodic in time. Let us call the period τ, so that

$$f(t+\tau) = f(t), \quad \text{for all } t \geq 0.$$

For example, the function in Figure 5.1 has period $\tau = 2\pi$.

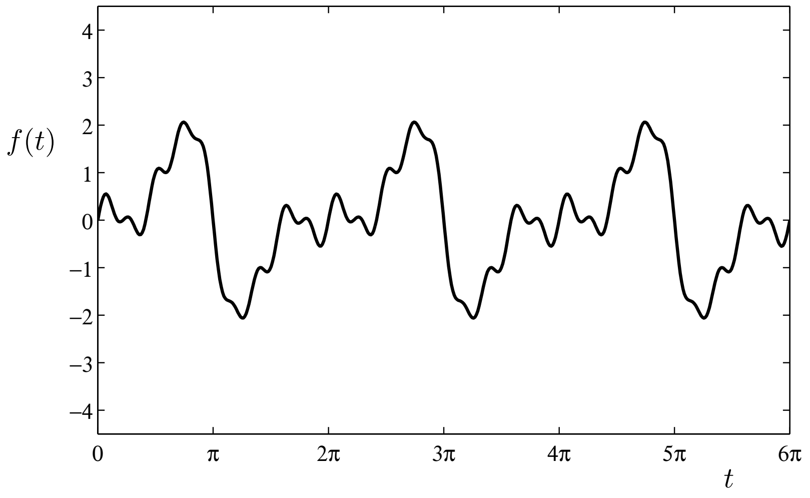


Figure 5.1: A function $f(t)$ with period $\tau = 2\pi$.

Notice that this function has components of various different time scales: some rapid vibrations appear to be superimposed upon a slower structure. This hints that we might be able to write f as a linear combination of trigonometric functions, each with a period that perfectly divides τ:

$$f(t) = a_0 + \sum_{k=1}^{\infty} a_k \cos\Big(\frac{2\pi k}{\tau}t\Big) + \sum_{k=1}^{\infty} b_k \sin\Big(\frac{2\pi k}{\tau}t\Big). \tag{5.1}$$

In the language of linear algebra, we are trying to write the periodic function f as a linear combination of 1, $\cos(2\pi t/\tau)$, $\sin(2\pi t/\tau)$, $\cos(4\pi t/\tau)$, $\sin(4\pi t/\tau)$, $\ldots$. Such an expansion is called a *Fourier series*, and you might find it remarkable that any τ-periodic function can be written in this form. We shall postpone the details, including how the coefficients a_k and b_k in (5.1) are found, to a future course, but we wish to stress this key concept: the size of those coefficients reveals how much each frequency $2\pi k/\tau$ contributes to the function f.

Our immediate interest is the discovery of a more compact way to write the Fourier series (5.1) that will ultimately prove extremely useful. The derivation requires a bit of *complex arithmetic*; we shall provide a crash course here, and you can also consult Chapter 7 of the CAAM 335 notes.

▶ Digression I: complex exponentials

Any *complex number* z can be written in the form $z = a + ib$, where $i = \sqrt{-1}$. The real number a is called the *real part of* z and the real number b is the *imaginary part of* z.

How might we extend familiar real-variable functions to work with complex variables? One approach is to plug the complex number z into the familiar Taylor series introduced in basic calculus. For example, if for real t we have

$$e^t = 1 + t + \tfrac{1}{2!}t^2 + \tfrac{1}{3!}t^3 + \cdots,$$

then for complex z we can *define*

$$e^z = 1 + z + \tfrac{1}{2!}z^2 + \tfrac{1}{3!}z^3 + \cdots.$$

What can we say about the exponential of a purely imaginary number? By unpacking the first few terms of the Taylor series, we shall discover a gorgeous pattern. Since $(ib)^2 = -b^2$, $(ib)^3 = -ib^3$, etc., we find that

$$\begin{aligned} e^{ib} &= 1 + ib + \tfrac{1}{2!}(ib)^2 + \tfrac{1}{3!}(ib)^3 + \cdots \\ &= 1 + ib - \tfrac{1}{2!}b^2 - \tfrac{1}{3!}ib^3 + \cdots \\ &= \Big(1 - \tfrac{1}{2!}b^2 + \tfrac{1}{4!}b^4 - \cdots\Big) + i\Big(b - \tfrac{1}{3!}b^3 + \tfrac{1}{5!}b^5 - \cdots\Big). \end{aligned}$$

In this last line we have separated the terms into those that are purely real and purely imaginary, and we see that each of these is a Taylor series in the *real variable* b. Do you recognize these series? They are simply Taylor expansions for $\cos(b)$ and $\sin(b)$. We have just derived *Euler's formula*:

$$e^{ib} = \cos(b) + i\sin(b).$$

▶ Complex Fourier series

Euler's formula allows us to write Fourier series in a compact manner, by replacing trigonometric functions with exponentials, at the cost of complex coefficients. For starters, note that

$$\begin{aligned}
(a+ib)e^{i\omega t} &= (a+ib)(\cos(\omega t)+i\sin(\omega t)) \\
&= (a\cos(\omega t)-b\sin(\omega t))+i(a\sin(\omega t)+b\cos(\omega t)) \\
(a-ib)e^{-i\omega t} &= (a-ib)(\cos(\omega t)-i\sin(\omega t)) \\
&= (a\cos(\omega t)-b\sin(\omega t))-i(a\sin(\omega t)+b\cos(\omega t)).
\end{aligned}$$

Adding these two expressions gives the identity

$$(a+ib)e^{i\omega t}+(a-ib)e^{-i\omega t} = 2a\cos(\omega t)-2b\sin(\omega t). \tag{5.2}$$

Thus we can write any linear combination of $\cos(\omega t)$ and $\sin(\omega t)$ as a realted linear combination of $e^{i\omega t}$ and $e^{-i\omega t}$. Using this new nomenclature, the trigonometric Fourier series

$$f(t) = a_0 + \sum_{k=1}^{\infty} a_k \cos\left(\frac{2\pi k}{\tau}t\right) + \sum_{k=1}^{\infty} b_k \sin\left(\frac{2\pi k}{\tau}t\right)$$

can now be written as

$$f(t) = \sum_{k=-\infty}^{\infty} c_k e^{i(2\pi k/\tau)t}.$$

The formula (5.2) gives a way to relate the new $c_{\pm k}$ coefficients to the old a_k and b_k coefficients:

$$c_{\pm k} = \begin{cases} (a_k \pm i b_k)/2, & \text{if } k \neq 0; \\ a_0, & \text{if } k = 0. \end{cases}$$

▶ Discrete Fourier transforms

In practice, we are unable to measure the function $f(t)$ at *all* t: we can only measure the function at distinct points in time. By sampling the function in this way, we are *discretizing* it, that is, converting a continuous quantity to an N-vector:

$$\mathbf{f} = \begin{bmatrix} f_0 \\ f_2 \\ \vdots \\ f_{N-1} \end{bmatrix},$$

where $f_k = f(t_k)$ for $k = 0, \ldots, N-1$. (Why go from 0 to $N-1$ instead of 1 to N? This makes the notation easier later.) Figure 5.2 shows one period of the function f from Figure 5.1 with samples at $N = 8$ points. We would still like to uncover the frequencies of vibration inherent in this function, even though we only know its value at certain points. Can we still determine the Fourier series?

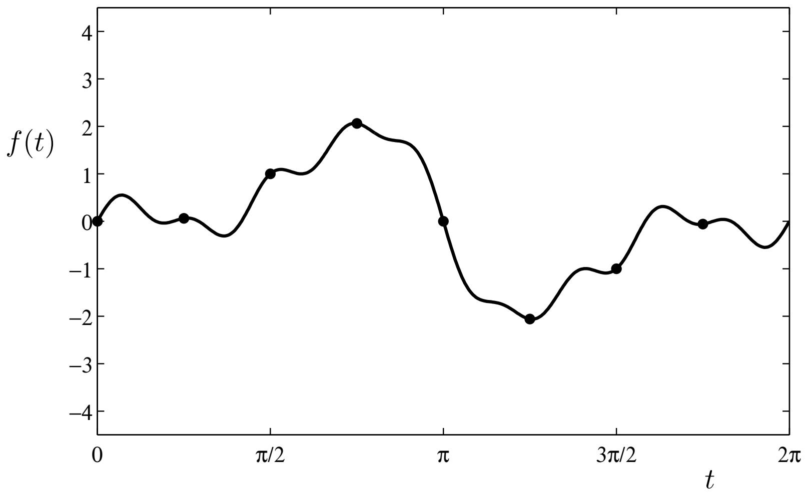


Figure 5.2: One period of the function $f(t)$ from Figure 5.2, with dots representing samples at $N = 8$ equally spaced points.

There is good news: we *can* still perform Fourier analysis on discrete data – but there is a significant caveat. As we only have data at N points, we should not expect to be able to uniquely determine the infinitely many Fourier coefficients. In fact, suppose g is any periodic function that happens to be zero at every point we sample: $g(t_k) = 0$ for $k = 0, \ldots, N-1$. In this case, the data we have cannot distinguish between $f(t)$ and $f(t) + g(t)$. Figure 5.3 illustrates this phenomenon, known as *aliasing*: the dark line shows $f(t)$ and the gray line illustrates $f(t) + g(t)$; both functions agree at the data points, $f_k = f(x_k) = f(x_k) + g(x_k)$. The problem becomes obvious when considered from the perspective of linear algebra: we are trying to determine infinitely many

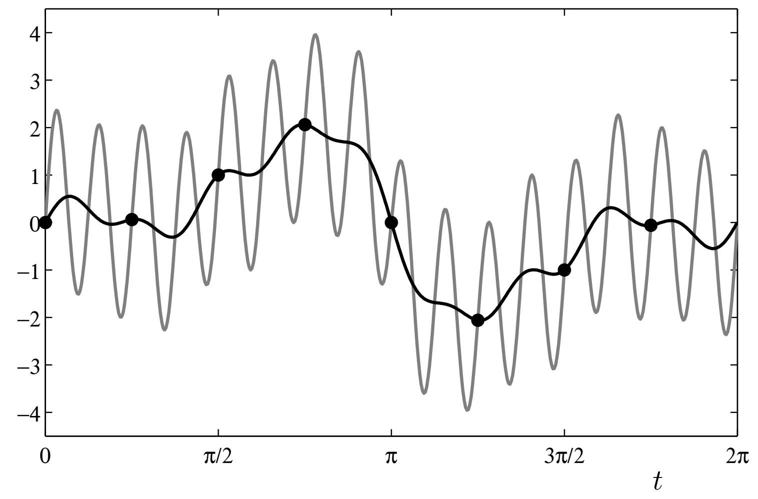


Figure 5.3: The functions $f(t)$ (dark line) and $f(t) + g(t)$ (gray line) are identical at the eight sample points (dots).

unknowns $c_0, c_{\pm 1}, c_{\pm 2}, \ldots$ in terms of only N equations, one for each of the f_ℓ, $\ell = 0, \ldots, N-1$. Clearly the solution to such an underdetermined system cannot be unique!

We only hope to recover N unknowns from N data points. Thus, we shall try to find coefficients for an N-term Fourier series that agrees with our data. That is, we seek $c_0, \ldots, c_{N-1}$ such that

$$\sum_{k=0}^{N-1} c_k e^{i(2\pi k/\tau)t_\ell} = f_\ell, \qquad \ell = 0, \ldots, N-1.$$

The c_k values here will generally differ from those in the infinite Fourier series, since the finite sum is only an approximation.

We presume that the data is measured at evenly spaced points starting with $t_0 = 0$; that is,

$$t_\ell = \ell\tau/N, \qquad\qquad \ell = 0, \ldots, N-1.$$

Substituting this into our sum, we find that

$$\sum_{k=0}^{N-1} c_k e^{2\pi i \ell k/N} = f_\ell, \qquad \ell = 0, \ldots, N-1.$$

Before we write this system in matrix form, we should make one more notational simplification. Define

$$\omega = e^{2\pi i/N},$$

which we call an *Nth root of unity* since $\omega^N = e^{2\pi i} = \cos(2\pi) + i\sin(2\pi) = 1$. With this shorthand, the finite Fourier series simply becomes

$$\sum_{k=0}^{N-1} c_k \omega^{\ell k} = f_\ell, \qquad \ell = 1, \ldots, N.$$

Let us look at the $N = 4$ case. We have the four equations:

$$\begin{aligned}
\ell = 0\text{:} \quad & c_0\omega^{0\cdot 0} + c_1\omega^{0\cdot 1} + c_2\omega^{0\cdot 2} + c_3\omega^{0\cdot 3} = f_0;\\
\ell = 1\text{:} \quad & c_0\omega^{1\cdot 0} + c_1\omega^{1\cdot 1} + c_2\omega^{1\cdot 2} + c_3\omega^{1\cdot 3} = f_1;\\
\ell = 2\text{:} \quad & c_0\omega^{2\cdot 0} + c_1\omega^{2\cdot 1} + c_2\omega^{2\cdot 2} + c_3\omega^{2\cdot 3} = f_2;\\
\ell = 3\text{:} \quad & c_0\omega^{3\cdot 0} + c_1\omega^{3\cdot 1} + c_2\omega^{3\cdot 2} + c_3\omega^{3\cdot 3} = f_3.
\end{aligned}$$

In matrix form, this becomes

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & \omega & \omega^2 & \omega^3 \\ 1 & \omega^2 & \omega^4 & \omega^6 \\ 1 & \omega^3 & \omega^6 & \omega^9 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \end{bmatrix}.$$

For general N, we have

$$\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{N-1} & \omega^{2(N-1)} & \cdots & \omega^{(N-1)^2} \end{bmatrix}$$

This matrix is called the *Fourier matrix of order N*; we shall denote it by $\mathbf{F}$. Recalling that matrix-vector multiplication amounts to taking a linear combination of matrix columns, we see that we are writing our function $\mathbf{f}$ in terms of the Fourier basis vectors

$$\begin{bmatrix} 1 \\ \omega^k \\ \omega^{2k} \\ \vdots \\ \omega^{(N-1)k} \end{bmatrix}, \quad k = 0, \ldots, N-1,$$

and the coefficients $c_0, \ldots, c_{N-1}$ tell us how strong our signal is in each of these basis vectors. The larger the value of k, the more oscillations in the corresponding sin and cos, and thus the 'higher the frequency'.

For $N = 4$, things look particularly clean: $\omega = e^{\pi i} = i$ (note that $i^4 = 1$), and hence we can also write the matrix equation as

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \end{bmatrix},$$

which we shall denote as

$$\mathbf{F}\mathbf{c} = \mathbf{f}. \tag{5.3}$$

Given the samples of $f(t)$ stored in the vector $\mathbf{f}$, we wish to find the coefficients $\mathbf{c}$ as rapidly as possible – we may have thousands, even millions, or samples! To understand how to invert $\mathbf{F}$ rapidly, we need to understand the linear algebra of complex matrices and vectors.

▶ Digression II: norms and orthogonality for complex vectors

If you have some vector with real entries, say,

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix},$$

then we measure its 'length', or, more properly, its *norm*, by

$$\|\mathbf{x}\| = \sqrt{\mathbf{x}^T\mathbf{x}} = \sqrt{(x_1)^2 + (x_2)^2}.$$

From this formula it should be clear that only the zero vector has zero length. For complex vectors, the situation requires more care. For example, if

$$\mathbf{x} = \begin{bmatrix} i \\ 1 \end{bmatrix},$$

then

$$\mathbf{x}^T\mathbf{x} = \begin{bmatrix} i & 1 \end{bmatrix} \begin{bmatrix} i \\ 1 \end{bmatrix} = i^2 + 1 = 0,$$

so our conventional definition suggests that this nonzero vector has norm zero. Hence we need to refine the notion of length for complex vectors. If $z = a + ib$ is a complex scalar, then we can define its *magnitude* to be $|z| = \sqrt{a^2+b^2}$, and this is always a positive quantity when $z \neq 0$. We also define the *conjugate* of z to be

$$\overline{z} = a - ib,$$

i.e., we just flip the sign on the imaginary part. The conjugate provides a slick way to define the magnitude:

$$|z| = \sqrt{\overline{z}z}.$$

Now we are ready to define the *norm of a complex vector*:

$$\|\mathbf{x}\| = \sqrt{|x_1|^2 + |x_2|^2 + \cdots |x_N|^2} = \sqrt{\mathbf{x}^*\mathbf{x}},$$

where $\mathbf{x}^* = [\, \overline{x}_1 \quad \overline{x}_2 \quad \cdots \quad \overline{x}_N \,]$ is called the *conjugate-transpose* of the vector $\mathbf{x}$.

The conjugate-transpose also allows us generalize the familiar dot product to complex vectors: the *inner product* of complex vectors $\mathbf{x}$ and $\mathbf{y}$ is the scalar $\mathbf{x}^*\mathbf{y}$. Genearlizing from the real case, we say that $\mathbf{x}$ and $\mathbf{y}$ are orthogonal if

$$\mathbf{x}^*\mathbf{y} = 0.$$

This digression will allow us to observe a fundamental property of the Fourier matrix.

▶ Inverting the Fourier matrix

If we look at the Fourier matrix for $N = 4$ displayed above, we observe a remarkable fact. Take the inner product of the first and second columns:

$$\begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ i \\ -1 \\ -i \end{bmatrix} = 1 + i - 1 - i = 0.$$

Thus the first two columns are orthogonal. In fact, *all columns of the Fourier matrix are orthogonal to one another* (try it!), and this remains true for any value of N. The norm of each column is also easy: each is precisely $\sqrt{N}$, as the inner product of any column with itself is N. We can summarize this information in the matrix $\mathbf{F}^*\mathbf{F}$, whose (j,k) entry is the inner product of the j and k columns of $\mathbf{F}$. Our previous observations tell us that, in the $N = 4$ case,

$$\mathbf{F}^*\mathbf{F} = \begin{bmatrix} 4 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 \\ 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 4 \end{bmatrix},$$

and for general N,

$$\mathbf{F}^*\mathbf{F} = N\mathbf{I},$$

where $\mathbf{I}$ is the identity matrix. Do you see that $\mathbf{F}^*$ must thus be related to the inverse of $\mathbf{F}$? Multiply each side of this last equation on the right by $\mathbf{F}^{-1}$ and divide by N to conclude

$$\mathbf{F}^{-1} = \frac{1}{N}\mathbf{F}^*.$$

To solve the linear system

$$\mathbf{F}\mathbf{c} = \mathbf{f}$$

for the coefficients $c_0, \ldots, c_{N-1}$, we need only premultiply both sides by the conjugate-transpose of the Fourier matrix, and scale:

$$\mathbf{c} = \frac{1}{N}\mathbf{F}^*\mathbf{f}.$$

This operation is called the *Fourier transform* (or *DFT*, for Discrete Fourier Transform) of $\mathbf{f}$.

With this technique, we can transform back and forth between the 'time domain' (the vector $\mathbf{f}$, which contains samples of the function f at fixed times) and the 'frequency domain' (the vector $\mathbf{c}$, which is the representation of $\mathbf{f}$ in the Fourier basis) via multiplication by the Fourier matrix and its conjugate transpose. The number of basic arithmetic operations required to compute a generic matrix-vector product grows inproportion to N^2; we say that matrix-vector multiplication is an $O(N^2)$ algorithm. If the columns of $\mathbf{F}$ were not orthogonal, we would have to invert this matrix using Gaussian elimination to solve (5.3), an operation that would require $O(N^3)$ operations. Actually, we have it even better: In 1965 Cooley and Tukey rediscovered and popularized something that was known to Gauss some 160 years earlier: the special structure of the Fourier matrix allows matrix-vector multiplication with $\mathbf{F}$ and $\mathbf{F}^*$ to be performed in $O(N \log N)$ operations. The resulting algorithm, *fast Fourier transform* (FFT), makes it possible to efficiently process signals with very large values of N. MATLAB incorporates a particularly efficient implementation known as FFTW (the Fastest Fourier Transform in the West).

▶ A nagging little question

It is about time we admit to twisting your arm a bit back on page 31, when we truncated the infinite Fourier series

$$\sum_{k=-\infty}^{\infty} c_k e^{i(2\pi k/\tau)t}$$

to the finite series

$$\sum_{k=0}^{N-1} c_k e^{i(2\pi k/\tau)t}.$$

Did you wonder why we chose the limits $k = 0, \ldots, N$ on this sum, rather than something like $k = -N, \ldots, N$? After all, when we first wrote down the infinite series, we saw that we would need both positive and negative k to represent real-valued functions f (recall that $c_k = \overline{c_{-k}} = (a_k + ib_k)/2$ on page 3). We took nonnegative k values in our finite sum to respect time-honored conventions; we shall now see that nothing has been lost, and the reason boils down to aliasing.

Each row of the formula $\mathbf{Fc} = \mathbf{f}$ that determines the coefficients $c_0, \ldots, c_N$ takes the form

$$\sum_{k=0}^{N-1} c_k \omega^{\ell k} = f_\ell, \qquad \ell = 0, \ldots, N-1.$$

It will prove convenient to look at an odd value of N; let us take $N = 5$ for a concrete example. This gives the linear system

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \omega^4 \\ 1 & \omega^2 & \omega^4 & \omega^6 & \omega^8 \\ 1 & \omega^3 & \omega^6 & \omega^9 & \omega^{12} \\ 1 & \omega^4 & \omega^8 & \omega^{12} & \omega^{16} \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix} = \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix}.$$

For odd N, define $n = (N-1)/2$, so for the present example $n = 2$. Suppose we now write out a finite Fourier series that has N terms, but now running from $k = -n$ to n:

$$\sum_{k=-n}^{n} \gamma_k \omega^{\ell k} = f_\ell, \qquad \ell = 0, \ldots, N-1.$$

(We use constants γ_k to distinguish them from the c_k values obtained from the $k = 0, \ldots, N$ sum.) In the $N = 5$ case, these N equations in the N unknowns $\gamma_{-n}, \ldots, \gamma_n$ give the linear system

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ \omega^{-2} & \omega^{-1} & 1 & \omega & \omega^2 \\ \omega^{-4} & \omega^{-2} & 1 & \omega^2 & \omega^4 \\ \omega^{-6} & \omega^{-3} & 1 & \omega^3 & \omega^6 \\ \omega^{-8} & \omega^{-4} & 1 & \omega^4 & \omega^8 \end{bmatrix} \begin{bmatrix} \gamma_{-2} \\ \gamma_{-1} \\ \gamma_0 \\ \gamma_1 \\ \gamma_2 \end{bmatrix} = \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix}.$$

This system so closely resembles the old one (the last three columns of the new matrix match the first three columns of the old), that we have to wonder if there is some way to resolve the modest differences. Recall that ω was constructed so that $\omega^N = 1$. It thus follows that $\omega^{-\ell} = \omega^N \omega^{-\ell} = \omega^{N-\ell}$. Similarly, for *any integer m* we have

$$\omega^{-\ell} = \omega^{mN-\ell}.$$

This enables us to observe, in the $N = 5$ case, that

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ \omega^{-2} & \omega^{-1} & 1 & \omega & \omega^2 \\ \omega^{-4} & \omega^{-2} & 1 & \omega^2 & \omega^4 \\ \omega^{-6} & \omega^{-3} & 1 & \omega^3 & \omega^6 \\ \omega^{-8} & \omega^{-4} & 1 & \omega^4 & \omega^8 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ \omega^{5-2} & \omega^{5-1} & 1 & \omega & \omega^2 \\ \omega^{10-4} & \omega^{10-2} & 1 & \omega^2 & \omega^4 \\ \omega^{15-6} & \omega^{15-3} & 1 & \omega^3 & \omega^6 \\ \omega^{20-8} & \omega^{20-4} & 1 & \omega^4 & \omega^8 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ \omega^{3} & \omega^{4} & 1 & \omega & \omega^2 \\ \omega^{6} & \omega^{8} & 1 & \omega^2 & \omega^4 \\ \omega^{9} & \omega^{12} & 1 & \omega^3 & \omega^6 \\ \omega^{12} & \omega^{16} & 1 & \omega^4 & \omega^8 \end{bmatrix}.$$

Notice that this last matrix has the same columns as our old $k = 0, \dots, N$ matrix, but in a different order. We obtain that old matrix for our new system by reordering the unknowns:

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \omega^4 \\ 1 & \omega^2 & \omega^4 & \omega^6 & \omega^8 \\ 1 & \omega^3 & \omega^6 & \omega^9 & \omega^{12} \\ 1 & \omega^4 & \omega^8 & \omega^{12} & \omega^{16} \end{bmatrix} \begin{bmatrix} \gamma_0 \\ \gamma_1 \\ \gamma_2 \\ \gamma_{-1} \\ \gamma_{-2} \end{bmatrix} = \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix}.$$

We arrive at the beautiful conclusion that $\gamma_0 = c_0$, $\gamma_1 = c_1$, $\gamma_2 = c_2$, $\gamma_{-1} = c_3$ and $\gamma_{-2} = c_4$. In general,

$$\gamma_k = \begin{cases} c_k & \text{if } 0 \le k \le n; \\ c_{n-k} & \text{if } -n \le k < 0. \end{cases}$$

Thus we have neither gained nor lost anything by working with the $k = 0 \dots, N$ sum rather than the $k = -n, \dots, n$ sum: the only difference is the order of the coefficients. The convention is to use the $k = 0, \dots, N$ sum, owing to the high elegance of the $\mathbf{F}$ matrix. (For example, MATLAB's `fft` command uses this ordering.) To work with the more natural $-n \dots, n$ sum, we simply rearrange the coefficients.

In fact, we need not make things so complicated. For real data, $f_0, \dots, f_N$, we will always have

$$\gamma_k = \overline{\gamma_{-k}}$$

just as for the infinite Fourier series. Thus we get full knowledge by only examining γ_k for $k \ge 0$, in which case $\gamma_k = c_k$. In many applications we only care about the 'amount of energy' in each frequency k, and thus we typically produce a plot of $|c_k|$ versus k for $k = 0, \dots, n$.

▶ Tasks: experiments with discrete Fourier analysis

The first three exercises will introduce you to the `fft` command in MATLAB. The fourth exercise involves analysis of the human voice.

1. Try `fft(eye(4))` and `ifft(eye(4))`. How do these matrices relate to $\mathbf{F}$ and $(1/N)\mathbf{F}^*$?

2. Consider the function

$$f(t) = 1 + \sin(t) + \tfrac{1}{2}\cos(t) + \sin(2t) + \cos(3t).$$

Let the vector `t` consist of `N` points uniformly spaced on $[0, 2\pi)$. For example,

```
t=linspace(0,2*pi,N+1); t=t(1:N);
```

In MATLAB, compute the `fft` of `f(t)` for some odd `N` of your choosing. Explain your results.

3. The `C:\CAAMlab\Scripts` directory on the lab PC contains a file called `mystery_f.p` that produces the function shown in Figure 5.1. You are unable to see the source for this routine, but you can call `mystery_f` just as if it were a usual m-file; e.g., `ft = mystery_f(t)` for a vector `t`. This function is a linear combination of trigonometric functions. Use `fft` to determine a formula for `mystery_f`.

4. What frequency is your 'ee'?

   Run MATLAB on the lab computer, add the `C:\CAAMlab\Scripts` directory to your MATLAB path. The `getSound` script allows you to record data from the lab microphone, using the PC's soundcard. For example, to record for 5 seconds, type

   ```
   [f,dt] = getSound(5);
   ```

   and hit the return key. (Be sure the microphone is turned on!)

   The vector `f` contains the amplitude of the sound at times $0, \mathtt{dt}, 2\mathtt{dt}, \ldots, 5 - \mathtt{dt}$.

   To replay your recording, type

   ```
   wavplay(f,dt);
   ```

   The goal of this exercise is to study the frequency fluctuations in the human voice as it pronounces the long-e ('ee') sound.

   Record your voice speaking the 'ee' sound, stored in the vector `f`. Take the `fft` of your `f`, and produce a `semilogy` plot of the magnitude of the Fourier coefficients γ_k, $k = 0, \ldots, n$, versus k (use the `abs` command to compute the magnitudes).

   Experiment with different recording lengths and judicious choices of `axis` to reveal the most prominent frequencies and fluctuations in your voice. Submit plots of the time domain(`f` versus `t`) in addition to your plots of the frequency domain (i.e., the `fft`).

   A note about scaling. The `fft` command assumes that your `f` is sampled over an interval of length 2π. If you record for T seconds, you will need to scale the horizontal axis by $2\pi/T$. For example,

   ```
   fftf = fft(f);
   semilogy([0:n]*(2*pi/T)), abs(fftf(1:n+1))/(2*n+1))
   ```

   [This problem, like much of this lab, is inspired by and adapted from Chapter 4 of Gilbert Strang's *Introduction to Applied Mathematics*, 1986.]

Lab 6: Discovering Frequencies in a Time Series

▶ Introduction

The last lab covered the basic techniques of Fourier analysis; now it is time to put these tools to use on data from a real application. All the remaining labs deal with *vibrations*, be they electrical fluctuations (Lab 7), motion of a swinging compound pendulum (Lab 8), or the oscillations of a string loaded with beads (Labs 9 and 10). The present short lab is designed to help you understand how to identify the 'peaks' that reveal the frequencies at which a system vibrates, and in the process to discover how sample rate and the duration of a sample affect the quality of the data. An understanding of such vibrations is fundamental throughout science and engineering, from the tuning of a radio to the design of buildings that will not resonate with the quaking earth.

To isolate our situation as much as possible, in this lab we will work exclusively with synthetic data that mimics the physical experiments you will be conduct in Lab 8. Consider a (massless) string suspended from a height, loaded with three objects of equal mass (750 g) that are separated at uniform intervals (25 cm) along the length of the string. At rest, these masses hang in a vertical line, as illustrated in left image of Figure 6.1. When displaced from this equilibrium, gravity causes the entire configuration to vibrate (forever, in principle, if there are no damping forces to brake the motion). We are interested in relatively small displacements from the vertical equilibrium, and in this regime we can *approximate* that each mass moves only in the horizontal direction. In a few weeks we shall derive a mathematical model for this motion, then measure the displacements with the help of a webcam. For the time being, we shall be content to use a black-box routine (called `pendulum.p`) to simulate the motion of this three-mass pendulum. Snapshots of the motion

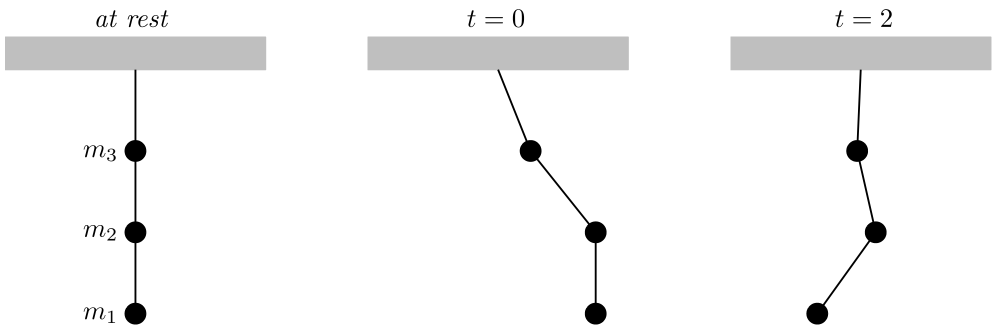


Figure 6.1: A compound pendulum with three masses at rest (left), with an initial displacement (middle), and after two seconds of evolution.

are shown in Figure 6.1: the masses are initially displaced as in the middle image, and released at time $t = 0$. After two seconds, the three masses have swung into the position shown in the rightmost plot. These few still images poorly communicate the beauty of this motion; please run the `pendulum_movie` script to view the action it its full glory.

▶ Synthetic data generation

The data that drives the movie is generated with the `pendulum.p` routine, which you can find on the class website, or in the `C:\CAAMlab\Scripts` directory on the lab computer. (Remember to `addpath C:\CAAMlab\Scripts` for convenience.) This routine is called as

```
        x = pendulum(duration, samp_per_sec);
```

The variable `duration` specifies the length of the simulation (in seconds), and `samp_per_sec` specifies the number of samples to be recorded per second. The function returns a matrix `x` with three columns (one for each mass) and $1+$ `duration` $\times$ `samp_per_sec` rows (which includes a sample at time $t = 0$). In particular, `x(j,k)` records the horizontal displacement (in meters) of mass `k` at time $(j-1)\times$ `samp_per_sec` seconds.

The `pendulum` routine is compiled into a ‘p-file’, which can be run just like an m-file, but the guts are hidden. You will learn to program them for yourself from scratch in a few weeks, but for the purpose of this lab, it does not matter how this data is generated: think of `pendulum` as a black-box that records measurements from an experiment.

From `pendulum_movie`, perhaps you can see that the configuration vibrates in some roughly periodic manner. With a keen eye you might even observe that there seem to be several different

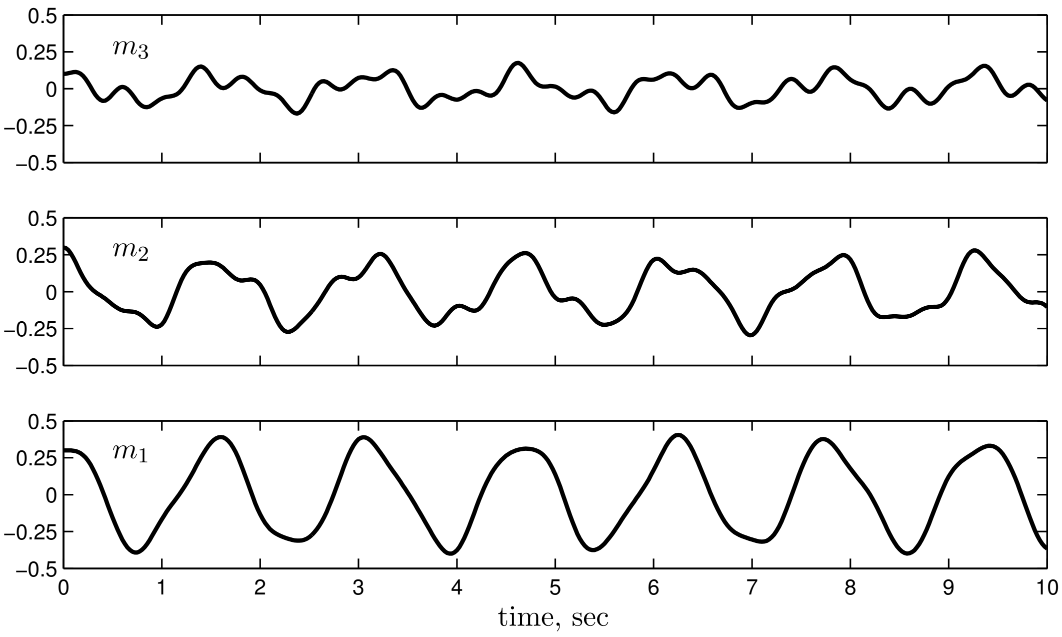


Figure 6.2: Horizontal displacements (meters) of the three masses over 10 seconds, 40 samples/sec.

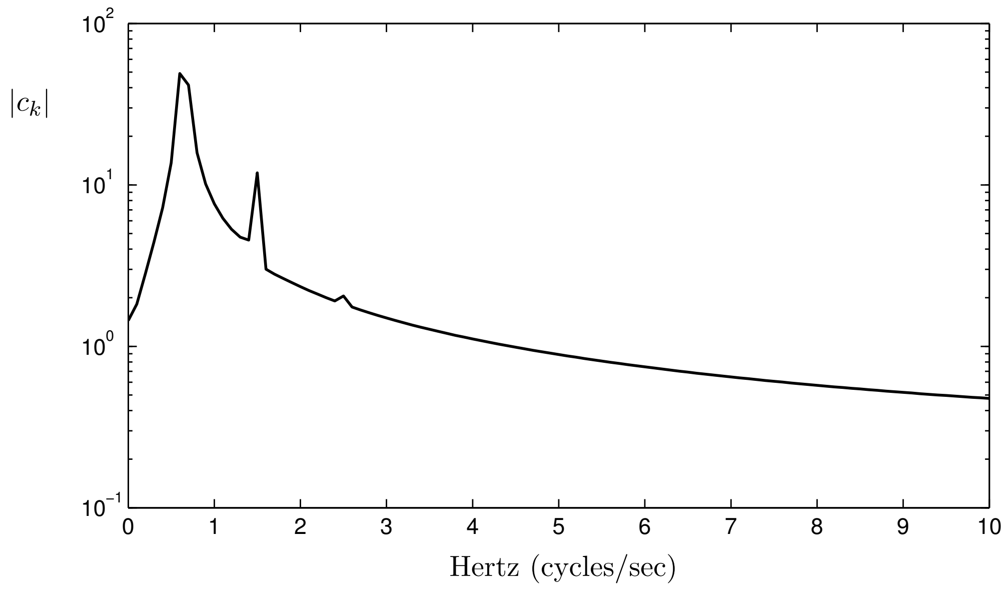


Figure 6.3: Fourier coefficients from the displacement of the first mass, based on 10 seconds of data, 40 samples/sec.

patterns that occur with varying frequency. This becomes more evident if we plot the displacements against time, as in Figure 6.2. One can detect multiple frequencies of vibration, particularly for the top two masses. The goal of this lab is to use the techniques of Fourier analysis that you mastered in Lab 5 to determine these frequencies to high precision.

Figure 6.3 shows the Fourier transformation of the data for mass m_1 that was given in Figure 6.2. You can obtain a similar plot with

```
semilogy([0:length(x(:,1))-1]/duration,abs(fft(x(:,1))))
```

and adjusting the axis to look at the leading Fourier coefficients. Scaling the horizontal axis in this way gives units of cycles per second. (Can you explain why? Recall that c_k is the coefficient that multiplies $e^{i(2\pi k/\tau)t}$ in the finite Fourier series, where τ represents the period.) You can verify that this scaling looks reasonable by noticing that the most prominent feature in the m_1 plot in Figure 6.2 repeats a little less than once per second, which corresponds to the dominant peak in Figure 6.3.

What is perhaps most surprising about Figure 6.3 is the appearance of two smaller peaks in this data, indicating that there could be features of the motion of m_1 that vibrate at a higher frequency. In the exercises, we want to explore these secondary peaks in greater detail, to see if they are legitimate, or merely an artifact of `duration` and `samp_per_sec`.

▶ Lab Instructions

The exercises for this lab are very basic, and do not require the use of any special lab equipment. The goal is for you to deeply understand how the `duration` and `samp_per_sec` parameters affect the accuracy with which you can determine the frequency of vibration in displacement data.

1. Produce Figure 6.3 for yourself, using `duration=10` and `samp_per_sec=40`. Now adjust the parameters `duration` and `samp_per_sec` independently. How does each of the parameters affect the shape of the peaks in the Fourier transform? In particular: If one wants to plot peaks over a large range of frequencies (cycles/sec), should one increase `duration` or `samp_per_sec`? What if one wants to determine sharp peaks over a fixed frequency band?

2. Submit several plots showing sharp peaks for the displacement of m_1. Write down the frequencies (in Hertz) accurate to several digits.

3. Repeat the experiment in the previous question for m_2 and m_3.

4. Our synthetic data is essentially perfect – it does not bear the scars of experimental error that mar real data. *Simulate* experimental error by polluting the measured displacement data. For example, a random error on the order of ten-percent can be produced via

   ```
   x_err = x+.10*randn(size(x))*max(max(abs(x)));
   ```

   Conduct experiments with various levels of error for mass m_3. What affect does this have on your frequency plots? (Please include a few examples.) At what error threshold does it become difficult to determine the three peaks in the data?

   (Such numerical investigations are very important in practice, for they suggest something about the robustness of your model, as well as the precision to which you will need to build you experimental apparatus.)

Lab 7: RLC Circuit Dynamics

The last two labs have provided you with a background in signal processing: given a function in time that is sampled at discrete points, identify the fundamental frequencies latent in that signal. These previous labs anticipate our study of *dynamical systems.* In this lab we begin that investigation by solving an ordinary differential equation that models a resistor–inductor–capacitor (RLC) circuit, and we see how this circuit reacts as we drive it at a single frequency.

▶ Capacitors and Inductors

The first two circuit labs involved only resistors and a voltage source. Now we shall add *capacitors* and *inductors* into the mix, as seen in the circuit diagram in Figure 7.1. Capacitance is measured in farads, inductance in henrys; like resistance, you can determine these quantities using the lab's multimeter.

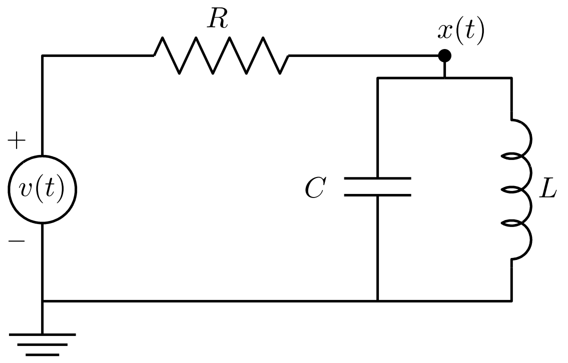


Figure 7.1: A simple resistor–inductor–capacitor (RLC) circuit.

The circuit is now driven by a potential source of $v(t)$ volts that can depend on time, and our goal is to describe how this source affects the unknown potential $x(t)$. (Note that this differs from the examples in the CAAM 335 notes, where $v(t)$ is replaced by a static *current source*, denoted i_0, and from the Circuits I and II labs, where we have the constant potential v_0 in place of $v(t)$.)

We wish to discover how the potential $x(t)$ is affected by the current and the circuit elements. This modest (but still interesting) circuit has the single unknown potential $x(t)$, so we are dealing in the realm of scalar quantities rather than vectors and matrices. This is the simplest setting to investigate these new electronic components, but the techniques generalize readily to far more sophisticated circuits.

The above circuit introduces a capacitor and an inductor in parallel. The current across a capacitor of capacitance C is given by the relation

$$\text{current} = (\text{capacitance}) \times (\text{time rate-of-change of potential}).$$

If $y(t)$ denotes the current and $e(t)$ the potential drop across the capacitor, then we write the capacitor law as

$$y(t) = Ce'(t).$$

An inductor of inductance L obeys a complementary law:

$$\text{potential} = (\text{inductance}) \times (\text{time rate-of-change of current}),$$

which we write in the form

$$e(t) = Ly'(t). \tag{7.1}$$

Notice that this equation involves $y'(t)$, while the analogous formulae for resistors and capacitors used $y(t)$. To allow us to combine expressions for all three electrical components, we shall transform $y'(t)$ in (7.1) into $y(t)$ with the help of the fundamental theorem of calculus, yielding

$$y(t) = \frac{1}{L}\int_0^t e(\tau)\,d\tau - y(0).$$

▶ Solution of a Simple RLC Circuit

We shall now see how to assemble these new elements into a differential equation, once again following the Strang Quartet methodology.

- Step 1: The potential drop across the resistor is given by $v(t) - x(t)$, which we label

  $$e(t) := v(t) - x(t).$$

  The potential drop across the parallel capacitor and inductor is $x(t) - 0$, so we do not introduce any new notation in this simple setting.

- Step 2: We apply the constitutive laws to compute the current across each component:

  $$\begin{aligned} \text{resistor:} &\quad y_{\text{res}}(t) = \frac{1}{R}e(t); \\ \text{capacitor:} &\quad y_{\text{cap}}(t) = Cx'(t); \\ \text{inductor:} &\quad y_{\text{ind}}(t) = \frac{1}{L}\int_0^t x(\tau)\,d\tau - y_{\text{ind}}(0). \end{aligned}$$

  (We have used x rather than e in the last two formulas, as $x(t)$ denotes the potential drop across the capacitor and inductor for the circuit in Figure 7.1. To analyze more complicated circuits we would introduce variables $e_1, e_2, \ldots$ for the potential drop across each of the components. Given the simplicity of the circuit at hand, we have opted for a leaner notation.)

- Step 3: Kirchhoff's Current Law at the node $x(t)$ implies that

  $$y_{\text{res}}(t) = y_{\text{cap}}(t) + y_{\text{ind}}(t),$$

  so incorporating the formulas from Step 2 gives the *integro-differential equation*

  $$\frac{1}{R}e(t) = Cx'(t) + \frac{1}{L}\int_0^t x(\tau)\,d\tau - y_{\text{ind}}(0). \tag{7.2}$$

- Step 4: As we wish to solve for the unknown potential $x(t)$, we substitute the definition of $e(t)$ into (7.2) to get

  $$\frac{1}{R}(v(t) - x(t)) = Cx'(t) + \frac{1}{L}\int_0^t x(\tau)\,d\tau - y_{\text{ind}}(0).$$

  We could take the Laplace transform of this equation directly (provided we know a rule for transforming integrals). Alternatively, we can take a derivative of the equation and rearrange to obtain

  $$x''(t) = -\frac{1}{LC}x(t) - \frac{1}{RC}x'(t) + \frac{1}{RC}v'(t),$$

  a *second-order* ordinary differential equation for $x(t)$. We could solve this by Laplace transform (provided we know a rule for transforming second derivatives), but we will instead use a little linear algebra. Notice that this second-order equation can be written in the form of a *first-order system* of equations

  $$\frac{d}{dt}\begin{bmatrix} x(t) \\ x'(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{1}{LC} & -\frac{1}{RC} \end{bmatrix}\begin{bmatrix} x(t) \\ x'(t) \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{RC}v'(t) \end{bmatrix}. \tag{7.3}$$

  The first row of (7.3) merely states a tautology:

  $$\frac{d}{dt}x(t) = x'(t).$$

  The second row captures the differential equation we are after:

  $$\frac{d}{dt}x'(t) = -\frac{1}{LC}x(t) - \frac{1}{RC}x'(t) + \frac{1}{RC}v'(t).$$

  The first-order system (7.3) has the general form

  $$z'(t) = Kz(t) + f(t),$$

  an equation familiar to you from Chapter 6 of the course notes; you will recall that such an equation can be solved via can be solved exactly via the Laplace transform, or approximately via the backward Euler method. To do so, we need an initial condition. The simplest approach is to take

  $$v(t) = 0 \quad \text{for all } t \le 0,$$

  which then suggests

  $$\begin{bmatrix} x(0) \\ x'(0) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

▶ Backward Euler method

In this lab we shall construct *approximate* solutions to the differential equation $z'(t) = Kz(t) + f(t)$ using the backward Euler method. This algorithm is described in the CAAM 335 course notes; we recap here for convenience.

Given a *time-step* $h > 0$, we seek solutions to the differential equations at times

$$t_k = kh.$$

Notice that we can write

$$\frac{z(t_k) - z(t_{k-1})}{h} \approx z'(t_k) = Kz(t_k) + f(t_k);$$

the approximation follows from the definition of the derivative, and the equality follows from the differential equation. To obtain an approximate solution to the differential equation, we simply replace $\approx$ with $=$. The values we obtain shall no longer be the exact solution to the differential equation, and we reflect this fact by replacing the notation $z(t_k)$ by z_k, and $z(t_{k-1})$ by z_{k-1}. In short, we have

$$\frac{z_k - z_{k-1}}{h} = Kz_k + f(t_k). \tag{7.4}$$

We are given

$$z_0 = z(0) = \begin{bmatrix} x(0) \\ x'(0) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix},$$

and we will use the formula (7.4) to step forward in time to z_1, then z_2, etc. To do so, we solve (7.4) for z_k. We find that

$$(I - hK)z_k = z_{k-1} + hf(t_k),$$

and hence

$$z_k = (I - hK)^{-1}(z_{k-1} + hf(t_k)). \tag{7.5}$$

From the first component of the vector $z_k \in \mathbb{R}^2$, we extract an approximation to $x(t_k)$.

▶ Band-Pass Filter

The circuit shown in Figure 7.1 is called a *band-pass filter*, because it responds sympathetically only to a particular range of input frequencies. More specifically, suppose we use the lab's function generator to produce an input potential

$$v(t) = \sin(\omega t)$$

for some frequency ω. For $\omega = 0$ there is obviously no input, and the initial condition $x(0) = x'(0) = 0$ ensures that nothing happens: $x(t) = 0$ for all time.

What can be said as we increase ω? As you can verify through experiments with the backward Euler method:

- $|x(t)|$ is consistently small for small ω values;
- $|x(t)|$ is large for an intermediate range of ω values, called the *band*;
- $|x(t)|$ is consistently small for large ω values.

[In fact, within the band there is a distinguished value of ω for which $v(t) = \sin(\omega t)$ passes through optimally. The explanation for this fact follows from *spectral theory*, a topic covered later in CAAM 335.]

▶ Lab Instructions

In this lab you will investigate this circuit through both numerical and physical experiments.

1. Develop a MATLAB implementation of the backward Euler method (7.5) for solving the RLC differential equation (7.3). Test it with the values $R = 1/2$, $L = C = 1$, and $v(t) = \sin(t)$ for $t \in [0, 50]$ with $x(0) = x'(0) = 0$, and compare your results exact solution

$$x(t) = \sin(t) - te^{-t}$$

   for several time-steps $h > 0$. (Turn in plots of $x(t)$ versus t, and plots of the difference between the true and backward Euler solutions versus time.)

2. We wish to investigate a bandpass filter with $R = 982\ \Omega$, $L = 5.30$ mH, and $C = 70.2$ nF. Before implementing such a circuit on a breadboard, we wish to simulate the predicted behavior with the backward Euler method. These values of L and C are somewhat extreme, and you need to be careful to integrate on the correct times scale. Integrate over $t \in [0, .001]$ sec. with appropriately small time-step h. Produce plots showing your backward Euler approximation to $x(t)$ versus t with $v(t) = \sin(\omega t)$ and $x(0) = x'(0) = 0$ for a range of values of

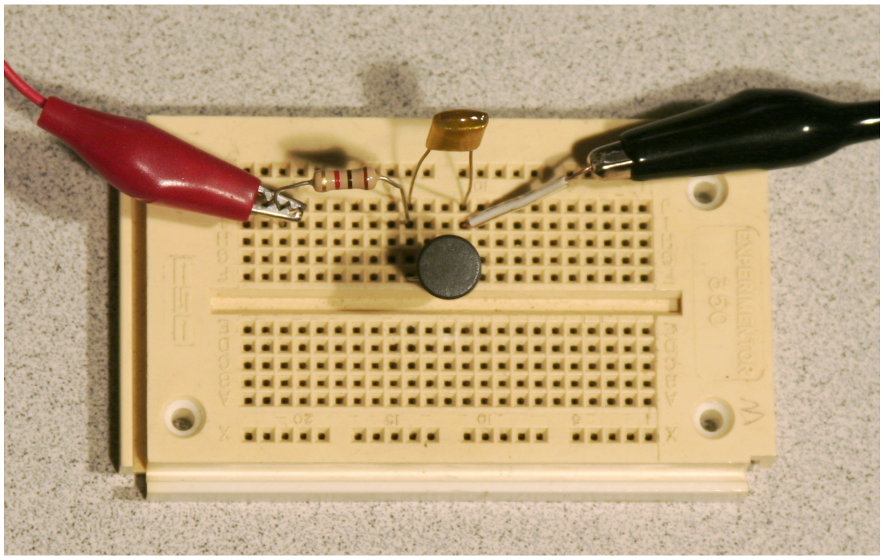


Figure 7.2: An implementation of the band-pass filter circuit from Figure 7.1 on a small breadboard.

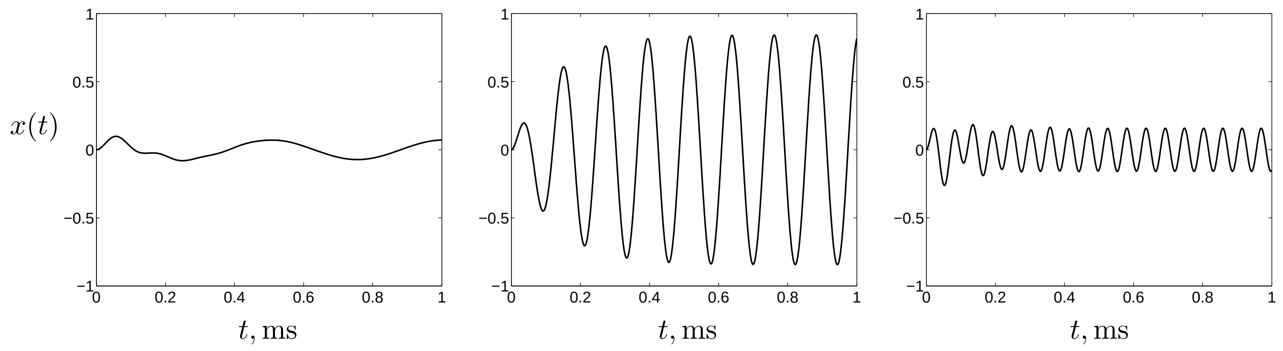


Figure 7.3: Backward Euler solutions to (7.3) with $R = 982\ \Omega$, $L = 5.30$ mH, and $C = 70.2$ nF driven by $v(t) = \sin(\omega t)$ for values of ω corresponding to 2000 Hz, 8200 Hz, and 18000 Hz.

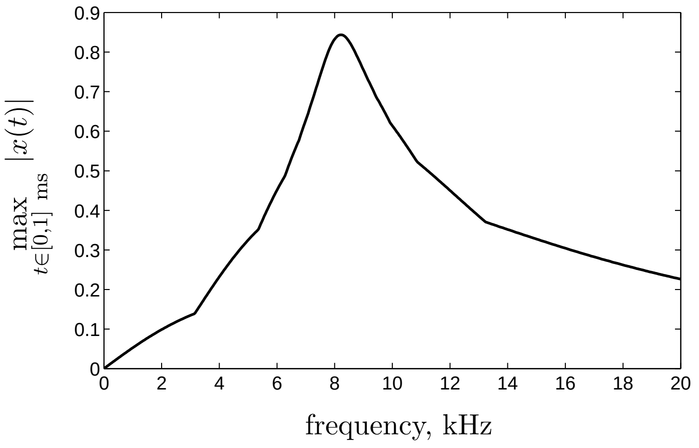


Figure 7.4: Maximum amplitude of the potential $x(t)$ over $t \in [0, 1]$ ms for $R = 982\ \Omega$, $L = 5.30$ mH, and $C = 70.2$ nF driven by $v(t) = \sin(\omega t)$ over a range of ω values ($\omega = 2\pi f$, where f denotes the frequency in Hertz).

$\omega \in [10^4, 10^5]$. (Notice that since Hertz denote cycles per second, $\sin(\omega t)$ corresponds to a frequency of $\omega/(2\pi)$ Hz.) Show plots for three values of ω: one below, one in, and above one the band; use the same `axis` in each figure to aid comparison. Your output should resemble Figure 7.3. You should observe the band-pass phenomenon described above; see also Figure 7.4.

How does the value of ω that you found in the band compare to the magnitude of the eigenvalues of K (`abs(eig(K))`)?

(As an alternative to working with these badly scaled values of C, L, and t, you may change the time-scale of the problem from seconds to milliseconds. This gives the values $R = 982$ V/A, $L = 5.3$ V · ms/A, and $C = 7.02 \times 10^5$ A · ms/V. The values of ω must also be scaled; say, take $\omega \in [0, 120]$. Now integrate over $t \in [0, 1]$ ms.)

3. Implement the band-pass filter circuit, specified in Figure 7.1, on a small breadboard in the lab; see Figure 7.2. Use a resistor with $R = 982\ \Omega$, an inductor with $L = 5.30$ mH, and a capacitor with $C = 70.2$ nF. (These components will be set aside in the lab; use the multimeter to measure these values, which may exhibit some slight deviation from those specified.) Connect the function generator to the circuit using the black and red terminals.

   Tune the function generator to 8 kHz. Then slowly increase the amplitude until you observe a response peak, then reduce amplitude slightly. (This ensures that we do not overload the circuit.) Now connect the oscilloscope probe to the function generator input and measure the amplitude of the signal.

   Reconnect the oscilloscope probe to the node $x(t)$ in Figure 7.1. Now reduce the frequency to 2 kHz and step through in 1 kHz intervals to 18 kHz. At each frequency, record the maximum amplitude. Submit a MATLAB plot of frequency versus amplitude, which should resemble numerically-computed plot in Figure 7.4.

As usual, please include all MATLAB scripts in your lab report.

▶ Optional: Band-stop Filter

A simple rearrangement of the band-pass filter yields the circuit in Figure 7.5, now called a *band-stop filter*. Whereas a band-pass filter supports inputs $\sin(\omega t)$ at particular frequencies ω, a band-stop filter does just the opposite: it prevents inputs $v(t) = \sin(\omega t)$ at a distinct range of frequencies ω from passing. Band-stop filters are used in public address systems and musical instrument amplifiers.

1. Develop a second order differential equation akin to (7.3) for the band-stop circuit.

2. Produce backward Euler solutions to this equation with $v(t) = \sin(\omega t)$ for a range of ω values and the same R, L, and C values measured for the band-pass filter. Produce plots for various ω that confirm the expected band-stop behavior.

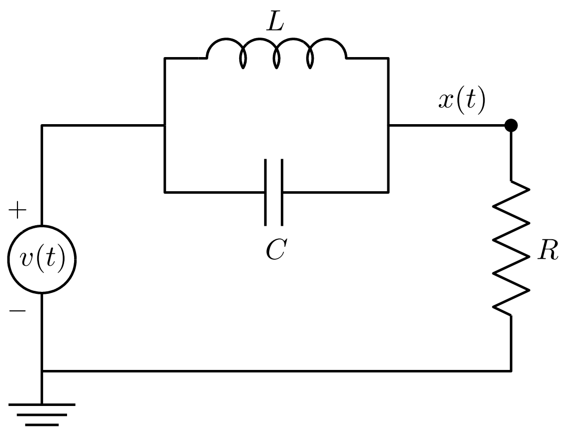


Figure 7.5: A band-stop circuit.

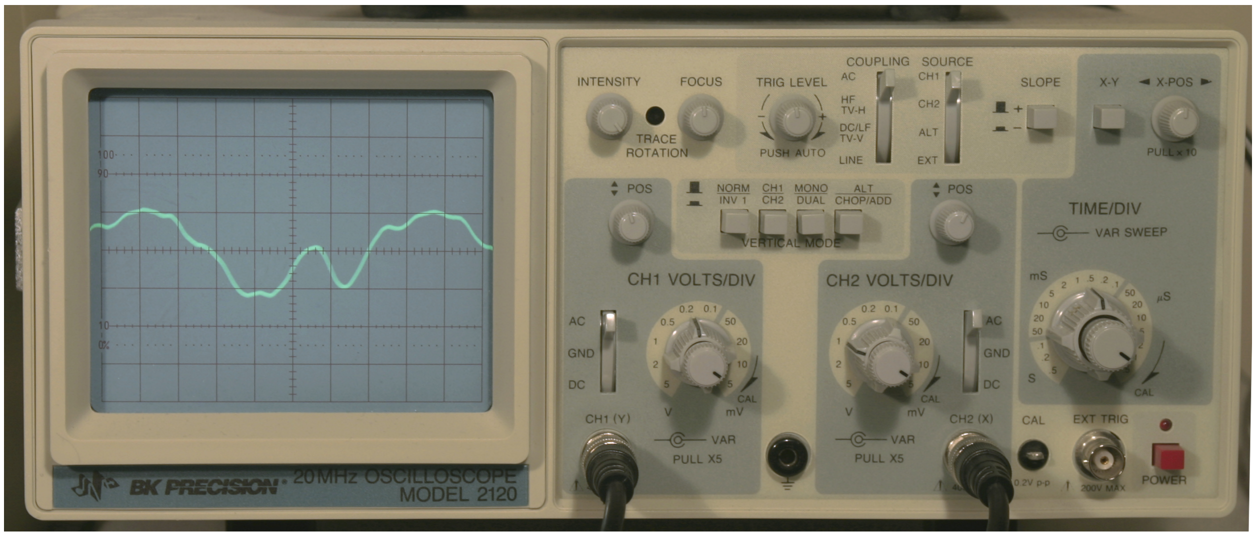


Figure 7.6: The oscilloscope in the lab.

3. Confirm your mathematical model and numerical results by implementing the circuit on the lab breadboard, measuring the amplitude, and comparing the results to predictions from the backward Euler method. Use ω values below the band, in the band, and above the band.

▶ Appendix: Using the oscilloscope

Oscilloscopes contain many advanced features but you'll only need to work with a small subset. On the left there is a CRT screen on to which a line is projected representing the amplitude as a function of time. For this lab we will only need to work on the first channel. Its settings are located immediately to the right of the screen. The switch with options 'AC', 'GND', and 'DC' should be set to DC. Use the dial labeled 'POS' to move the trace line up and down. The dial labeled 'CH1 VOLTS/DIV' indicates the scale for each of the horizontal lines crossing the screen.

To measure amplitude, measure the number of vertical units spanned by the signal on the screen. The solid lines denote whole units; the secondary hash marks give further divisions of 0.2 units. Check that the signal shown in Figure 7.6 has an amplitude of 2.2 units. Now multiply this amplitude by the VOLTS/DIV setting. Here this is $2.2 \times 0.2 = 0.44$; this is the peak-to-peak amplitude. We are interested in peak to ground amplitude, so divide this result by two to arrive at the final amplitude, 0.22 V.

Lab 8: Dynamics of Compound Pendula

In the first part of the semester, we followed our initial labs on static circuits with analogous labs involving mechanical systems. We shall follow that pattern again, now with dynamical systems. In the last lab we studied how electrical potential evolves in time in an RLC circuit. Now, we study the motion of swinging masses attached to a cable: a compound pendulum. Our inspiration for this study is a fundamental paper published in 1733 by the Swiss mathematician Daniel Bernoulli ("*Theoremata de Oscillationibus Coporum filo Flexili Connexorum et catena Verticaliter Suspensae*"), and the systemization of this work described by Russian mathematicians Felix Gantmacher and Mark Krein in their book, *Oscillation Matrices and Kernels and Small Vibrations of Mechanical Systems* (AMS, 2002).

▶ Setting

We are concerned with a (massless) string suspended from a fixed block. Upon this string we mount n point masses m_1, m_2, …, m_n. Let ℓ_k denote the distance between mass m_k and m_{k+1} (or, in the case of $k = n$, between mass m_k and the block). The physical arrangement of this *compound pendulum* is illustrated in Figure 8.1, where the radii of the black circles are drawn proportional to the (point) masses.

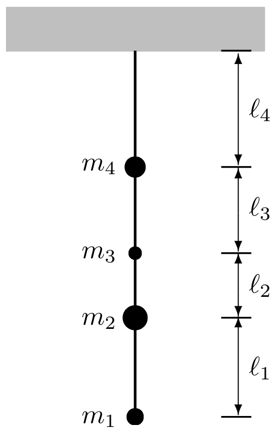


Figure 8.1: Pendulum with four masses at rest.

In the absence of external forces, this configuration with all masses hanging in a vertical line will remain still. Suppose we displace one or more of the masses from the vertical and release it. How will the system evolve in time? The mechanics are described by *nonlinear* differential equations, and when the displacements are quite large the behavior can be exceptionally complex. In this lab we restrict our attention to small displacements, which allow us to approximate to high accuracy the evolution of the system in time using *linear* differential equations. We shall see that the behavior of these equations can be completely understood from the *spectral properties* (eigenvalues and eigenvectors) of an $n \times n$ matrix.

▶ Kinetic and Potential Energies

Let $x_1(t), \ldots, x_n(t)$ denote the horizontal displacement of masses $m_1, \ldots, m_n$ from the vertical rest configuration at time $t \geq 0$. Given knowledge of the masses and string lengths, and the state of the system at time $t = 0$ (specified through $x_1(0), \ldots, x_n(0)$), we wish to determine the displacements of the masses at all future times. Our main tool for this will be the *Principle of Least Action* and the allied *Euler–Lagrange equation*, which relates a system's kinetic and potential energies.

The kinetic energy in the entire pendulum is found by summing the kinetic energy $\frac{1}{2}m_k[x_k'(t)]^2$ associated with each mass:

$$T(t) = \frac{1}{2}\sum_{k=1}^{n} m_k[x_k'(t)]^2.$$

Similarly, the total potential energy must be summed over all the masses; on each segment we need the product of the tension and elongation. The tension σ_k on the kth segment depends on the total mass suspended under it:

$$\sigma_k = g\sum_{j=1}^{k} m_j,$$

where g is the acceleration due to gravity. The elongation requires approximation. As can be deduced from Figure 8.2 with the help of the Pythagorean Theorem, the stretched length of the kth segment is simply

$$\sqrt{\ell_k^2 + (x_k - x_{k+1})^2} - \ell_k.$$

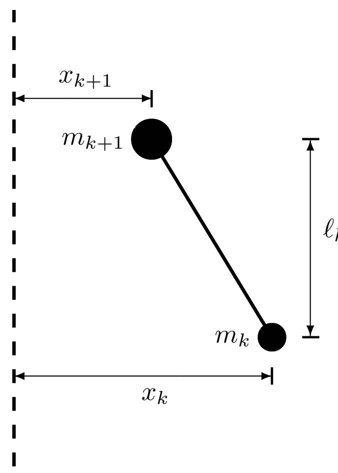


Figure 8.2: Stretched segment.

(You might be concerned about the $k = n$ case: the nth segment is anchored at the top; there is no m_{k+1} to be displaced. If we define $x_{n+1} = 0$, this elongation formula works without a hitch.) The square root puts the elongation formula beyond the realm of *linear* algebra. However, since the displacements x_k and x_{k+1} are both assumed to be small relative to ℓ_k, we can use the same *linear approximation* that arose earlier in the semester during our study of the biaxial truss. Using the Taylor expansion $\sqrt{1+\xi} = 1 + \frac{1}{2}\xi - \frac{1}{8}\xi^2 + \cdots$, we have

$$\begin{aligned}
\sqrt{\ell_k^2 + (x_k - x_{k+1})^2} - \ell_k &= \sqrt{\ell_k^2\Big(1 + \frac{(x_k - x_{k+1})^2}{\ell_k^2}\Big)} - \ell_k \\
&= \ell_k\sqrt{1 + \frac{(x_k - x_{k+1})^2}{\ell_k^2}} - \ell_k \\
&= \ell_k\Big(1 + \frac{1}{2}\frac{(x_k - x_{k+1})^2}{\ell_k^2} - \frac{1}{8}\frac{(x_k - x_{k+1})^4}{\ell_k^4} + \cdots\Big) - \ell_k \\
&\approx \ell_k\Big(1 + \frac{1}{2}\frac{(x_k - x_{k+1})^2}{\ell_k^2}\Big) - \ell_k \\
&= \frac{(x_k - x_{k+1})^2}{2\ell_k}.
\end{aligned}$$

The total potential energy is then the sum of the products of tension and elongation:

$$V(t) = \frac{1}{2}\sum_{k=1}^{n} \frac{\sigma_k}{\ell_k}(x_k - x_{k+1})^2.$$

▶ Euler–Lagrange Equation

The Principle of Least Action posits that a system will move in time in the manner that yields the smallest "action". This is quantified through the Euler–Lagrange equation:

$$\frac{d}{dt}\frac{\partial T}{\partial x'_j} + \frac{\partial V}{\partial x_j} = 0, \qquad j = 1, \ldots, n.$$

We shall apply these equations to our system one step at a time. First, consider $\partial T/\partial x'_j$. The notation merits a little explanation: we are treating the entity x'_k as a variable, and differentiating with respect to that symbol. Hence

$$\frac{\partial T(t)}{\partial x'_j} = \frac{\partial}{\partial x'_j}\Big(\frac{1}{2}\sum_{k=1}^{n} m_k[x'_k(t)]^2\Big) = \frac{1}{2}\sum_{k=1}^{n} m_k\Big(\frac{\partial}{\partial x'_j}[x'_k(t)]^2\Big) = m_j\, x'_j(t).$$

With this formula in hand, it is simple to take a time derivative:

$$\frac{d}{dt}\frac{\partial T}{\partial x'_j} = \frac{d}{dt}\Big(m_j\, x'_j(t)\Big) = m_j\, x''_j(t).$$

We turn our attention to the potential energy:

$$\frac{\partial V(t)}{\partial x_j} = \frac{\partial}{\partial x_j}\Big(\frac{1}{2}\sum_{k=1}^{n} \frac{\sigma_k}{\ell_k}(x_k - x_{k+1})^2\Big) = \frac{1}{2}\sum_{k=1}^{n} \frac{\sigma_k}{\ell_k}\Big(\frac{\partial}{\partial x_j}(x_k - x_{k+1})^2\Big)$$

The partial derivative in the rightmost term will be zero except when $j = k$ or $j = k+1$, and hence

$$\begin{aligned} \frac{\partial V(t)}{\partial x_j} &= \frac{1}{2}\Big(\frac{\sigma_{j-1}}{\ell_{j-1}}(-2x_{j-1} + 2x_j) + \frac{\sigma_j}{\ell_j}(2x_j - 2x_{j+1})\Big) \\ &= \Big(-\frac{\sigma_{j-1}}{\ell_{j-1}}\Big)x_{j-1} + \Big(\frac{\sigma_{j-1}}{\ell_{j-1}} + \frac{\sigma_j}{\ell_j}\Big)x_j + \Big(-\frac{\sigma_j}{\ell_j}\Big)x_{j+1}. \end{aligned}$$

The cases $j = 1$ and $j = n$ require some scrutiny. In the former case, there is no $j-1$ term, so we have

$$\begin{aligned} \frac{\partial V(t)}{\partial x_1} &= \frac{1}{2}\sum_{k=1}^{n} \frac{\sigma_k}{\ell_k}\Big(\frac{\partial}{\partial x_1}(x_k - x_{k+1})^2\Big) = \frac{1}{2}\frac{\sigma_1}{\ell_1}\Big(\frac{\partial}{\partial x_1}(x_1 - x_2)^2\Big) \\ &= \Big(\frac{\sigma_1}{\ell_1}\Big)x_1 + \Big(-\frac{\sigma_1}{\ell_1}\Big)x_2. \end{aligned}$$

For $j = n$, substituting the identity $x_{n+1} = 0$ into general formula for $\partial V/\partial x_j$ gives

$$\frac{\partial V(t)}{\partial x_n} = \Big(-\frac{\sigma_{n-1}}{\ell_{n-1}}\Big)x_{n-1} + \Big(\frac{\sigma_{n-1}}{\ell_{n-1}} + \frac{\sigma_n}{\ell_n}\Big)x_n.$$

Inserting these various pieces into the Euler–Lagrange equation, we obtain

$$m_j x_j''(t) = \begin{cases} \Big(-\dfrac{\sigma_1}{\ell_1}\Big)x_1 + \Big(\dfrac{\sigma_1}{\ell_1}\Big)x_2, & j=1; \\[2ex] \Big(\dfrac{\sigma_{j-1}}{\ell_{j-1}}\Big)x_{j-1} + \Big(-\dfrac{\sigma_{j-1}}{\ell_{j-1}} - \dfrac{\sigma_j}{\ell_j}\Big)x_j + \Big(\dfrac{\sigma_j}{\ell_j}\Big)x_{j+1}, & 1<j<n; \\[2ex] \Big(\dfrac{\sigma_{n-1}}{\ell_{n-1}}\Big)x_{n-1} + \Big(-\dfrac{\sigma_{n-1}}{\ell_{n-1}} - \dfrac{\sigma_n}{\ell_n}\Big)x_n, & j=n. \end{cases}$$

This equation describes the coupling between the displacements of the various masses (notice the form: mass times acceleration equals force). All the x_j variables appear linearly, hence we have a linear system of second-order differential equations. To simplify computation and analysis, we shall write this out using vectors and matrices.

▶ Matrix Formulation

Let $x \in \mathbb{R}^n$ denote the vector of horizontal displacements,

$$x(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_n(t) \end{bmatrix}.$$

We wish to find $n \times n$ matrices M and K that collect the Euler–Lagrange equations for $j = 1, \dots, n$ into one matrix differential equation of the form

$$Mx''(t) = -Kx(t),$$

whose jth row encodes the Euler–Lagrange equation for index j. (The negative sign on the right will prove useful later.) You should verify that these matrices take the form

$$\underbrace{\begin{bmatrix} m_1 & & & & \\ & m_2 & & & \\ & & m_3 & & \\ & & & \ddots & \\ & & & & m_n \end{bmatrix}}_{M} \underbrace{\begin{bmatrix} x_1''(t) \\ x_2''(t) \\ x_3''(t) \\ \vdots \\ x_n''(t) \end{bmatrix}}_{x''(t)} = \underbrace{\begin{bmatrix} -\frac{\sigma_1}{\ell_1} & \frac{\sigma_1}{\ell_1} & & & \\ \frac{\sigma_1}{\ell_1} & -\frac{\sigma_1}{\ell_1}-\frac{\sigma_2}{\ell_2} & \frac{\sigma_2}{\ell_2} & & \\ & \frac{\sigma_2}{\ell_2} & -\frac{\sigma_2}{\ell_2}-\frac{\sigma_3}{\ell_3} & \ddots & \\ & & \ddots & \ddots & \frac{\sigma_{n-1}}{\ell_{n-1}} \\ & & & \frac{\sigma_{n-1}}{\ell_{n-1}} & -\frac{\sigma_{n-1}}{\ell_{n-1}}-\frac{\sigma_n}{\ell_n} \end{bmatrix}}_{-K} \underbrace{\begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \\ \vdots \\ x_n(t) \end{bmatrix}}_{x(t)},$$

with all blank entries equal to zero.

The matrix M is diagonal, and as the masses are positive it must be invertible. Hence we can write

$$x''(t) = -M^{-1}Kx(t).$$

▶ Solving the Differential Equation: An Introduction to Eigenvalues and Eigenvectors

What can we say about solutions to the differential equation $x''(t) = -M^{-1}Kx(t)$? There are several ways to proceed; perhaps the most direct way is to "guess" that $x(t)$ has the special form

$$x(t) = \cos(\omega t)u,$$

where $u \in \mathbb{R}^n$ *does not depend on time.* That is, we are looking for a solution $x(t)$ that always has the same direction u in $\mathbb{R}^n$, but has an amplitude $\cos(\omega t)$ that varies sinusoidally in time. (See the Appendix for a more satisfying approach that does not require the same degree of inspiration.)

We want to check that this particular form of $x(t)$ satisfies the differential equation $x''(t) = -M^{-1}Kx(t)$. First we compute the left hand side,

$$x''(t) = -\omega^2 \cos(\omega t)u,$$

and compare it to the right hand side,

$$-M^{-1}Kx(t) = -\cos(\omega t)M^{-1}Ku.$$

Canceling $\cos(\omega t)$, these two sides agree only in the case that ω and u satisfy

$$M^{-1}Ku = \omega^2 u.$$

Is it possible that such ω and (nonzero) u exist to satisfy this equation?

The answer to this question is found in the study of *spectral theory*: ω^2 is called an *eigenvalue* of $M^{-1}K$, and u is an associated *eigenvector*. As you will see in the CAAM 335 lectures, an $n \times n$ matrix can never have more than n distinct eigenvalues, and it turns out that for matrices of the kind we have derived for the swinging pendulum, there will always be *exactly* n distinct positive real eigenvalues, which we shall label

$$\omega_1^2 < \omega_2^2 < \cdots < \omega_n^2.$$

With each eigenvalue, we associate corresponding eigenvectors

$$u_1, u_2, \ldots, u_n,$$

which we write componentwise as

$$u_1 = \begin{bmatrix} u_{1,1} \\ u_{1,2} \\ \vdots \\ u_{1,n} \end{bmatrix}, \quad u_2 = \begin{bmatrix} u_{2,1} \\ u_{2,2} \\ \vdots \\ u_{2,n} \end{bmatrix}, \quad \cdots, \quad u_n = \begin{bmatrix} u_{n,1} \\ u_{n,2} \\ \vdots \\ u_{n,n} \end{bmatrix}.$$

For these problems, we are guaranteed that $\{u_1, u_2, \ldots, u_n\}$ are *linearly independent*, and hence form a *basis* for $\mathbb{R}^n$. Notice that the eigenvectors are only specified up to a scalar multiple: If $M^{-1}Ku_j = \omega_j^2 u_j$, then $M^{-1}K(\alpha u_j) = \omega_j^2(\alpha u_j)$ for any α. It is often convenient to normalize, scaling so that $\|u_j\| = 1$ or $u_{j,n} = 1$.

To compute these eigenvalues and eigenvectors in MATLAB, one can call

```
[U,W2] = eig(M\K);
```

To impose the ordering $\omega_1^2 < \cdots < \omega_n^2$, we can command

```
[junk,indx] = sort(real(diag(W2)));
U = U(:,indx);
W2 = W2(indx,indx);
```

To normalize so that $u_{j,n} = 1$ for all j, we might further issue

```
U = U*diag(1./U(n,:));
```

At this point, we have

$$\begin{aligned} \omega_j^2 &= \texttt{W2(j,j)} \\ u_j &= \texttt{U(:,j)} \end{aligned}$$

▶ Examining Distinguished Solutions

We have discovered n special solutions to the differential equation $x''(t) = -M^{-1}Kx(t)$, namely:

$$x(t) = \cos(\omega_j t)\, u_j.$$

If we displace the pendulum's masses by the entries of the eigenvector u_j and then simultaneously release, the ensemble will oscillate according to this solution: all future displacements will be multiples of the eigenvector, and the amplitude will vary sinusoidally with angular frequency ω_j. In this lab, you will verify this statement for several small ensembles. (Of course, experimental errors in measurement, and modeling errors such as the neglect of friction, will cause some deviation from the ideal.)

In Figure 8.3 we illustrate the two key displacements for a system with $n = 2$ and

$$m_1 = m_2 = 1, \qquad \ell_1 = \ell_2 = 2,$$

with units selected such that $g = 1$. In this case, we have

$$M = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \qquad K = \begin{bmatrix} 1/2 & -1/2 \\ -1/2 & 3/2 \end{bmatrix},$$

and corresponding eigenvalues and eigenvectors (see Figure 8.3)

$$\omega_1^2 = 1 - \frac{\sqrt{2}}{2}, \qquad u_1 = \begin{bmatrix} 1 + \sqrt{2} \\ 1 \end{bmatrix}$$

and

$$\omega_2^2 = 1 + \frac{\sqrt{2}}{2}, \qquad u_2 = \begin{bmatrix} 1 - \sqrt{2} \\ 1 \end{bmatrix}.$$

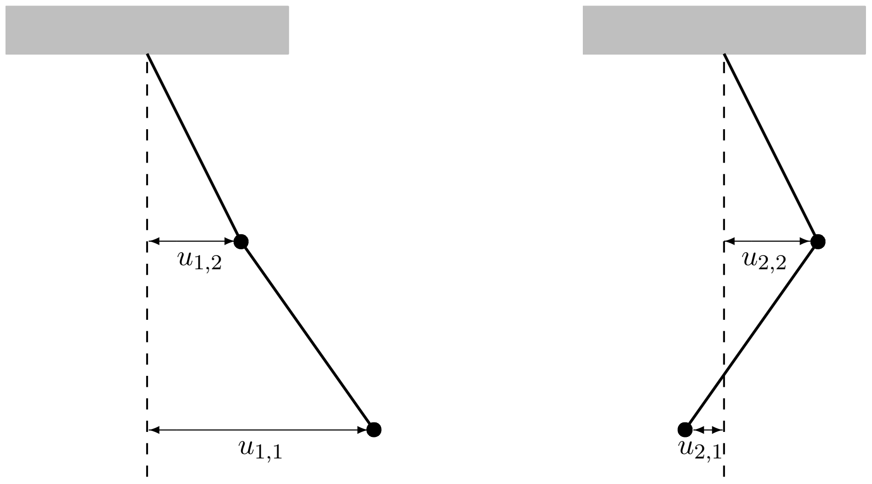


Figure 8.3: Distinguished displacements in the direction u_1 (left) and u_2 (right).

▶ Lab Instructions

Does this mathematical model give an accurate description of pendulum motion? Can you detect the n different frequencies present in an n-mass system? We shall investigate these questions using the apparatus shown in Figure 8.4.

1. Select $n = 2$ masses (see Figure 8.5), weigh them, set up the M and K matrices, and compute the eigenvalues and eigenvectors of $M^{-1}K$ (in MATLAB or by hand, as you prefer).

2. For each eigenvalue, use the eigenvector entries to plot out displacements for each mass (as in Figure 8.3) on the white-board backdrop of the lab A-frame. Displace the masses to these points. Release the masses and film the motion, as described in the section "Making the Recording" below. Use the MATLAB script `track`, as described in the section "Analyzing the Data" below, to process these images into horizontal displacements. The `track` script also takes the `fft` of the

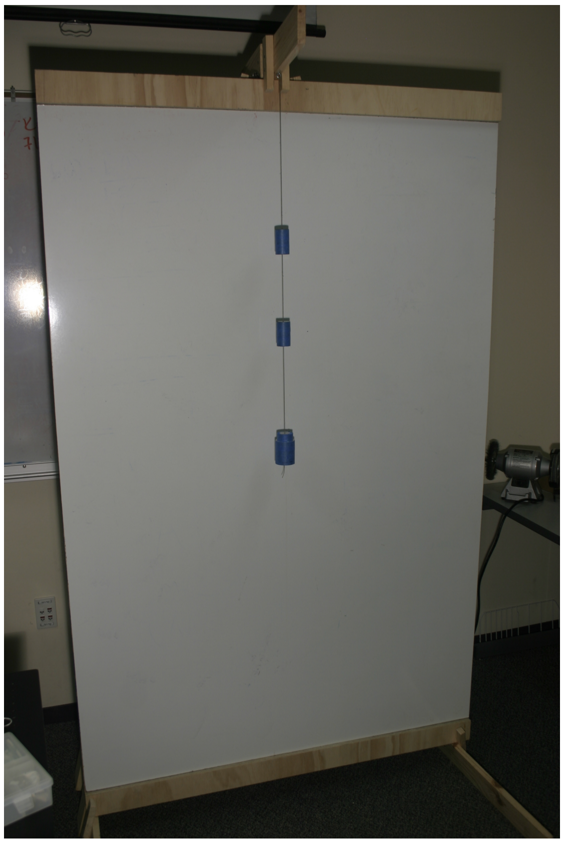


Figure 8.4: Experimental set-up: three masses are suspended on a flexible cable.

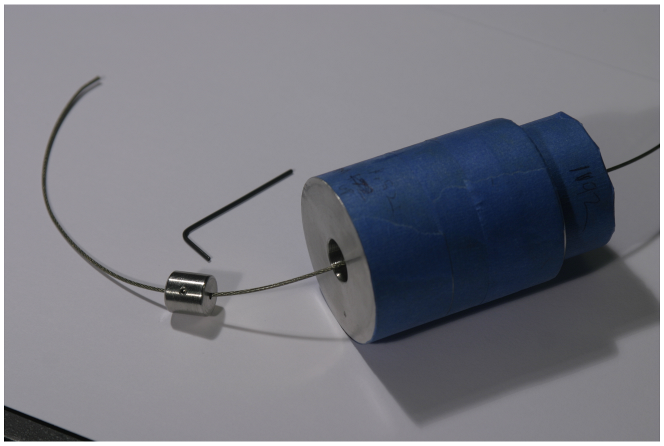


Figure 8.5: steel mass, wrapped with blue tape to allow for image capture against a white blackdrop. The mass is 'attached' to the cable that runs through its vertical axis by means of a small nut. The nut has a small screw that can be tightened against the cable with the pictured Allen wrench.

horizontal displacements for each mass. Take note of the peaks in this frequency plot. Do the most prominent frequencies obtained from the different masses agree?

The frequencies are plotted in Hertz, i.e., cycles per second. A sinusoid vibrating at f Hz would thus have the form, e.g., $\cos(2\pi f t)$: over one unit of time, this cosine completes f periods. In our notation, this corresponds to $\omega = 2\pi f$, i.e., the eigenvalue

$$\omega^2 = 4\pi^2 f^2.$$

For each of your n measured frequencies f, derive the eigenvalue ω^2. How well do these agree with the eigenvalues of $M^{-1}K$?

3. Repeat the above experiment, but now with a random displacement of the masses. Can you detect the presence of both eigenvalues in the data you record? How pure were your excitations in the previous section? Quantify by computing the ratio of the desired mode's amplitude to the sum of each mode's amplitude.

4. Repeat exercises 1–3 for a system with $n = 3$ masses.

▶ Making the Recording

1. Open `VRecord` under `Programs->Philips ToUCam Camera->Philips Vrecord`.

2. Under `Options->Video Format` set `Output size` to 640x480.

3. Under `File->Set Capture File`, give a unique file name in `C:\CAAMLab\Videos`. You do not want to place the recording on your network mounted directory, as the latency in writing to that disk will affect the recording.

4. Start your pendulum in motion.

5. Start the recording using `Capture->Start Recording`.

6. After some time, stop the recording. (Try 30 seconds; this will be sufficient to capture a number of periods of vibration. Longer capture times should yield greater accuracy; use the intuition you developed with synthetic data in Lab 6.)

7. At this point, your recording is ready but the avi encoding used by `VRecord` does not cooperate well with MATLAB; we need a kludge to fix things up. Open the program `VirtualDub` by clicking on the link on the desktop. Open your video file in `VirtualDub`, and then save it as an avi file using a different name.

▶ Analyzing the Data

The MATLAB function `track` takes the name of an avi file and tracks the motion of the masses in a similar manner to the `findnodes` routine used in the spring network lab. To call this function, give the name of the avi file, including the `.avi` extension (`name`) and the number of masses to track (`n`) and enter

```
        [X,Y] = track(name,n);
```

where `X` and `Y` contain the displacement of the masses in the horizontal and vertical directions. (Recall that we have assumed that the masses undergo negligible vertical displacement.)

▶ Appendix: Another Approach to the Differential Equation

As the eigenvectors $\{u_1, \ldots, u_n\}$ form a basis for $\mathbb{R}^n$, any displacement of the masses can be written as a linear combination,

$$x(t) = \sum_{j=1}^{n} \gamma_j(t) u_j. \tag{8.1}$$

In particular, suppose we the initial disturbance

$$x(0) = \sum_{j=1}^{n} \gamma_j(0) u_j.$$

As this is equivalent to

$$\begin{bmatrix} x_1(0) \\ x_2(0) \\ \vdots \\ x_n(0) \end{bmatrix} = \begin{bmatrix} | & | & & | \\ u_1 & u_2 & \cdots & u_n \\ | & | & & | \end{bmatrix} \begin{bmatrix} \gamma_1(0) \\ \gamma_2(0) \\ \vdots \\ \gamma_n(0) \end{bmatrix},$$

we could determine the initial coefficients $\gamma_1(0), \ldots, \gamma_n(0)$ via

$$\begin{bmatrix} \gamma_1(0) \\ \gamma_2(0) \\ \vdots \\ \gamma_n(0) \end{bmatrix} = \begin{bmatrix} | & | & & | \\ u_1 & u_2 & \cdots & u_n \\ | & | & & | \end{bmatrix}^{-1} \begin{bmatrix} x_1(0) \\ x_2(0) \\ \vdots \\ x_n(0) \end{bmatrix}.$$

Now suppose further that the masses have no initial velocity, $x'(0) = 0$, i.e.,

$$\gamma_j'(0) = 0.$$

This initial data is sufficient to determine the coefficients $\gamma_j(t)$ for all times $t \geq 0$. To see this, substitute the formula (8.1) into the differential equation $x''(t) = -M^{-1}Kx(t)$ and use the fact that u_j is an eigenvector to obtain

$$\begin{aligned} \sum_{j=1}^{n} \gamma_j''(t) u_j &= -\sum_{j=1}^{n} \gamma_j(t) M^{-1} K u_j \\ &= -\sum_{j=1}^{n} \omega_j^2 \gamma_j(t) u_j. \end{aligned}$$

Because the vectors $u_1, \ldots, u_n$ are linearly independent, we can equate the coefficients in this sum:

$$\gamma_j''(t) = -\omega_j^2 \gamma_j(t), \qquad j = 1, \ldots, n.$$

By writing our solution in the basis of eigenvectors, we have transformed an $n \times n$ matrix differential equation into n much simpler scalar equations. Each of these scalar second order homogeneous differential equations has the general solution

$$\gamma_j(t) = \alpha_j \sin(\omega_j t) + \beta_j \cos(\omega_j t).$$

We can use the initial value of $\gamma_j(0)$ and the fact that $\gamma_j'(0) = 0$ to determine α_j and β_j. The general solution gives

$$\gamma_j'(t) = \omega_j \alpha_j \cos(\omega_j t) - \omega_j \beta_j \sin(\omega_j t),$$

and in particular $\gamma_j'(0) = \omega_j \alpha_j$. As the system starts with $\gamma_j'(0) = 0$, we conclude $\alpha_j = 0$. Thus

$$\gamma_j(t) = \beta_j \cos(\omega_j t),$$

and so $\gamma_j(0) = \beta_j$. Thus, the solution to the differential equation $x''(t) = -M^{-1}Kx(t)$ is given by

$$x(t) = \sum_{j=1}^{n} \gamma_j(0) \cos(\omega_j t)\, u_j.$$

Do you see the great beauty of this expression? It says that the motion of the pendulum is simply a superposition of eigenvectors, and the component in each eigenvector varies at a different frequency, ω_j. Let us examine two particular cases.

- When you start the pendulum with some random initial displacement, the motion will generically exhibit all frequencies $\omega_1, \ldots, \omega_n$, and these will be exhibited when you compute a Fourier transform of the displacement measurements for any of the masses.

- If you started the pendulum with displacements exactly corresponding to the eigenvector u_k, you would have $\gamma_j(0) = 0$ for $j \neq k$, and the solution would be

$$x(t) = \gamma_k(0)\cos(\omega_k t)\,u_k.$$

  Of course, in practice you will only have $\gamma_j(0) \approx 0$, but since $|\gamma_j(t)| = |\gamma_j(0)\cos(\omega_j t)| \leq |\gamma_j(0)|$, the components in the unwanted eigenvectors do not get magnified: $\gamma_k(0)\cos(\omega_k t)\,u_k$ will still dominate overall.

Lab 9: Dynamics of a Beaded String

In this lab we investigate a second vibration problem: the motion of beads threaded on a taut string that is anchored at both ends. Like the last lab's compound pendulum, this problem stimulated great developments in mechanics during the eighteenth century. As we take more and more masses and pack them ever closer together, we obtain a model of a *continuous* string, as you might find on a violin or guitar. It turns out that the eigenvalues associated with such a string explain the pleasant progression of musical notes—but that is a story best saved for a course in partial differential equations.

Here, we imagine n point-masses $m_1, \ldots, m_n$ fixed horizontally from left to right on a taut massless string of total length ℓ. The rest positions of the masses provide a natural partition of the string, so the length between the left support and the first bead is ℓ_0, between the first and second bead is ℓ_1, etc., as illustrated for $n = 4$ in Figure 10.1, so that $\ell = \sum_{k=0}^{n} \ell_k$.

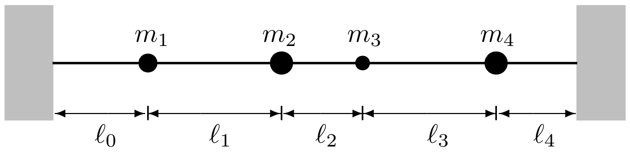


Figure 9.1: A string with four beads at rest.

Now we pluck the string lightly, so as to induce a vibration among the beads. As the beads are fixed on the string, they do not slide horizontally. For small vibrations, the dominant displacement of each mass will be in perpendicular to the rest state, i.e., vertical. We denote the displacement of mass j at time t by $y_j(t)$, as illustrated in Figure 9.2.

Our goal is to determine the displacements $y_j(t)$, given a description of the initial pluck and the various masses and lengths. In next week's lab, the culminating project of the semester, we shall solve the *inverse problem*: Given knowledge of how a beaded string vibrates, how heavy are the beads and where are they on the string?

Unsurprisingly, the mathematics behind bead vibrations is essentially identical to that of the compound pendulum; the only difference is the horizontal arrangement of the masses and the fact that both ends are now fixed. For details on general systems of this sort, see Gantmacher and Krein, *Oscillation Matrices and Kernels and Small Vibrations of Mechanical Systems* (AMS, 2002). We begin by deriving formulas for the kinetic and potential energies, then set up a system of linear

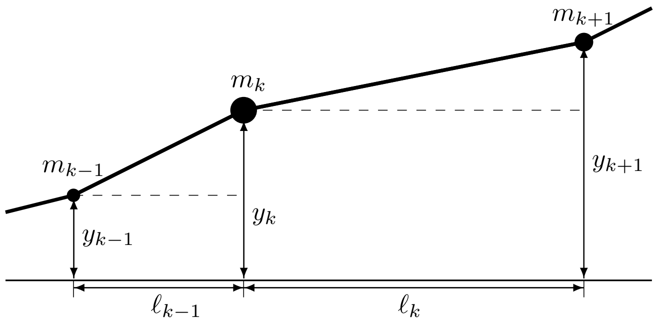


Figure 9.2: Close-up of a stretched segment of the string.

differential equations.

▶ Kinetic and Potential Energies

As before, we use the Euler–Lagrange equation to derive a system of differential equations that describe the displacements. The formula for kinetic energy is unchanged from the pendulum case; adding up '$\frac{1}{2}mv^2$' at each bead gives

$$T(t) = \frac{1}{2}\sum_{k=1}^{n} m_k[y_k'(t)]^2.$$

The potential energy requires a more subtle computation; again, it will be the sum of the potential energies in each segment, that is, the sum of the products of tension and elongation. Since the beads are arranged horizontally, rather than vertically as in the pendulum, we presume that the string is held in a constant tension σ throughout the entire ensemble. The elongation computation is identical to the calculation for the pendulum. Suppose we wish to measure the elongation of the kth segment, as illustrated in Figure 9.2. Assuming that the beads are only moving vertically, the stretched length of this segment is simply

$$\sqrt{\ell_k^2 + (y_{k+1} - y_k)^2} - \ell_k,$$

and as in the previous lab, we use the fact that ℓ_k is much larger than the displacements y_k and y_{k+1} to argue that

$$\begin{aligned}\sqrt{\ell_k^2 + (y_{k+1} - y_k)^2} - \ell_k &= \ell_k\sqrt{1 + \frac{(y_{k+1} - y_k)^2}{\ell_k^2}} - \ell_k \\ &\approx \ell_k\Big(1 + \frac{1}{2}\frac{(y_{k+1} - y_k)^2}{\ell_k^2}\Big) - \ell_k \\ &= \frac{(y_{k+1} - y_k)^2}{2\ell_k}.\end{aligned}$$

Some special attention must be paid to the end cases, $k = 0$ and $k = n$. These can be handled just as easily as the top segment of the pendulum: since the ends of the string remain fixed, we *define* $y_0 = y_{n+1} = 0$, and the elongation formula works out just right. As we have $n + 1$ segments corresponding to $\ell_0, \dots, \ell_n$, the formula for the potential energy takes the form

$$V(t) = \frac{\sigma}{2} \sum_{k=0}^{n} \frac{(y_{k+1} - y_k)^2}{\ell_k}.$$

▶ Euler–Lagrange Equations

We substitute our formulas into the Euler–Lagrange equation

$$\frac{d}{dt}\frac{\partial T}{\partial y_j'} + \frac{\partial V}{\partial y_j} = 0, \qquad j = 1, \dots, n$$

just as in the last lab. From the kinetic energy formula, we have

$$\frac{d}{dt}\frac{\partial T}{\partial y_j'}(t) = m_j y_j''(t).$$

The potential energy term,

$$\frac{\partial V}{\partial y_j} = \frac{\sigma}{2} \sum_{k=0}^{n} \frac{\partial}{\partial y_j} \frac{(y_{k+1} - y_k)^2}{\ell_k},$$

again requires a little more work. The partial derivative within the sum on the right will generally be nonzero except in the cases where $k = j - 1$ or $k = j$. That is,

$$\begin{aligned}
\frac{\partial V}{\partial y_j} &= \frac{\sigma}{2}\Big(\frac{\partial}{\partial y_j}\Big[\frac{(y_j - y_{j-1})^2}{\ell_{j-1}}\Big] + \frac{\partial}{\partial y_j}\Big[\frac{(y_{j+1} - y_j)^2}{\ell_j}\Big]\Big) \\
&= \frac{\sigma}{2}\Big(\frac{2y_j - 2y_{j-1}}{\ell_{j-1}} + \frac{2y_j - 2y_{j+1}}{\ell_j}\Big) \\
&= \Big(-\frac{\sigma}{\ell_{j-1}}\Big)y_{j-1} + \Big(\frac{\sigma}{\ell_{j-1}} + \frac{\sigma}{\ell_j}\Big)y_j + \Big(-\frac{\sigma}{\ell_j}\Big)y_{j+1}.
\end{aligned}$$

The cases of $j = 1$ and $j = n$ are special, as $y_0 = y_{n+1} = 0$, and hence

$$\begin{aligned}
\frac{\partial V}{\partial y_1} &= \Big(\frac{\sigma}{\ell_0} + \frac{\sigma}{\ell_1}\Big)y_1 + \Big(-\frac{\sigma}{\ell_1}\Big)y_2 \\
\frac{\partial V}{\partial y_n} &= \Big(-\frac{\sigma}{\ell_{n-1}}\Big)y_{n-1} + \Big(\frac{\sigma}{\ell_{n-1}} + \frac{\sigma}{\ell_n}\Big)y_n.
\end{aligned}$$

These partial derivatives couple each of the n masses to its nearest neighbors, and so we must solve the Euler–Lagrange equation at all masses simultaneously: these n coupled linear equations lead to a single differential equation involving $n \times n$ matrices.

▶ Matrix Formulation

Substituting the partial derivatives of kinetic and potential energies into the Euler–Lagrange equation gives

$$m_j y_j''(t) = \begin{cases} \Big(-\dfrac{\sigma}{\ell_0} - \dfrac{\sigma}{\ell_1}\Big)y_1 + \Big(\dfrac{\sigma}{\ell_1}\Big)y_2, & j = 1; \\[2ex] \Big(\dfrac{\sigma}{\ell_{j-1}}\Big)y_{j-1} + \Big(-\dfrac{\sigma}{\ell_{j-1}} - \dfrac{\sigma}{\ell_j}\Big)y_j + \Big(\dfrac{\sigma}{\ell_j}\Big)y_{j+1}, & 1 < j < n; \\[2ex] \Big(\dfrac{\sigma}{\ell_{n-1}}\Big)y_{n-1} + \Big(-\dfrac{\sigma}{\ell_{n-1}} - \dfrac{\sigma}{\ell_n}\Big)y_n, & j = n. \end{cases} ,$$

or, in matrix form,

$$\underbrace{\begin{bmatrix} m_1 & & & & \\ & m_2 & & & \\ & & m_3 & & \\ & & & \ddots & \\ & & & & m_n \end{bmatrix}}_{M} \underbrace{\begin{bmatrix} y_1''(t) \\ y_2''(t) \\ y_3''(t) \\ \vdots \\ y_n''(t) \end{bmatrix}}_{y''(t)} = \underbrace{\begin{bmatrix} -\frac{\sigma}{\ell_0} - \frac{\sigma}{\ell_1} & \frac{\sigma}{\ell_1} & & & \\ \frac{\sigma}{\ell_1} & -\frac{\sigma}{\ell_1} - \frac{\sigma}{\ell_2} & \frac{\sigma}{\ell_2} & & \\ & \frac{\sigma}{\ell_2} & -\frac{\sigma}{\ell_2} - \frac{\sigma}{\ell_3} & \ddots & \\ & & \ddots & \ddots & \frac{\sigma}{\ell_{n-1}} \\ & & & \frac{\sigma}{\ell_{n-1}} & -\frac{\sigma}{\ell_{n-1}} - \frac{\sigma}{\ell_n} \end{bmatrix}}_{-K} \underbrace{\begin{bmatrix} y_1(t) \\ y_2(t) \\ y_3(t) \\ \vdots \\ y_n(t) \end{bmatrix}}_{y(t)}.$$

This differential equation

$$y''(t) = -M^{-1}Ky(t) \tag{9.1}$$

has the same form as the equation that described motion of the compound pendulum masses, and we can solve it in the same manner. Let (ω_j^2, u_j) denote pairs of eigenvalues and eigenvectors of $M^{-1}K$, i.e., $M^{-1}Ku_j = \omega_j^2 u_j$ for $j = 1, \dots, n$. Stacking each of these n eigenvalue–eigenvector equations side by side results in the matrix equation

$$M^{-1}KU = U\Lambda,$$

where the $n \times n$ matrices U and Λ are defined as

$$U = \begin{bmatrix} | & | & & | \\ u_1 & u_2 & \cdots & u_n \\ | & | & & | \end{bmatrix}, \qquad \Lambda = \begin{bmatrix} \omega_1^2 & & \\ & \omega_2^2 & \\ & & \ddots & \omega_n^2 \end{bmatrix}.$$

The differential equation (9.1) can thus be translated to

$$y''(t) = -U\Lambda U^{-1}y(t),$$

whereby premultiplication by U^{-1} gives

$$U^{-1}y''(t) = -\Lambda U^{-1}y(t). \tag{9.2}$$

Let us focus for a moment on this vector $U^{-1}y(t)$, which we define as

$$\widehat{y}(t) = \begin{bmatrix} \gamma_1(t) \\ \gamma_2(t) \\ \vdots \\ \gamma_n(t) \end{bmatrix} := U^{-1}y(t).$$

Notice that

$$y(t) = UU^{-1}y(t) = U\,\widehat{y}(t) = \sum_{j=1}^{n} \gamma_j(t)u_j, \tag{9.3}$$

so the entries of $\widehat{y}(t)$ are simply the coefficients we need to express $y(t)$ in the basis of eigenvectors. Moreover, equation (9.2) is simply

$$\widehat{y}''(t) = -\Lambda\widehat{y}(t),$$

and since Λ is a diagonal matrix, this $n \times n$ system reduces to n independent scalar equations:

$$\gamma_j''(t) = -\omega_j^2\gamma_j(t),$$

each of which has a very simple general solution,

$$\gamma_j(t) = \gamma_j(0)\cos(\omega_j t) + \left(\frac{\gamma_j'(0)}{\omega_j}\right)\sin(\omega_j t), \tag{9.4}$$

in terms of the initial state $\gamma_j(0)$ and velocity $\gamma_j'(0)$.

The general solution (9.4) simplifies if we presume that the string begins with zero initial velocity,

$$y'(0) = 0 \quad\Longrightarrow\quad \widehat{y}'(0) = 0 \quad\Longrightarrow\quad \gamma_j'(0) = 0,$$

in which case

$$\gamma_j(t) = \gamma_j(0)\cos(\omega_j t).$$

Substituting these equations into the expansion of $y(t)$ in eigenvector coordinates (9.3) reveals a beautiful solution:

$$y(t) = \sum_{j=1}^{n} \gamma_j(0)\cos(\omega_j t)u_j.$$

Enjoy this equation for a moment: it says that the masses are displaced in concert as the superposition of n independent vectors vibrating at distinct frequencies. Can we detect these frequencies in the laboratory?

▶ Experimental Set-Up

The CAAM String Lab features a high-precision monochord that we shall use for this lab, illustrated in Figure 9.3. Tension is measured with a *force transducer* placed at the end of the string. The string then passes through a *collet*, which itself is mounted in a *collet vise* that, when tightened,

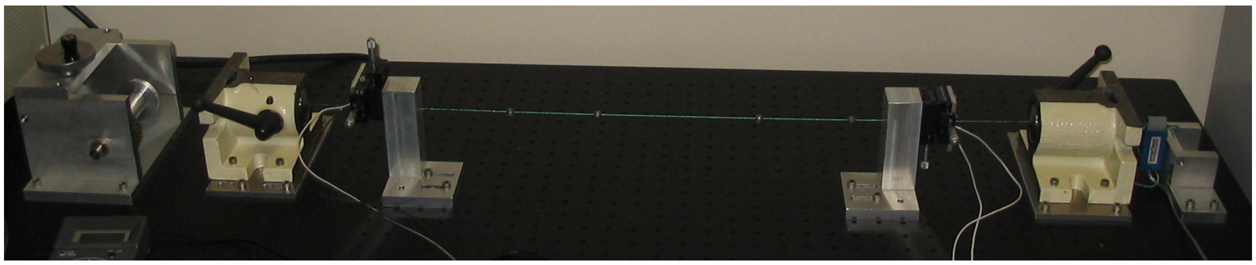


Figure 9.3: The CAAM String Lab monochord.

holds the end of the string tight. The string then passes through a photodetector that is used to measure the string vibrations at a single point along the string. Brass beads are threaded onto the string, which then passes through a second collet. (These beads have been carefully machined so as to snugly fit onto the properly chosen metal string.) Finally, the string is wound upon a spindle, which is used to apply tension to the string. (This tensioning requires a find hand: even metal strings will snap!) Close-ups of a collet in a vise and brass beads on a string are shown in Figure 9.4.

Notice that we do not measure the displacements of any one bead: the photodetector only measures the displacement of a single point along the string, but this point must vibrate at the same frequencies as the individual beads. As explained below, you will record displacements at a single point in time, then compute the Fourier transform of this data. The MATLAB code given below will produce a plot with horizontal axis in Hertz, i.e., cycles per second. As discussed in the last lab, a peak at f Hz corresponds to a frequency of

$$\omega_j = 2\pi f.$$

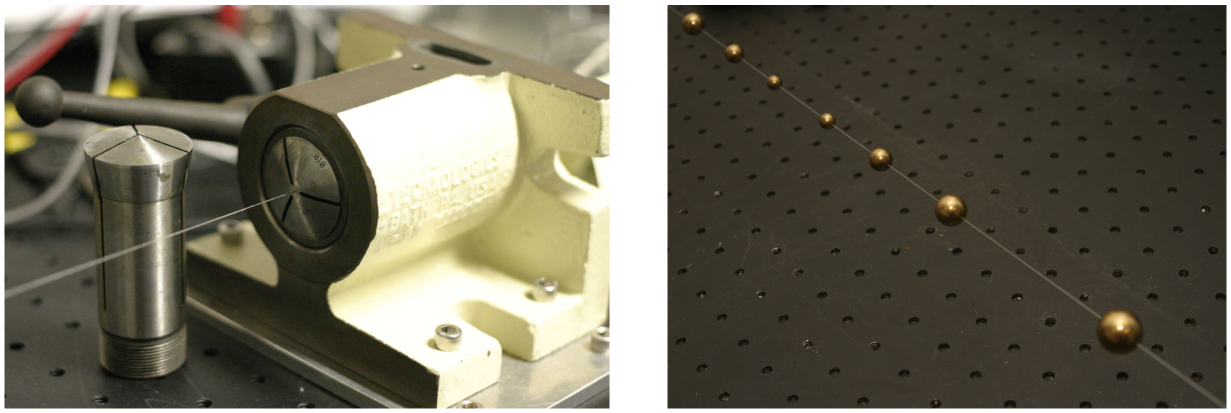


Figure 9.4: A collet clasping a string (left) and brass beads on a string (right).

▶ Setting Up a Beaded String

We describe the process starting with no string in the device.

- Place the beads on the string in the desired order.
- Run one end through the photodetector, the collet, the force transducer, and the eye bolt (which prevents the string from slipping through the force transducer). Run the other end through the second collet.
- On both sides, run the string through the clamps and tighten them.
- On the side of the tensioner, attach the bolt sticking out of the clamp through the hole in the spindle. On the other side, take up slack so the tensioner will not need to be turned so far.
- Turn the knob on the top of the tensioner until the string is straight.
- Now turn the handles on the collet holders to lock them down on the string.
- *Positioning the beads.* As the table is currently set up, the center of the string is between two grid holes. These holes are at one inch intervals. The best way to position the beads is to place your eye perpendicular to the table above a hole and then position the bead over it. Be sure to place the heavier beads to the outside, otherwise the vibrations of the lowest frequency mode are likely to be larger than the range of the photodetector.

▶ Taking Measurements

- Turn on the power to the measurement electronics by switching on the surge protector on the table. A red light should turn on at this point.
- Now run the `scope` script located in the `C:\CAAMlab\Scripts` directory.
- You should now see four plots. Expand the figure to fill the screen, so as to clearly resolve the text and features. The top row consists of the string displacement in the y direction parallel to the surface of the table and perpendicular to the string and in the z direction (which points up toward the ceiling). The blue line shows the displacement of the string at a point over a small time interval. The red line is a time average of this displacement. The two black lines represent the limit of the sensors and the cyan line represents the middle of the detector. The bottom row consists of the Fourier transform of each of the two plots above. Unfortunately, for the beaded string, most of features you will observe will be below about 200 Hz and not be distinguishable for reasonable refresh rates. Measurements of the tension of the string appear in the middle of the plots.

You can also use the script `stringWatch`, which omits the Fourier transform and thus is slightly faster.

- Now using the micrometer dials, center the string in the sensor by placing the blue line over the cyan line.

- Unlock the collets by turning their handles, increase tension to the desired amount (between 150 and 220 Netwons is safe), then lock the collets back. Note there will be some 'creep' in the string, illustrated by a constantly decreasing tension for a short time. Let this effect settle down before taking measurements.

- Practice plucking the string so that vibrations stay bounded within the two black lines and do not exhibit noise. Clipping, when the string travels outside of the range of the sensor, will create unwanted artifacts in the Fourier transform. Noise can also occur if the vibrations get too close to the boundary; this will manifest itself in jagged lines when the string approaches the boundary. If you have a hard time keeping the vibrations within the limits, increase the tension on the string, or pluck more gently.

- Stop the `scope` script by pressing `Control-C`.

- Now that you have practiced plucking, you are prepared to record a pluck. Recall that a rule of thumb for the resolution of a Fourier transform is that the Fourier transform of a signal of length T will have T points per Hertz. Since most of the features we want to resolve are between 10 and 200 Hertz, we require a sample that covers something like 10 or 100 seconds. We advise a sample rate of 5×10^4 samples per second, which is fast enough to prevent aliasing problems. Record the data using the command

  ```
  [x,sr] = getData(5e4,10);
  ```

  Remember the semicolon or else you will have quite a few numbers spewing forth.

- *Analyzing the data.* As we saw in the last lab, the peaks in the Fourier transform relate to the eigenvalues of the system. To obtain a plot of the Fourier transform, try the following:

  ```
  samp_per_sec = 5e4;
  duration = 10;
  [x,sr] = getData(samp_per_sec, duration);
  N = length(x);
  semilogy([0:N-1]/duration,abs(fft(x)));
  xlim([0 200]);
  ```

Noise in the Fourier transform comes in the form of a jittery magnitudes. Typically the first channel (blue) is cleaner than the second (green), an artifact of the particular photodetectors used in our assembly.

- *Which peaks?* Our measurements do not actually reflect a discretely beaded string, but a continuous string with non-uniform mass distribution. Because of that, we see many more peaks than we expect. Peaks coming from the beaded component tend to have larger magnitudes and wider bases (because they decay). However, one good way to tell which peaks are the right ones is to cheat: compare the experimental data to the frequencies you predict from the eigenvalues of $M^{-1}K$.

▶ Miscellaneous Notes on Using the String Lab

- Please do not move the collet plates, as they are delicately placed.
- The current string length is 42.125 inches.
- If your string breaks and another is not available, contact the lab assistant who will make another one. (The string diameter so closely fits the beads that the crimping that inevitably occurs when you cut a string must be milled off.)
- Occasionally MATLAB will pick up an error when you close `scope`, and this will prevent you from using any other data acquisition script. If this happens, restart MATLAB.

▶ Laboratory Instructions

1. Select $n = 2$ beads, weigh them on the lab scale, set them up as described above.
2. Pluck the string several times, each time recording the eigenvalues and the fundamental frequencies obtained from the `getData` script.
3. Compare these values to the eigenvalues that you would expect from the mathematical derivation described above.
4. Repeat steps 1–3 for $n = 4$ beads.

Lab 10: Hearing the Beads on a String

Our last lab studied the vibrations of a massless string loaded with beads, fixed at both ends. Given the location of these beads and their masses, we saw how to predict to good accuracy the frequencies of vibration. Now we turn the tables: if we know those frequencies, what can we say about the bead locations and the masses? The answer was discovered by Mark Krein about sixty years ago. Once more, our guide to the mathematics is the book by Gantmacher and Krein, *Oscillation Matrices and Kernels and Small Vibrations of Mechanical Systems* (AMS, 2002); see Supplement II.

We begin with the same physical setting as the last lab: a massless string of length ℓ on which we suspend n beads with masses m_1, m_2, …, m_n, held at the uniform tension σ. We assume that the string is lightly plucked, allowing us to approximate that the beads do not move horizontally, only vertically. Let $y_j(t)$ denote that transverse displacement of mass m_j at time t.

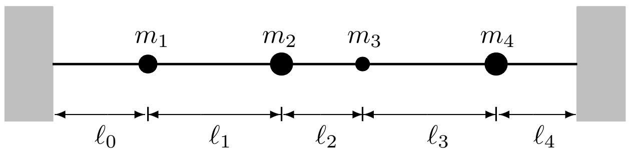


Figure 10.1: A string with four beads at rest.

▶ Construction of Fundamental Polynomials

We approach this problem by first building two sets of polynomials that will play a central role. In the last lab, we saw that the bead vibrations were governed by the eigenvalues and eigenvectors of the matrix $M^{-1}K$, that is, we seek values λ and vectors u such that $M^{-1}Ku = \lambda u$, i.e.,

$$Ku = \lambda Mu. \tag{10.1}$$

Here $u \in \mathbb{R}^n$ is a vector, and it will be most convenient if we change our notation slightly from the last lab: u_j shall denote the jth entry in the vector u, rather than the jth eigenvector. It will also be helpful for us to refer to the displacements at the left and right ends of the string by u_0 and u_{n+1}. Using this notation, we can write out the jth row of equation (10.1) as

$$\Big(-\frac{\sigma}{\ell_{j-1}}\Big)u_{j-1} + \Big(\frac{\sigma}{\ell_{j-1}} + \frac{\sigma}{\ell_j}\Big)u_j + \Big(-\frac{\sigma}{\ell_j}\Big)u_{j+1} = \lambda m_j u_j. \tag{10.2}$$

We always assume that the left end of the string is fixed:

$$u_0 = 0.$$

Given the kinds of matrices that M and K are, one can show that an eigenvector u of $M^{-1}K$ cannot have any zero entries: $u_j \neq 0$ for $j = 1, \dots, n$. In particular,

$$u_1 \neq 0.$$

Now notice that we can rearrange equation (10.2) into the form

$$u_{j+1} = \Big(-\frac{\ell_j}{\ell_{j-1}}\Big)u_{j-1} + \Big(1 + \frac{\ell_j}{\ell_{j-1}} - \frac{\lambda \ell_j m_j}{\sigma}\Big)u_j. \tag{10.3}$$

Since we know values for u_0 and u_1, we can use this last formula to produce u_2, then to produce u_3, and so on: we can construct eigenvectors one entry at a time! The only trick is that we must know an eigenvalue, λ.

What goes wrong if λ is *not* and eigenvalue, but we still use $u_0 = 0$, $u_1 = 1$ and the recurrence formula (10.3) to generate an "eigenvector"? Everything will work out just fine until we compute u_{n+1}. The fact that our string is fixed at the right end requires that $u_{n+1} = 0$, but for arbitrary values of λ the recurrence (10.3) will generally produce a nonzero value for u_{n+1}, indicating that λ is not a true eigenvalue and $[u_1, \dots, u_n]^T$ is not a true eigenvector of $M^{-1}K$. If we compute $u_{n+1} = 0$ from (10.3), then the value of λ used must indeed be an eigenvalue.

This suggests a procedure for computing eigenvalues known as the *shooting method*: assume initial values $u_0 = 0$ and $u_1 = 1$, and adjust λ until (10.3) produces $u_{n+1} = 0$. (The method gets its name from the act of progressively adjusting the angle of a cannon barrel to zero a projectile in on a target.) Rather than simply guessing values of λ, we can get a nicer characterization through the clever use of polynomials.

When $j = 1$, the formula (10.3) implies

$$\begin{aligned} u_2 &= \Big(-\frac{\ell_1}{\ell_0}\Big)u_0 + \Big(1 + \frac{\ell_1}{\ell_0} - \frac{\lambda \ell_1 m_1}{\sigma}\Big)u_1 \\ &= \Big(1 + \frac{\ell_1}{\ell_0} - \frac{\lambda \ell_1 m_1}{\sigma}\Big)u_1 \\ &= p_1(\lambda)u_1, \end{aligned}$$

where

$$p_1(\lambda) := \Big(1 + \frac{\ell_1}{\ell_0}\Big) - \lambda\Big(\frac{\ell_1 m_1}{\sigma}\Big)$$

is a linear polynomial in λ. For shorthand, we write $p_1 \in \mathcal{P}_1$, where $\mathcal{P}_k$ denotes the space of polynomials of degree k or less. Similarly, we have

$$u_3 = \Big(-\frac{\ell_2}{\ell_1}\Big)u_1 + \Big(1 + \frac{\ell_2}{\ell_1} - \frac{\lambda \ell_2 m_2}{\sigma}\Big)u_2$$

$$\begin{aligned} &= \Big(-\frac{\ell_2}{\ell_1}\Big)u_1 + \Big(1+\frac{\ell_2}{\ell_1} - \frac{\lambda\ell_2 m_2}{\sigma}\Big)p_1(\lambda)u_1 \\ &= p_2(\lambda)u_1, \end{aligned}$$

where

$$p_2(\lambda) := \Big(-\frac{\ell_2}{\ell_1}\Big) + \Big[\Big(1+\frac{\ell_2}{\ell_1}\Big) - \lambda\Big(\frac{\ell_2 m_2}{\sigma}\Big)\Big]p_1(\lambda),$$

which is a quadratic polynomial: $p_2 \in \mathcal{P}_2$.

We can continue this process for the subsequent entries in the eigenvector:

$$\begin{aligned} u_{j+1} &= \Big(-\frac{\ell_j}{\ell_{j-1}}\Big)u_{j-1} + \Big(1+\frac{\ell_j}{\ell_{j-1}} - \frac{\lambda\ell_j m_j}{\sigma}\Big)u_j \\ &= \Big(-\frac{\ell_j}{\ell_{j-1}}\Big)p_{j-2}(\lambda)u_1 + \Big(1+\frac{\ell_j}{\ell_{j-1}} - \frac{\lambda\ell_j m_j}{\sigma}\Big)p_{j-1}(\lambda)u_1 \\ &= p_j(\lambda)u_1, \end{aligned}$$

where

$$p_j(\lambda) := \Big(-\frac{\ell_j}{\ell_{j-1}}\Big)p_{j-2}(\lambda) + \Big[\Big(1+\frac{\ell_j}{\ell_{j-1}}\Big) - \lambda\Big(\frac{\ell_j m_j}{\sigma}\Big)\Big]p_{j-1}(\lambda).$$

(Note that $p_j \in \mathcal{P}_j$, and that this recurrence works also for $j = 2$ if we define $p_0(\lambda) = 1$ for all λ.) Continuing this process, we eventually arrive at

$$u_{n+1} = p_n(\lambda)u_1,$$

where $p_n \in \mathcal{P}_n$. Since $u_1 \neq 0$, we see that

$$u_{n+1} = 0 \qquad \text{if and only if} \qquad p_n(\lambda) = 0.$$

In other words

$$\lambda \text{ is an eigenvalue of } M^{-1}K \qquad \text{if and only if} \qquad p_n(\lambda) = 0.$$

Since p_n is a polynomial of degree n, it has precisely n roots. We can build p_n by following the recurrence specified above, which requires us to know the lengths $\{\ell_j\}_{j=0}^n$ and masses $\{m_j\}_{j=1}^n$. On the other hand, we can always build a polynomial directly from its roots, up to a constant factor. In other words, if we know the eigenvalues $\lambda_1, \ldots, \lambda_n$ (e.g., by physical measurement), then we know that

$$p_n(\lambda) = \gamma\prod_{j=1}^{n}(\lambda - \lambda_j),$$

for some constant γ.

Suppose that we know the eigenvalues $\lambda_1, \ldots, \lambda_n$, and use them to build the polynomial p_n. Can we reverse engineer the process described above to discover the lengths and masses? In general, the answer is *no* – and this should not be entirely surprising, as we are trying to extract $2n + 1$

pieces of information from $n+1$ pieces of data (the n eigenvalues and the total length, ℓ). To make the problem well determined, we need n more pieces of data.

One could imagine various sources for this data (e.g., some information about the eigenvectors). It shall ultimately prove most convenient for us to obtain another set of eigenvalues from a closely related problem: keep the masses and lengths the same and the left end fixed, but now allow the right end to move in such a way that the string has *zero slope* at the end, i.e., $u_{n+1} = u_n$. You might naturally wonder whether such a condition could be reliably implemented in practice (e.g., by attaching the right end of the string to a ring mounted in a frictionless fashion to a pole ...), but suspend disbelief for the moment. (We will eventually find a sneaky way to get this data.)

The 'zero slope' condition over the last segment of the string will prove to be very useful. In particular, we shall develop a recurrence akin to (10.3) for the slopes, then use this to derive a polynomial whose roots correspond to those values of λ for which the eigenvector u satisfies the zero slope condition.

To start, rearrange (10.3) to obtain a relation between consecutive slopes:

$$\frac{u_{j+1}-u_j}{\ell_j} = \frac{u_j - u_{j-1}}{\ell_{j-1}} - \Big(\frac{\lambda m_j}{\sigma}\Big)u_j. \tag{10.4}$$

We can use the polynomials $\{p_j\}$ to write the left hand side of (10.4) in the form

$$\frac{u_{j+1}-u_j}{\ell_j} = \frac{p_j(\lambda)u_1 - p_{j-1}(\lambda)u_1}{\ell_j}.$$

Since $p_j \in \mathcal{P}_j$ and $p_{j-1} \in \mathcal{P}_{j-1}$, the right hand side is a polynomial in $\mathcal{P}_j$, which we define to be

$$q_j(\lambda) := \frac{1}{\ell_j}(p_j(\lambda) - p_{j-1}(\lambda)), \tag{10.5}$$

for $j = 1, \ldots, n$, so that $q_j(\lambda)u_1$ denotes the slope induced by eigenvector components u_j and u_{j+1}. The right hand side of (10.4) can then be written as

$$\frac{u_j - u_{j-1}}{\ell_{j-1}} - \Big(\frac{\lambda m_j}{\sigma}\Big)u_j = q_{j-1}(\lambda)u_1 - \Big(\frac{\lambda m_j}{\sigma}\Big)p_{j-1}(\lambda)u_1.$$

Equating our expressions for the left and right sides of (10.4), we obtain

$$q_j(\lambda)u_1 = q_{j-1}(\lambda)u_1 - \Big(\frac{\lambda m_j}{\sigma}\Big)p_{j-1}(\lambda)u_1,$$

and since $u_1 \neq 0$, we arrive at a relation between the slope and displacement polynomials:

$$q_j(\lambda) = q_{j-1}(\lambda) - \Big(\frac{\lambda m_j}{\sigma}\Big)p_{j-1}(\lambda). \tag{10.6}$$

A similar relation follows from rearranging the definition of the slope polynomials, (10.5):

$$p_j(\lambda) = \ell_j q_j(\lambda) + p_{j-1}(\lambda). \tag{10.7}$$

For the string with the zero-slope boundary condition on the right, we must have

$$q_n(\lambda)u_1 = 0,$$

and hence λ must be a root of the polynomial $q_n(\lambda)$. Thus, knowledge of the n eigenvalues of the string with zero-slope boundary condition on the right allows us to construct the polynomial q_n up to a scaling factor.

Now we shall see how to determine material properties of the beaded string from the polynomials p_n and q_n. Using the recurrences for the slope and displacement polynomials, we have

$$\begin{aligned}
\frac{p_n(\lambda)}{q_n(\lambda)} &= \frac{\ell_n q_n(\lambda) + p_{n-1}(\lambda)}{q_n(\lambda)} && \text{[by (10.7)]} \\
&= \ell_n + \frac{p_{n-1}(\lambda)}{q_n(\lambda)} \\
&= \ell_n + \cfrac{1}{\cfrac{q_n(\lambda)}{p_{n-1}(\lambda)}} \\
&= \ell_n + \cfrac{1}{\cfrac{-(m_n/\sigma)\lambda p_{n-1}(\lambda) + q_{n-1}(\lambda)}{p_{n-1}(\lambda)}} && \text{[by (10.6)]} \\
&= \ell_n + \cfrac{1}{-(m_n/\sigma)\lambda + \cfrac{1}{\cfrac{p_{n-1}(\lambda)}{q_{n-1}(\lambda)}}} && (10.8) \\
&= \ell_n + \cfrac{1}{-(m_n/\sigma)\lambda + \cfrac{1}{\cfrac{\ell_{n-1}q_{n-1}(\lambda) + p_{n-2}(\lambda)}{q_{n-1}(\lambda)}}} && \text{[by (10.7)]} \\
&\ \vdots \\
&= \ell_n + \cfrac{1}{-(m_n/\sigma)\lambda + \cfrac{1}{\ell_{n-1} + \cfrac{1}{-(m_{n-1}/\sigma)\lambda + \cdots + \cfrac{1}{\ell_1 + \cfrac{1}{-(m_1/\sigma)\lambda + \cfrac{1}{\ell_0}}}}}}, && (10.9)
\end{aligned}$$

which is called a *continued fraction* form of the rational function p_n/q_n, and from this beautiful decomposition we can simply read off the masses and string lengths.

To summarize: the eigenvalues of the original system give us the polynomial p_n up to a scaling factor; the eigenvalues of the system with zero-slope on the right end give us q_n up to a scaling factor. To construct the continued fraction decomposition (10.9), we used knowledge of the subordinate polynomials p_{n-1}, q_{n-1}, $\ldots$, which we could not immediately construct directly from the two sets of eigenvalues. Thus we seek some method for computing the continued fraction decomposition (10.9) that does not require us to know p_{n-1}, q_{n-1}, etc. If we had a recipe for constructing this decomposition, then we could then read off the values of ℓ_j and m_j/σ. These would be off by a scaling factor inherited from p_n and q_n, but that can be resolved from our knowledge of the total length ℓ: we must have $\sum_{j=0}^{n} \ell_j = \ell$. Our next goal is to determine an algorithm that will deliver the continued fraction decomposition.

▶ A Recipe Recovering Material Parameters from Polynomials

Suppose we have the measured the eigenvalues of the string with both ends fixed,

$$\lambda_1 < \lambda_2 < \cdots < \lambda_n,$$

and the eigenvalues of the string with fixed left end and a zero-slope on the right,

$$\widehat{\lambda}_1 < \widehat{\lambda}_2 < \cdots < \widehat{\lambda}_n.$$

With these eigenvalues, we can construct a pair of degree-n polynomials,

$$a_n(\lambda) = \prod_{j=1}^{n} (\lambda - \lambda_j) \;=\; \lambda^n + \alpha_1 \lambda^{n-1} + \alpha_2 \lambda^{n-2} + \cdots + \alpha_{n-1}\lambda + \alpha_n \tag{10.10}$$

$$b_n(\lambda) = \prod_{j=1}^{n} (\lambda - \widehat{\lambda}_j) \;=\; \lambda^n + \beta_1 \lambda^{n-1} + \beta_2 \lambda^{n-2} + \cdots + \beta_{n-1}\lambda + \beta_n. \tag{10.11}$$

Because a_n and b_n of these have a coefficient of one multiplying the leading term, λ^n, they are called *monic polynomials.* These polynomials equal p_n and q_n up to constants, and so the ratio a_n/b_n differs from p_n/q_n only by a constant, which we shall call γ_n:

$$\gamma_n \frac{p_n(\lambda)}{q_n(\lambda)} = \frac{a_n(\lambda)}{b_n(\lambda)}.$$

Our goal is to determine a continued fraction decomposition of a_n/b_n that will expose the material properties of our beaded string. In particular, we seek to write the ratio of the degree n monic polynomials a_n and b_n as continued fractions involving degree $n-1$ monic polynomials a_{n-1} and b_{n-1}. In particular, we shall seek ξ_n and γ_{n-1} so that we can write

$$\gamma_n \frac{p_n(\lambda)}{q_n(\lambda)} \;=\; \frac{a_n(\lambda)}{b_n(\lambda)} \;=\; 1 + \cfrac{1}{-\xi_n \lambda + \cfrac{1}{\gamma_{n-1}^{-1} \dfrac{a_{n-1}(\lambda)}{b_{n-1}(\lambda)}}}.$$

Once we know how to compute this decomposition, we can apply the same ideas to the ratio a_{n-1}/b_{n-1}, and so on, until we arrive at the monic constants, $a_0(\lambda) = 1$ and $b_0(\lambda) = 1$. At this stage, we will have a decomposition that matches (10.9) up to a constant.

Before undertaking the somewhat intricate derivation of this decomposition, we summarize the results we shall soon justify in the form of recipe for recovering lengths and masses for the eigenvalues $\lambda_1, \ldots, \lambda_n$ and $\widehat{\lambda}_1, \ldots, \widehat{\lambda}_n$.
A note about notation: In what follows, $\mathbf{a}_1$ refers to the first element of the vector $\mathbf{a}$, $\mathbf{a}_{2:k}$ refers to vector consisting of the second through kth elements of $\mathbf{a}$, and similarly for $\mathbf{b}_1$ and $\mathbf{b}_{2:k}$.

```
Construct vectors $\mathbf{a} := [\alpha_1, \alpha_2, \ldots, \alpha_n]$ and $\mathbf{b} := [\beta_1, \beta_2, \ldots, \beta_n]$ from (10.10) and (10.11).
for $k := n, n-1, \ldots, 1$
    $\xi_k := 1/(\mathbf{b}_1 - \mathbf{a}_1)$
    $\mathbf{a} := \mathbf{a} - \mathbf{b}$
    $\mathbf{b} := \mathbf{b} + \xi_k[\mathbf{a}_{2:k}\ 0]$
    $\gamma_{k-1} := \mathbf{b}_1/\mathbf{a}_1$
    $\mathbf{a} := \mathbf{a}_{2:k}/\mathbf{a}_1$
    $\mathbf{b} := \mathbf{b}_{2:k}/\mathbf{b}_1$
end
$\gamma_n := \frac{1}{\ell}\Big(1 + \frac{1}{\gamma_{n-1}} + \frac{1}{\gamma_{n-2}\gamma_{n-1}} + \cdots + \frac{1}{\gamma_0\gamma_1\cdots\gamma_{n-1}}\Big)$
$\ell_n := 1/\gamma_n$
for $k := n, n-1, \ldots, 1$
    $\ell_{k-1} := \ell_k/\gamma_{k-1}$.
    $m_k := \sigma\xi_k/\ell_k$
end
```

▶ A Derivation of the Continued Fraction Decomposition of a_n/b_n

Now we shall explain where this algorithm comes from. Inspired by the formula (10.8) on the last page, we would like to write

$$\frac{a_n(\lambda)}{b_n(\lambda)} = \nu_n + \cfrac{1}{-\xi_n\lambda + \cfrac{1}{\cfrac{\widehat{a}_{n-1}(\lambda)}{\widehat{b}_{n-1}(\lambda)}}}$$

for some constants ξ_n and ν_n, and some polynomials $\widehat{a}_{n-1}$ and $\widehat{b}_{n-1}$, each of degree $n-1$. Does such a decomposition exist? Indeed it does, and we shall show how to determine its various terms. Algebraically manipulate the right hand side to obtain the equivalent expression

$$\frac{a_n(\lambda)}{b_n(\lambda)} = \nu_n + \frac{\widehat{a}_{n-1}(\lambda)}{-\xi_n\lambda\widehat{a}_{n-1}(\lambda) + \widehat{b}_{n-1}(\lambda)}. \tag{10.12}$$

Our first goal is to determine ν_n. The definitions of a_n and b_n imply that the left hand side of (10.12) tends to 1 as $\lambda \to \infty$, so the same would need to be true of the right hand side. Given that $\widehat{a}_{n-1}$

and $\widehat{b}_{n-1}$ are of degree $n-1$, the second term on the right of (10.12) goes to zero as $\lambda \to \infty$, so we must conclude that

$$\nu_n = 1.$$

This would then imply that

$$\frac{a_n(\lambda) - b_n(\lambda)}{b_n(\lambda)} = \frac{\widehat{a}_{n-1}(\lambda)}{\widehat{b}_{n-1}(\lambda) - \xi_n \lambda \widehat{a}_{n-1}(\lambda)}.$$

By equating the numerators and denominators, this formula suggests that we *define*

$$\begin{aligned} \widehat{a}_{n-1}(\lambda) &:= a_n(\lambda) - b_n(\lambda) \\ \widehat{b}_{n-1}(\lambda) &:= b_n(\lambda) + \xi_n \lambda \widehat{a}_{n-1}(\lambda). \end{aligned}$$

Do these definitions jibe with our demand that both $\widehat{a}_{n-1}$ and $\widehat{b}_{n-1}$ be degree $n-1$ polynomials? Since a_n and b_n have λ^n as a common leading term, the definition of $\widehat{a}_{n-1}$ ensures that its λ^n coefficient is zero, and hence $\widehat{a}_{n-1}$ has degree $n-1$. Now we check $\widehat{b}_{n-1}$:

$$\begin{aligned} \widehat{b}_{n-1}(\lambda) &= b_n(\lambda) + \xi_n \lambda \big(a_n(\lambda) - b_n(\lambda)\big) \\ &= \big(\lambda^n + \beta_1 \lambda^{n-1} + \cdots + \beta_n\big) + \xi_n \lambda \big((\alpha_1 - \beta_1)\lambda^{n-1} + (\alpha_2 - \beta_2)\lambda^{n-2} + \cdots + (\alpha_n - \beta_n)\big) \\ &= \big(1 + \xi_n(\alpha_1 - \beta_1)\big)\lambda^n + \big(\beta_1 + \xi_n(\alpha_2 - \beta_2)\big)\lambda^{n-1} + \cdots + \big(\beta_{n-1} + \xi_n(\alpha_n - \beta_n)\big)\lambda + \beta_n. \end{aligned}$$

This polynomial will have degree $n-1$ provided the coefficient of λ^n is zero, which we can ensure by assigning the only remaining free parameter to be

$$\xi_n := \frac{1}{\beta_1 - \alpha_1}.$$

In summary, we have determined that we can find constant ξ_n and degree $n-1$ polynomials $\widehat{a}_{n-1}$ and $\widehat{b}_{n-1}$ so that

$$\gamma_n \frac{p_n(\lambda)}{q_n(\lambda)} = \frac{a_n(\lambda)}{b_n(\lambda)} = 1 + \cfrac{1}{-\xi_n \lambda + \cfrac{1}{\cfrac{\widehat{a}_{n-1}(\lambda)}{\widehat{b}_{n-1}(\lambda)}}}.$$

To get the full continued fraction that reveals all the lengths and masses as in (10.9), we would like to apply this procedure recursively to $\widehat{a}_{n-1}/\widehat{b}_{n-1}$, but one small adjustment is needed. Recall that we started with polynomials a_n and b_n that had leading term λ^n; in general, $\widehat{a}_{n-1}$ and $\widehat{b}_{n-1}$ will have nontrivial coefficients multiplying λ^{n-1}, as given by

$$\begin{aligned} \widehat{a}_{n-1}(\lambda) &= \big(\alpha_1 - \beta_1\big)\lambda^{n-1} + \cdots + \big(\alpha_{n-1} - \beta_{n-1}\big)\lambda + \big(\alpha_n - \beta_n\big) \\ \widehat{b}_{n-1}(\lambda) &= \big(\beta_1 + \xi_n(\alpha_2 - \beta_2)\big)\lambda^{n-1} + \cdots + \big(\beta_{n-1} + \xi_n(\alpha_n - \beta_n)\big)\lambda + \beta_n, \end{aligned}$$

and these leading coefficients must be scaled out. Define a constant to capture the ratio of these coefficients,

$$\gamma_{n-1} := \frac{\beta_1 + \xi_n(\alpha_2 - \beta_2)}{\alpha_1 - \beta_1},$$

so that if we define

$$\begin{aligned} a_{n-1}(\lambda) &:= \frac{\widehat{a}_{n-1}(\lambda)}{\alpha_1 - \beta_1} \\ &= \lambda^{n-1} + \frac{\alpha_2 - \beta_2}{\alpha_1 - \beta_1}\lambda^{n-2} + \cdots + \frac{\alpha_n - \beta_n}{\alpha_1 - \beta_1} \end{aligned} \tag{10.13}$$

$$\begin{aligned} b_{n-1}(\lambda) &:= \frac{\widehat{b}_{n-1}(\lambda)}{\beta_1 + \xi_n(\alpha_2 - \beta_2)} \\ &= \lambda^{n-1} + \frac{\beta_2 + \xi_n(\alpha_3 - \beta_3)}{\beta_1 + \xi_n(\alpha_2 - \beta_2)}\lambda^{n-2} + \cdots + \frac{\beta_{n-1} + \xi_n(\alpha_n - \beta_n)}{\beta_1 + \xi_n(\alpha_2 - \beta_2)}\lambda + \frac{\beta_n}{\beta_1 + \xi_n(\alpha_2 - \beta_2)}, \end{aligned} \tag{10.14}$$

then

$$\gamma_{n-1}\frac{\widehat{a}_{n-1}(\lambda)}{\widehat{b}_{n-1}} = \frac{a_{n-1}(\lambda)}{b_{n-1}(\lambda)}.$$

Using this expression, our evolving continued fraction takes the form

$$\gamma_n \frac{p_n(\lambda)}{q_n(\lambda)} = \frac{a_n(\lambda)}{b_n(\lambda)} = 1 + \cfrac{1}{-\xi_n\lambda + \cfrac{1}{\gamma_{n-1}^{-1}\cfrac{a_{n-1}(\lambda)}{b_{n-1}(\lambda)}}}.$$

Note that the coefficients of a_{n-1} and b_{n-1} can be extracted directly from those of a_n and b_n by way of the formulas (10.13) and (10.14).

Now apply the procedure we have just described to a_{n-1}/b_{n-1} to obtain the decomposition

$$\frac{a_{n-1}(\lambda)}{b_{n-1}(\lambda)} = 1 + \cfrac{1}{-\xi_{n-1}\lambda + \cfrac{1}{\gamma_{n-2}^{-1}\cfrac{a_{n-2}(\lambda)}{b_{n-2}(\lambda)}}},$$

and continue this process until one finally arrives at $a_0(\lambda)/b_0(\lambda)$: this term must be trivial, since, by definition, $a_0(\lambda) = 1$ and $b_0(\lambda) = 1$ for all λ.

This construction might well seem extremely intricate, but if you keep a clear head you will find that it becomes rather mechanical. To illustrate this claim, we will work through an example for the $n = 2$ case. Once you see these details, larger values of n will be a snap.

Before we proceed to this concrete example, we should address one or two fine points that might have crossed your mind. (1) Can we be certain that this procedure does not *break down*? That is, can we be sure that we never encounter $\gamma_j = 0$, which would give a division by zero? (2) Are the choices for ξ_j and γ_j unique, or are there other equivalent continued fraction decompositions of p_n/q_n with different constants? For our beaded strings, these issues conveniently will never arise: the procedure never breaks down, and the decomposition is unique. The details are beyond the scope of these denote; for more information, consult Gantmacher and Krein (Supplement II, especially the lemma on page 284). (3) How robust is this procedure to small errors in the data, i.e., the eigenvalues? You could explore this fascinating question by conducting experiments with synthetic data: exact eigenvalues computed from the $M^{-1}K$ matrix we constructed in the last lab.

▶ Special case: $\boldsymbol{n = 2}$

In the case of a string with two beads, we twice apply the procedure outlined above to obtain

$$\gamma_2 \frac{p_2(\lambda)}{q_2(\lambda)} = \frac{a_2(\lambda)}{b_2(\lambda)} = 1 + \cfrac{1}{-\xi_2\lambda + \cfrac{1}{\gamma_1^{-1}\dfrac{a_1(\lambda)}{b_1(\lambda)}}} = 1 + \cfrac{1}{-\xi_2\lambda + \cfrac{1}{\gamma_1^{-1}\left[1 + \cfrac{1}{-\xi_1\lambda + \cfrac{1}{\gamma_0^{-1}\dfrac{a_0(\lambda)}{b_0(\lambda)}}}\right]}}$$

$$= 1 + \cfrac{1}{-\xi_2\lambda + \cfrac{1}{\gamma_1^{-1}\left[1 + \cfrac{1}{-\xi_1\lambda + \cfrac{1}{\gamma_0^{-1}}}\right]}}$$

$$= 1 + \cfrac{1}{-\xi_2\lambda + \cfrac{1}{\gamma_1^{-1} + \cfrac{1}{-\xi_1\gamma_1\lambda + \cfrac{1}{\gamma_0^{-1}\gamma_1^{-1}}}}}.$$

Dividing through by the as yet *unknown* constant γ_2, we obtain

$$\frac{p_2(\lambda)}{q_2(\lambda)} = \gamma_2^{-1} + \cfrac{1}{-\xi_2\gamma_2\lambda + \cfrac{1}{\gamma_1^{-1}\gamma_2^{-1} + \cfrac{1}{-\xi_1\gamma_1\gamma_2\lambda + \cfrac{1}{\gamma_0^{-1}\gamma_1^{-1}\gamma_2^{-1}}}}}. \tag{10.15}$$

Comparing this term-by-term with the formula (10.9), we identify

$$\ell_2 = \frac{1}{\gamma_2}, \qquad \ell_1 = \frac{1}{\gamma_1\gamma_2}, \qquad \ell_0 = \frac{1}{\gamma_0\gamma_1\gamma_2},$$

$$m_2 = \sigma\xi_2\gamma_2, \qquad m_1 = \sigma\xi_1\gamma_1\gamma_2.$$

Even this simple case provides enough clues for you to guess general formulas for $n > 2$.

Given the eigenvalues $\lambda_1, \lambda_2, \widehat{\lambda}_1$, and $\widehat{\lambda}_2$, we can determine the constants ξ_2, γ_2, ξ_1, γ_1, and γ_0 through the procedure outlined above. Furthermore, we presume that we can measure the tension σ from the force transducer in the laboratory, and that we also know the total length of the string, $\ell = \ell_0 + \ell_1 + \ell_2$. This last expression finally gives us a formula for last remaining unknown, γ_2: since

$$\ell = \frac{1}{\gamma_2} + \frac{1}{\gamma_1\gamma_2} + \frac{1}{\gamma_0\gamma_1\gamma_2},$$

we conclude that

$$\gamma_2 = \frac{1}{\ell}\left(1 + \frac{1}{\gamma_1} + \frac{1}{\gamma_0\gamma_1}\right).$$

With γ_2 in hand, we can compute all the lengths and masses.

▶ Symmetrically Loaded Strings

Having seen how to determine the bead locations and masses from two sets of eigenvalues, we now must address the practical issue of experimentally measuring those eigenvalues. When the string is fixed at both ends, we can approximate the eigenvalues using the reliable experiments performed in the last lab. It is much more difficult to work with a string with fixed left end and zero slope condition on the right. (We shall call such a string 'fixed–flat' to distinguish it from the usual 'fixed-fixed' string.) Fortunately, there is a interesting special case for which we can determine these eigenvalues without any modification to the experimental apparatus.

Suppose that the number of beads, N, is even, and the masses of the beads and lengths between them are symmetric about the midpoint of the string. To distinguish this case from the general scenario considered earlier, we will denote these masses by $M_1, \ldots, M_N$, and the lengths by $L_0, \ldots, L_N$, with the total length $L = \sum_{j=0}^{N} L_j$. Thus, the symmetric arrangement requires that

$$\begin{aligned} M_1 &= M_N & L_0 &= L_N \\ M_2 &= M_{N-1} & L_1 &= L_{N-1} \\ &\vdots & &\vdots \\ M_{N/2} &= M_{N/2+1}, & L_{N/2-1} &= L_{N/2+1} \end{aligned}$$

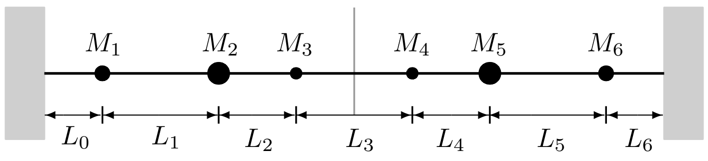


Figure 10.2: A string with six beads with lengths and masses arranged symmetrically about the middle of the string, denoted by the vertical gray line.

as illustrated for $N = 6$ in Figure 10.2. We can set up such a symmetric arrangement on our laboratory's monochord, and then experimentally measure the eigenvalues when both ends of the string are fixed; let us label these eigenvalues $\Lambda_1, \Lambda_2, \ldots, \Lambda_N$. In Figure 10.3 we show the eigenvectors for the configuration shown in Figure 10.2.

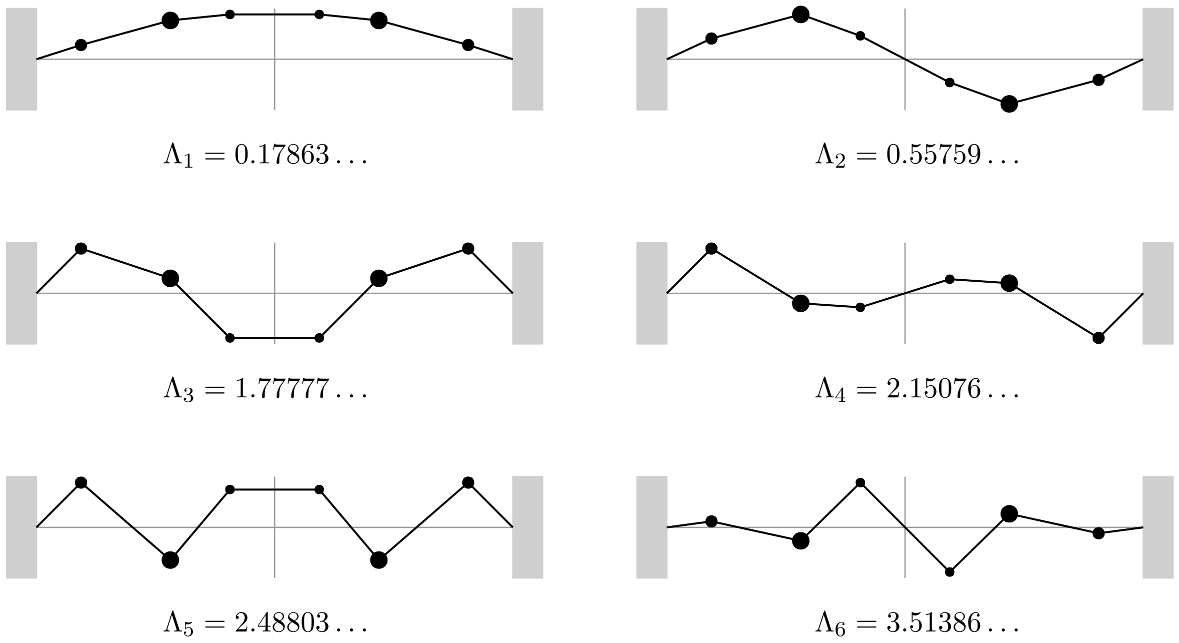

$\Lambda_1 = 0.17863\ldots$
$\Lambda_2 = 0.55759\ldots$
$\Lambda_3 = 1.77777\ldots$
$\Lambda_4 = 2.15076\ldots$
$\Lambda_5 = 2.48803\ldots$
$\Lambda_6 = 3.51386\ldots$

Figure 10.3: Eigenvectors for the string shown in Figure 10.2. In each plot, the vertical displacement of the jth mass indicates the jth entry of the eigenvector of $M^{-1}K$.

These eigenvectors reveal a remarkable property: eigenvectors corresponding to odd eigenvalues Λ_1, Λ_3, and Λ_5 are all *symmetric* about the middle of the string: If we cut the string in half, these eigenvectors would be fixed on the left end and have *zero slope* at the right. On the other hand, eigenvectors corresponding to even eigenvalues Λ_2, Λ_4, and Λ_6 are all *antisymmetric* about the midpoint: If we cut the string in half, these eigenvectors would be fixed at zero at both ends. These observations hold for all symmetric string configurations, and they hint at a key fact:

> *The N eigenvalues of a symmetric beaded string fixed at both ends exactly match the $N/2$ fixed-fixed and $N/2$ fixed-flat eigenvalues associated with half of the string.*

More precisely, consider the string of length $\ell = L/2$ loaded with $n = N/2$ beads with masses

$$m_j = M_j, \quad j = 1, \ldots, n$$

separated by lengths

$$\begin{aligned} \ell_j &= L_j, \quad j = 0, \ldots, n-1 \\ \ell_n &= L_n/2. \end{aligned}$$

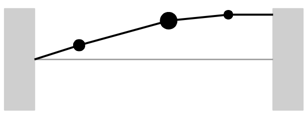
The odd eigenvalues for the symmetric string match the eigenvalues for the fixed-flat half string,

$$\widehat{\lambda}_j = \Lambda_{2j-1}, \quad j = 1, \ldots, n$$

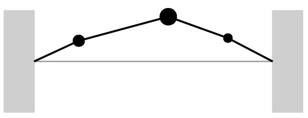
while the even eigenvalues for the symmetric string match the eigenvalues for the fixed-fixed half string,

$$\lambda_j = \Lambda_{2j}, \quad j = 1, \ldots, n.$$

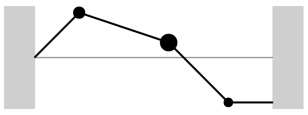
Figure 10.4 shows eigenvectors for the half-string corresponding to the symmetric string in Figure 10.2. The fixed-flat eigenvectors are shown in the left column; they correspond to the left halves of the eigenvectors show on the left of Figure 10.3, and the eigenvalues match Λ_1, Λ_3, and Λ_5 exactly. Similarly, the fixed-fixed eigenvectors on the half-string, shown on the right of Figure 10.4, equal the left halves of the eigenvectors on the right of Figure 10.3, and the eigenvalues match Λ_2, Λ_4, and Λ_6.



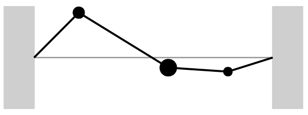
$\widehat{\lambda}_1 = 0.17863\ldots$



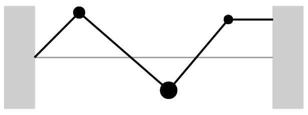
$\lambda_1 = 0.55759\ldots$



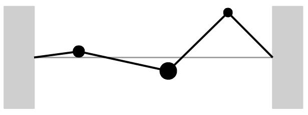
$\widehat{\lambda}_2 = 1.77777\ldots$



$\lambda_2 = 2.15076\ldots$



$\widehat{\lambda}_3 = 2.48803\ldots$



$\lambda_3 = 3.51386\ldots$

Figure 10.4: Eigenvectors for the half-string corresponding to the symmetric configuration in Figure 10.2. The eigenvectors on the left satisfy the fixed-flat conditions; those on the right satisfy fixed-fixed conditions. Compare to the eigenvectors for the symmetric string if Figure 10.3.

We have arrived at a method for obtaining the data required for the inversion procedure described in the last section, *provided our string has symmetrically arranged beads.*

1. Pluck the symmetric string and record the $N = 2n$ eigenvalues $\Lambda_1, \ldots, \Lambda_{2n}$.

2. Relabel these eigenvalues for the corresponding half-string: $\lambda_j = \Lambda_{2j}$ and $\widehat{\lambda}_j = \Lambda_{2j-1}$ for $j = 1, \ldots, n$.

3. Apply the inversion method to half-string eigenvalues to obtain $\ell_0, \ldots, \ell_n$ and $m_1, \ldots, m_n$.

4. Recover the parameters for the original symmetric string:

$$L_0 = L_N = \ell_0, \quad \ldots \quad L_{n-1} = L_{n+1} = \ell_{n-1}, \quad L_n = 2\ell_n;$$

$$M_1 = M_N = m_1, \quad \ldots \quad M_n = M_{n+1} = m_n.$$

Finally, we are prepared to attempt this procedure on some real data.

▶ Lab Instructions

1. For $n = 1$ and $n = 3$ beads, work out the form of the continued fraction expansion of p_n/q_n involving $\xi_1, \ldots, x_n$ and $\gamma_0, \ldots, \gamma_n$ (analogous to equation (10.15)), and give formulas for $\ell_0, \ldots, \ell_n$ and $m_1, \ldots, m_n$.

2. Collect data for a string with two symmetrically arranged beads, and use the $n = 1$ formulas to determine the bead positions and masses. How well do these agree with the true values?

3. Repeat the last experiment with four symmetrically arranged beads, now using the $n = 2$ formulas given above.

Optional, but strongly recommended:

4. Develop a MATLAB script that will implement the inversion procedure for a general value of n. Generate synthetic eigenvalues for a symmetric string using the code you developed for the last lab, and use these eigenvalues to test your inversion algorithm. How robust the the process to small changes in the eigenvalues? How reliable is your method for larger values of n?