The Physics of Springs*

Heather Williamson
Aneesh Mehta
Matthew Broussard
Jordon Cavazos
Jeffrey Bridge
Steven J. Cox

Abstract

This report summarizes work done as part of the Physics of Strings PFUG under Rice University's VIGRE program. VIGRE is a program of Vertically Integrated Grants for Research and Education in the Mathematical Sciences under the direction of the National Science Foundation. A PFUG is a group of Postdocs, Faculty, Undergraduates and Graduate students formed round the study of a common problem. This module describes experiments done on a spring system.

1 A Network of Springs

Our research is centered on a network of springs, built by Jeff Hokansen and Dr. Mark Embree for use in the CAAM 335 Lab. Over the table, we set up a webcam on a beam and connected it to a computer running MATLAB. Springs are connected to pennies (nodes), two of which are fixed to the table. Along the outside pennies, strings run over pullies set along the edge of the table and are attached to hooks, upon which we hang masses. These masses cause the nodes to move. We use the webcam to capture an image of the network, then use a MATLAB script to find the center of each node; the pennies have been painted red to make it easier for MATLAB to detect them. This gives us the displacement of each node, from which we can compute the elongation of each spring. We also know the force applied to each node ($9.8 * mass$ in units of Newtons) and can calculate the spring constant k for each spring using Hooke's Law, $f_{\text{restoring}} = -\,(\text{elongation}) * k$

1.1 A Forward Problem

In the forward problem, we seek to compare results from our physical model to the results predicted by solving a linear system of equations. Specifically, we wish to predict our displacements, given we know the load forces and spring constants in our system of springs.

Let us begin with an easier system of just two springs, three nodes, and two forces. Since only two of the nodes are moving, we will have two horizontal displacements denoted in the vector x. There are two elongations, one for each spring, denoted in the vector e.

*Version 1.1: Nov 9, 2009 11:05 pm -0600
†http://creativecommons.org/licenses/by/3.0/

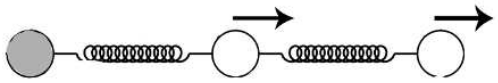


Figure 1: 2 Spring Network

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad e = \begin{bmatrix} e_1 \\ e_2 \end{bmatrix}, \tag{1}$$

Each spring elongation is a linear combination of node displacements. The equations can be written in the following manner.

$$e = \begin{bmatrix} e_1 \\ e_2 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 - x_1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} x = Ax \tag{2}$$

Now we have our adjacency matrix, A. This translates us from node displacement to spring elongation. It will have one more property which will we shall see shortly. Now let us consider finding the restoring force, y, which will have one component for each spring.

$$y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}, \tag{3}$$

We assume that each spring follows Hooke's Law, $y = ke$, where restoring force is directly proportional to elongation. Each spring has a corresponding stiffness, k_i which comprise the the diagonal elements of matrix, K.

$$y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} k_1 e_1 \\ k_2 e_2 \end{bmatrix} = \begin{bmatrix} k_1 & 0 \\ 0 & k_2 \end{bmatrix} e = Ke = KAx \tag{4}$$

The final step is to translate these restoring forces into the load forces acting on each node, denoted by vector f.

$$f = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} = \begin{bmatrix} y_1 - y_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix} = A^T y \tag{5}$$

Now we can see the second feature of the adjacency martrix. The transpose of A performs the reverse translation from edges to nodes. The final product of this example is the equation just shown: $f = A^T KAx$. Now we can expand the problem to any system of springs for which we can create an adjacency matrix A. For this project we focused on the spring network shown below.

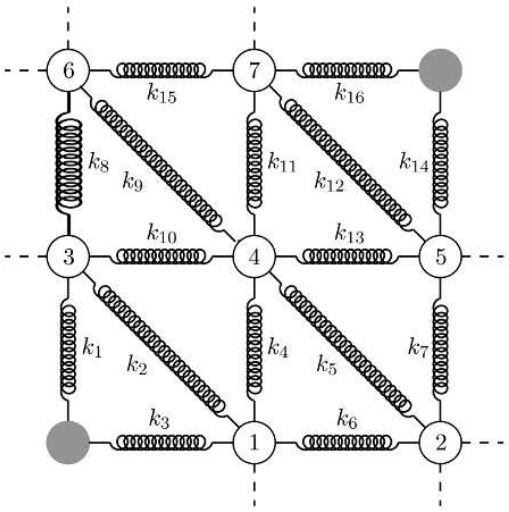


Figure 2: Spring Network

For this system, we have 16 strings and 7 seven nodes. So we will be working with vectors x, e, y, and f which contain the following information:

x: displacement of nodes

e: elongation of springs

y: restoring force in each spring

f: load force at each node

Since there are 7 nodes, each with a horizontal and vertical displacement, vectors x and f with be vectors of length 14. The 2n-1 entry will correspond to the horizontal displacement or load force on node n and the 2n entry will correspond to the vertical displacement or load force on node n. Since there are 16 springs, y and e will be vectors of length 16 with each entry corresponding to the restoring force or displacement of that spring.

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{13} \\ x_{14} \end{bmatrix}$$

$$\begin{aligned} x_1 &: \text{ horizontal displacement at Node 1} \\ x_2 &: \text{ vertical displacementd at Node 1} \\ &\vdots \\ x_{13} &: \text{ horizontal displacement at Node 7} \\ x_{14} &: \text{ vertical displacement at Node 7} \end{aligned} \tag{6}$$

As for the matrices translating us from one vector to the next, we will be working with A and K. A will have fourteen columns relating to the 14 degrees of freedom of the nodes of system, and it will have 16 rows, corresponding to the springs of the system. K will be a square matrix, 16x16. Diagonal elements of K will correspond to the stiffnesses of each spring

$$K = \begin{bmatrix} k_1 & 0 & 0 & \cdots & 0 \\ 0 & k_2 & 0 & \cdots & 0 \\ 0 & 0 & k_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & k_{16} \end{bmatrix}, \quad k_i = \text{spring constant of spring i} \tag{7}$$

To construct A, we use a pattern of three linear approximations. The first two assume that the elongation of horizontal springs will be approximately equal to the horizontal displacements of the two adjacent nodes. The same will hold true for vertical nodes. The third recurring equation in the adjacency matrix approximates the diagonal elongation with a first order Maclaurin expansion. This elongation is equal to $\sqrt{2}/2$ multiplied by the sum of the horizontal and vertical displacements. The specific adjacency matrix for this set up is the following:

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \alpha & -\alpha & 0 & 0 & -\alpha & \alpha & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \alpha & -\alpha & 0 & 0 & -\alpha & \alpha & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \alpha & -\alpha & 0 & 0 & -\alpha & \alpha & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\alpha & \alpha & 0 & 0 & \alpha & -\alpha \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \end{bmatrix}, \ \alpha = \frac{\sqrt{2}}{2},$$

Now that we have created our adjacency matrix for the system and know the relation $f = A^T K A x$, we can create the matrix defined by $A^T K A$. If this matrix is invertible, then we can solve what is referred to as the forward problem, finding x given f. This relationship is simply done by finding the inverse of our matrix and solving $x = \left(A^T K A\right)^{-1} f$.

1.2 An Inverse Problem

We move now to our inverse problem: given the applied forces and the displacements of nodes in our network, can we determine the stiffnesses of the springs? Although the problem may at first appear trivially similar

to the forward problem, we can see that this is not the case. From the previous section, we know that our calculations involve force vector f and displacement vector x,

$$f = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_{13} \\ f_{14} \end{bmatrix}, \quad x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{13} \\ x_{14} \end{bmatrix} \tag{8}$$

for 14 degrees of freedom. However, the spring constants involved are

$$K = \begin{bmatrix} k_1 & 0 & 0 & \cdots & 0 \\ 0 & k_2 & 0 & \cdots & 0 \\ 0 & 0 & k_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & k_{16} \end{bmatrix}, \quad k_i = \text{spring constant of spring } i \tag{9}$$

for 16 springs. This dimensional mismatch leads to an underdetermined system when we attempt to solve for K.

Our solution to this problem involves experimental stacking. Due to the fact that our experiment is easily repeatable under different conditions, we can arrive at more observations by simply altering the forces acting on the table. This, however, should not alter any of the spring constants. By doing so, we generate a second set of data which is still dependent on the spring constants. We are left with 16 springs, but we now have 28 degrees of freedom: an overdetermined system. The experiment can be repeated s times to minimize experimental error as well, yielding $14s$ degrees of freedom and the same 16 values in K.

With this experimental technique in hand, we can revisit our primary equation:

$$A^T K A x = f. \tag{10}$$

In order to solve for K, we re-express part of our equation:

$$KAx = Ke = \begin{bmatrix} k_1 & & & \\ & \ddots & & \\ & & & \\ & & & k_{16} \end{bmatrix} \begin{bmatrix} e_1 \\ \vdots \\ e_{16} \end{bmatrix} = \begin{bmatrix} e_1 & & & \\ & \ddots & & \\ & & & \\ & & & e_{16} \end{bmatrix} \begin{bmatrix} k_1 \\ \vdots \\ k_{16} \end{bmatrix}. \tag{11}$$

This simple substitution lets us rewrite our equation as

$$A^T diag\left(e^{(i)}\right) k = f^{(i)} - f^{(0)} \tag{12}$$

which allows us to solve for k.

At this point, we can apply our experimental technique of stacking. Using s experiments, we may construct the appropriate matrices such that

$$B = \begin{bmatrix} A^T diag\left(e^{(1)}\right) \\ A^T diag\left(e^{(2)}\right) \\ \vdots \\ A^T diag\left(e^{(s)}\right) \end{bmatrix}, \quad f = \begin{bmatrix} f^{(1)} - f^{(0)} \\ f^{(2)} - f^{(0)} \\ \vdots \\ f^{(s)} - f^{(0)} \end{bmatrix}. \tag{13}$$

We now recall Hooke's Law:

$$f = Bk. \tag{14}$$

With f and B, we are now ready to solve for k. However, due to the fact that our system is overdetermined, there does not have to be a unique solution. To find the solution of best fit, then, we turn to the least squares method so as to find k that satisfies

$$\underset{k \in \mathbb{R}^{16}}{min} \| Bk - f \|^2. \tag{15}$$

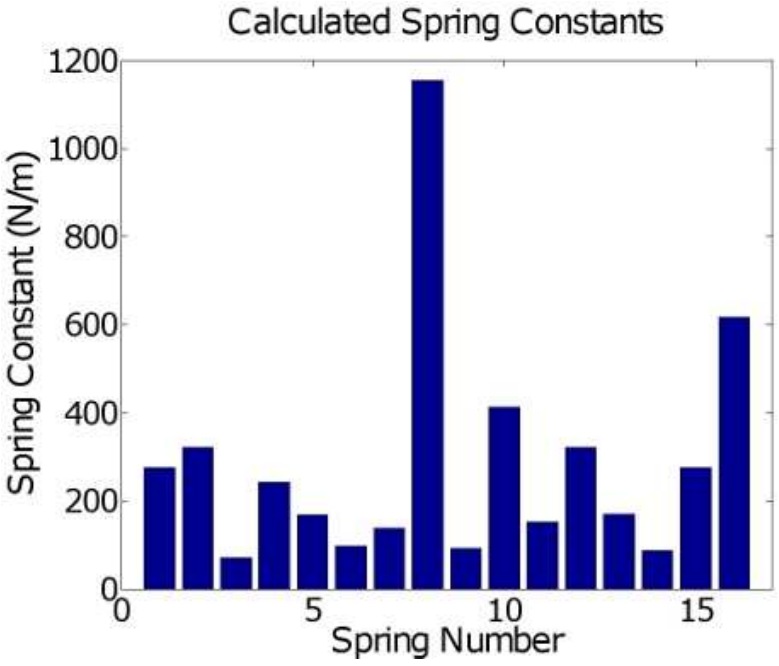
We go about this using the standard method of normal equations: we multiply both sides of the Hooke equation by B^T:

$$B^T Bk = B^T f. \tag{16}$$

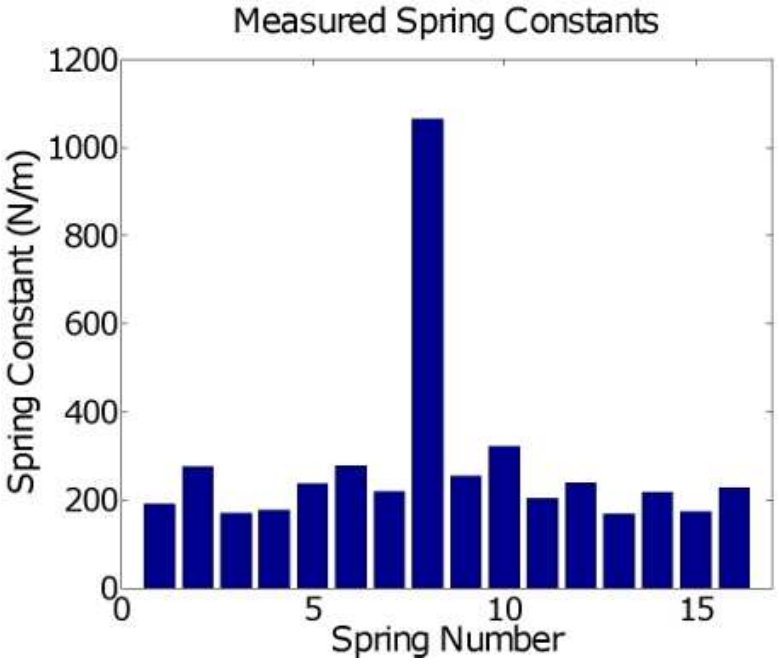
This allows us to use the Moore-Penrose psuedoinverse of B solve for k, our vector of spring constants:

$$k = \left(B^T B\right)^{-1} B^T f. \tag{17}$$

Bar graphs of k are presented from a laboratory implementation of this technique. Figure 3 shows the calculated spring constants using the first two experiments from Data Set A. These spring constants have 246.7% error (see "Notes: Our Data Sets, Measuring Spring Constants, and Error" (Section 1.5: Notes: Our Data Sets, Measuring Spring Constants, and Error) for notes on data sets and calculating error). Figure 3 shows the measured spring constants.



(a)



(b)

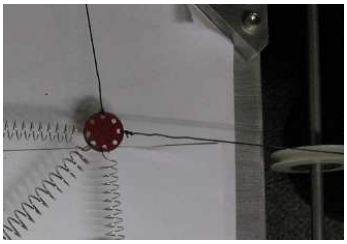
Figure 3: Results from Stacking Two Experiments

Due to our technique stacking of s experiments, our solution should minimize experimental error. However, it should be clear that, due to the amount of experimental error involved, we are extremely unlikely to arrive at an exact solution to the problem. It is important to note that, though wildly inaccurate, we can correctly identify the stiff spring in the system.

1.3 Our Question

In theory, this works out beautifully. Unfortunately, our experiments are carried out in the real world, so the entire process is rife with error. Measurements of both the masses and the positions of the nodes may be imprecise, and the alignment of the webcam may be off (See Figure 4), introducing the keystone effect, all of which dirties the displacement data. The forces may not be perfectly aligned along the horizontal and vertical (See Figure 4), and the masses we hang may not be exactly what we believe, introducing error in the force vector. Furthermore, Hooke's Law is an approximation, valid only in a certain range (although we keep our forces within that range). We also assume that the springs lie at angles of 0, $\pi/4$, or $\pi/2$ to the nodes, a belief that is reflected in the adjacency matrix and not at all accurate (See Figure 4). Our model also approximates the elongations of the springs linearly and assumes that the pennies are massless points, which introduces further error.

(a)



(b)

(c)

Figure 4: Sources of Error

Ideally, we think we are solving the equation $A^T Ek = f$, where $E \in \mathbb{R}^{16\times16}$ has the elongation of each

spring along the diagonal. In reality, we are actually attempting to solve the relation

$$\left(A^T + \alpha\right)(E + \varepsilon)(k + \kappa) \approx f + \gamma, \tag{18}$$

where α, ε, κ, and γ are error due to either the measurements or the model. We would like to use statistical inference to find a solution.

1.4 Applications

Our experiment is designed to test the elastic properties of a very simple network by stretching it in two dimensions. This is an instance of the biaxial test, which is used to study the properties of polymers, plastics, and tissue by material scientists and physicians. Our setup simulatesa biopsy that seeks to identify a flaw in an otherwise homogeneous material.

1.5 Notes: Our Data Sets, Measuring Spring Constants, and Error

We ran two large sets of experiments. Data Set A consists of 104 experiments on Network g (see "Conditioning" (Section 2.1: Conditioning) for a list of all networks). All the even experiments have (theoretically) identical forces; all the odd experiments have another set of (theoretically) identical forces. We are confident of the magnitude of the forces; we are less confident of the alignment. These 104 experiments are actually just two experiments, each repeated 52 times. Data Set B consists of 120 unique experiments on Network a. No set of forces was repeated. These two data sets will allow us to explore which sets of experiments produce the best results, a most important question for applications. Data Set A will allow us to approach the problem statistically.

In order to be able to say which set of experiments or computational method gives us the "best" results, we need to know the actual values of the spring constants. We would like each spring of a particular class (horizontal/vertical, diagonal, stiff) to have the same k, but because these springs were purchased at our local hardware store, we have no guarantee of that and much measure each spring individually.

To measure k for a particular spring, we attach the spring to one of the fixed nodes, with a penny and string on the other end. We begin with only the 50g hook on the string and use the webcam and MATLAB to take the position. Next, we add 10g to the hook at a time, taking the position after adding each mass, until 100g are on the hook. This gives us 11 data points to work with. We plot these, then find the line of best fit through these points using three different methods: MATLAB's polyfit function; the least squares approach, MATLAB's backslash, which forms the normal equations $k = e^T f / e^t e$; and the total least squares approach, described in "Total Least Squares" (Section 2.2: Total Least Squares). The results from each of these methods is recorded in Table 1.

k, (N/m)			
Spring	Least Squares	Total Least Squares	Normal Equations
1	190.3	191.8	191.5
2	285.0	276.3	273.7
3	153.0	170.2	165.7
4	166.8	176.6	173.3
5	230.9	237.1	236.6
6	187.9	278.0	252.6
7	200.0	219.6	217.4
8	915.0	1064.2	1005.4
9	240.3	254.0	252.5
10	208.9	321.4	249.8
11	196.1	202.7	200.0
12	233.3	238.7	236.4
13	197.5	167.4	162.2
14	223.1	218.1	215.9
15	181.2	173.2	171.5
16	249.0	227.3	225.3

Table 1: Values from Methods of Computing Spring Constants

The least squares approach to calculating the actual values of k fits the experimental data best. We run the calculations described in "An Inverse Problem" (Section 1.2: An Inverse Problem) three times, using the values of k from each of these approaches as the actual value once. The error from each of these methods is recorded in Table 2.

Method	% Error
Least Squares	235.4
Total Least Squares	237.9
Normal Equations	246.6

Table 2: Error in Methods of Computing Spring Constants

Mathematically, the total least squares approach should be the most accurate (again, see "Total Least Squares" (Section 2.2: Total Least Squares) for a description of this approach), but the least squares approach gives the least error. The computations for total least squares and the normal equations both require subtracting off the first entry from the displacement and force vector from the rest of the vectors, so these two computations work with one less data point. This may result in less accurate results, so we will use the values of k that we compute with the least squares approach when we need the "measured" values of k, such as when we compute percent error.

Throughout this report, we will refer to the "percent error" in the results of a particular method. To compute this, we consider first the percent error in each spring, $\left(k_{i,actual} - k_{i,measured}\right)/k_{i,actual}$, arrange these into a vector, take the norm of the error, and multiply by 100. For reference, if the result has 100% error in each spring, the percent error of the entire result will be 400%.

2 Linear Algebra Approaches

2.1 Conditioning

The condition number, C, of a matrix A is the ratio of the largest singular value to the smallest singular value. If $C = \infty$, A is singular; the closer C is to one, the better conditioned A is. In a linear equation of the form $Ax = b$, C gives a bound for how sensitive x is to small changes in b. If C is large, then even a small perturbation in b can produce a large change in x.

Our equation, $Bk = f$, has some error in f, so we are interested in the condition number of B. The smaller C of B is, the more confident we can be in our results.

Of the sixteen springs in the network, the twelve along the horizontal and vertical are important for holding the structure in place. The four diagonal springs can easily be switched back and forth from $\pi/4$ to $3\pi/4$ without significantly changing the stability of the network. Switching these four back and forth creates 16 different networks. We would like to find the network with the lowest condition number and use that for our experiments.

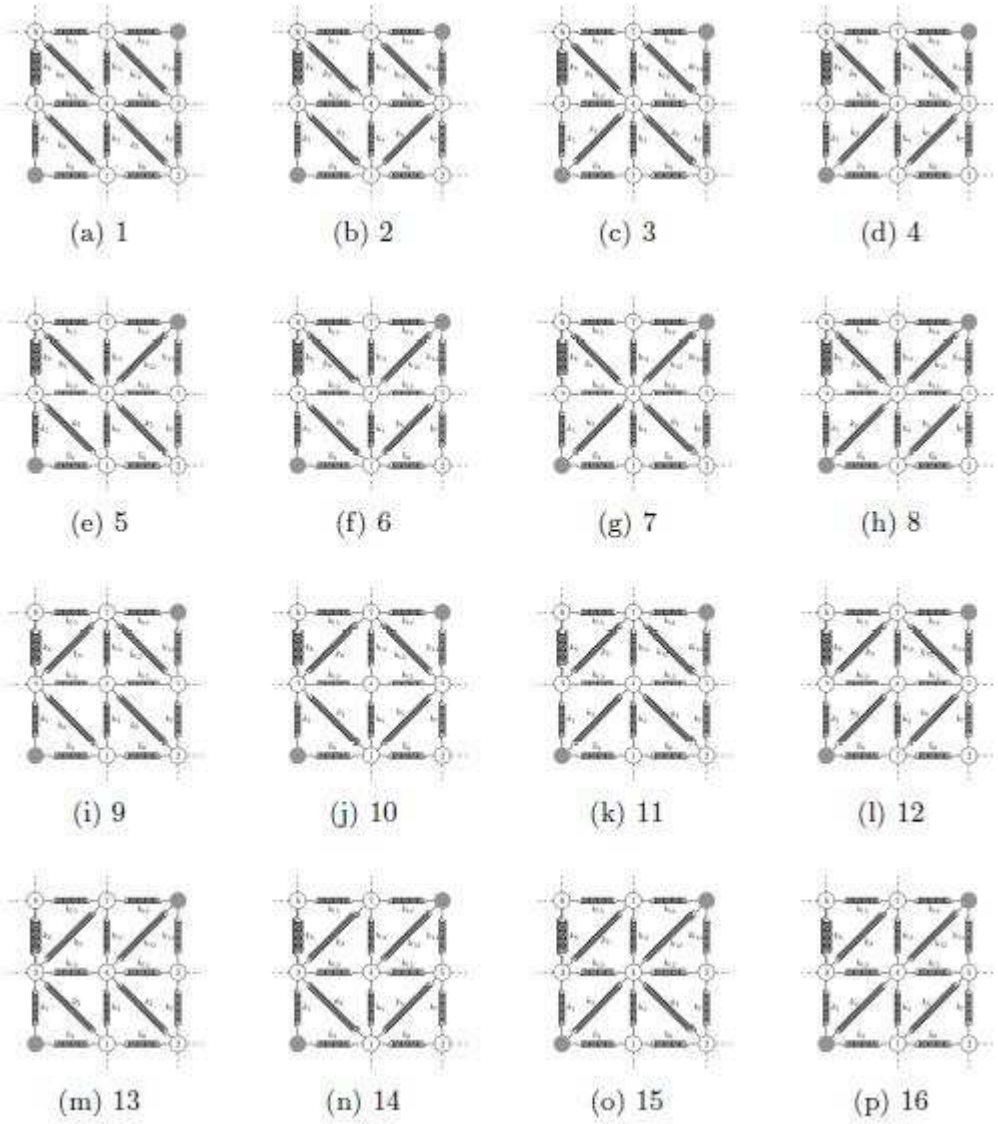


Figure 5: 16 Possible Networks

Because $B = A^T diag(Ax)$, B depends on the displacements x and so varies with each trial. Because we cannot compute it directly, we run a simulation that, given the measured spring constants, generates random forces of the correct size in the correct locations, finds the displacement of each node, then calculates B. We computed the average condition number of B over several thousand trials for each network.

However, there is a slight catch. For networks g, h, o, and p, B has a nullspace. This corresponds to rigid-body motion: that is, some of the nodes can move without the rest of the network being displaced. We find that all of the networks where the Spring 2 and Spring 12 are connected at the center node have

a nullspace corresponding to the free motion of these two nodes. We correct this problem by adding a row to the bottom of B (a 1 in the column corresponding to spring 2 and a -1 in the column corresponding to spring 12) and a 0 to the bottom of f that forces these two springs to have the same spring constant. Table 3 presents the results from these simulations.

Average Condition Number of Networks	
Network	$\overline{C}$
1	17.2
2	31.9
3	29.5
4	26.5
5	27.4
6	28.1
7	3815.0
8	2538.4
9	30.0
10	25.8
11	28.0
12	29.7
13	25.8
14	31.3
15	2133.3
16	2219.0

Table 3: Condition Number of Potential Networks

It would seem that all networks that we had to modify are significantly worse than networks we did not have to modify. Looking more closely at the singular-value decomposition, we find that there is one very large singular value corresponding to the row we added because the entries in this row are much larger than all the other entries in B. There is no particular reason why we need a 1 and a -1 (or any other pair of opposite numbers), so we scale this row down. We would like to improve the condition number as much as possible, so we look at the condition number of B as a function of the scaling factor of the last row. We find that, as the scaling factor increases, the condition number decreases.

Figure 6 shows the relationship between the scaling factor on the additional row and the condition number of B for Network g, using the first 4 experiments of Data Set A.

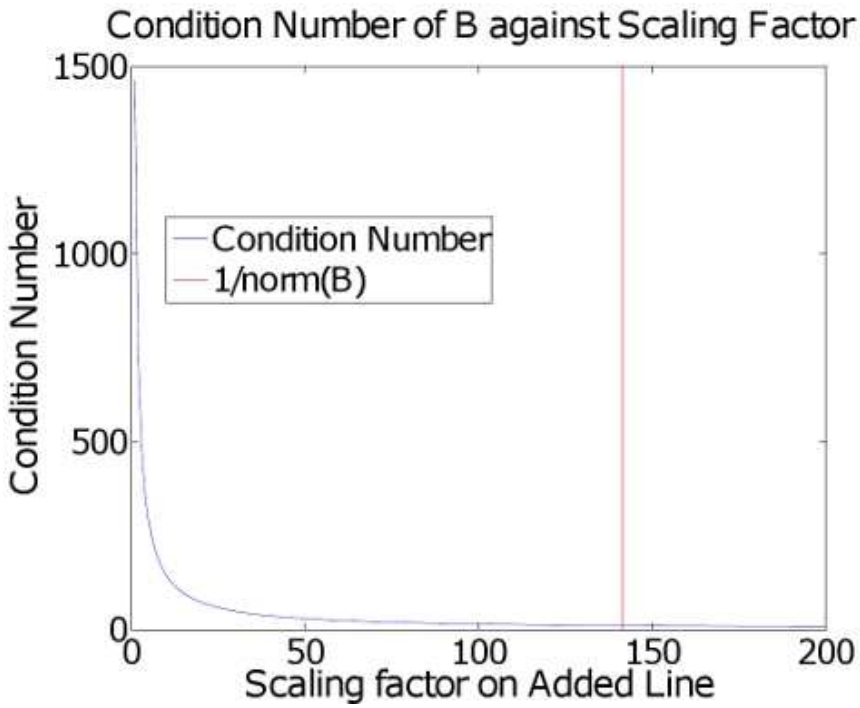


Figure 6: Effect of Scaling Factor on Condition Number of B

We repeated the simulations, scaling the additional row by a factor of 1/200. The scaled results are presented in Table 4; for reference, the unscaled results are also included. This method, then, greatly improves the condition number of the matrix.

Average Condition Number of Networks - Scaled		
Network	$\overline{C}$ not scaled	$\overline{C}$ scaled
1	17.2	17.2
2	31.9	31.9
3	29.5	29.5
4	26.5	26.5
5	37.5	27.4
6	28.1	28.1
7	3815.0	44.5
8	2538.4	26.1
9	30.0	30.0
10	25.8	25.8
11	28.0	28.0
12	29.7	29.7
13	25.8	25.8
14	31.3	31.3
15	2133.3	22.3
16	2219.0	22.4

Table 4: Condition Number of Potential Networks

Ultimately, we decided to work with Network g because it was the tautest, so it eliminated some experimental error. It was unfortunate that this network had the worst average condition number, but the benefit of accurate data was greater than the benefit of an accurate matrix. For future experimentation, however, building a new structure and using Network a might be the best.

We were also interested in how stacking impacts the condition number of a network. To study this, we add two experiments from Data Set A at a time and compute the condition number after each step. Figure 7 shows the singular values of B as we add more experiments; the lighter the color, the more experiments have been added. Figure 8 shows how the condition number of B changes as we add more experiments; the red line is the minimum. From this, we conclude that adding more experiments improves the condition number to a certain point, but it is close to constant from there.

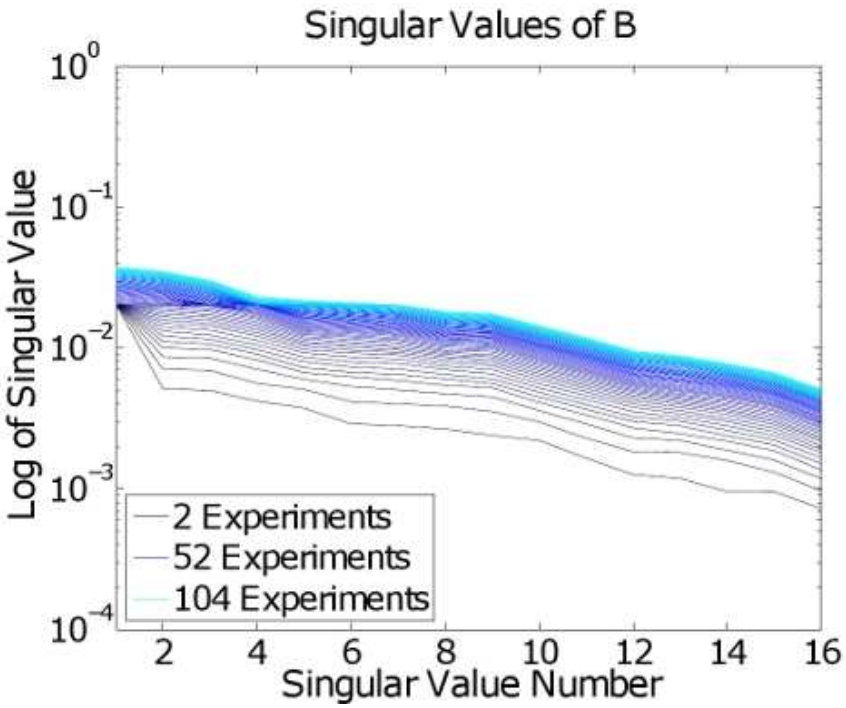


Figure 7: Singular Values of B

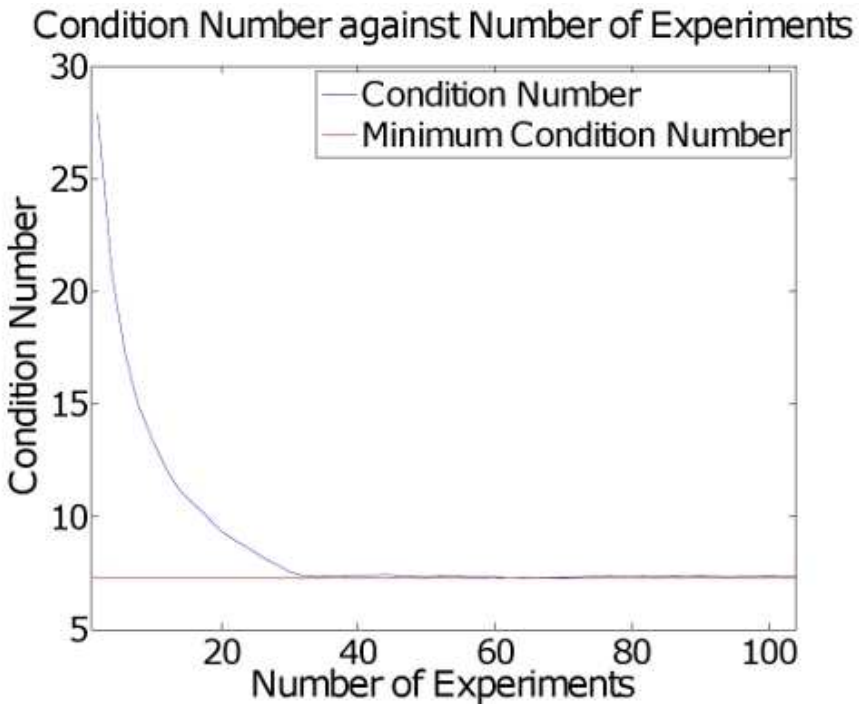
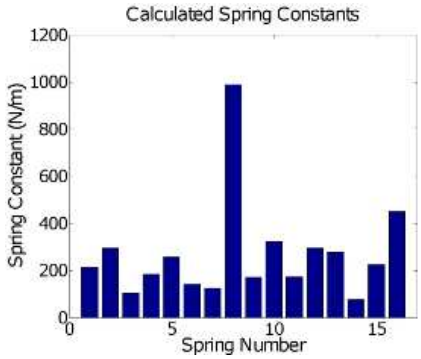
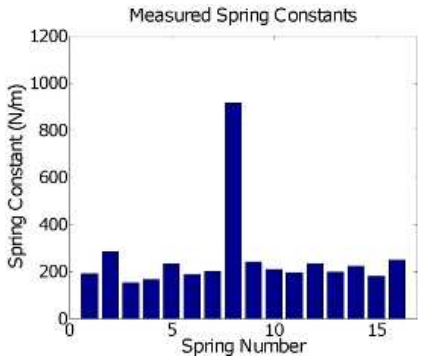


Figure 8: Condition Number of B

When we stack all of the results from Data Set A together and scale them appropriately, we get 145.6% error.



(a)



(b)

Figure 9: Spring Constants with Best Condition Number

Spring Constants with Best Condition Number		
Spring	Measured	Calculated
1	190.3	213.5
2	285.0	295.6
3	153.0	105.6
4	166.8	184.2
5	230.9	258.3
6	187.9	142.1
7	200.0	125.5
8	915.0	988.5
9	240.3	171.1
10	208.9	323.7
11	196.1	173.9
12	233.3	295.6
13	197.5	276.7
14	223.1	75.8
15	181.2	224.5
16	249.0	450.7

Table 5: Spring Constants with Best Condition Number

2.2 Total Least Squares

Consider the linear equation $Ax = b$, where $A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^{m \times 1}, x \in \mathbb{R}^{n \times 1}, \quad m > n$.

This equation is overdetermined and has no precise answer. The simplest approach to finding x is a least-squares fitting model, which finds the curve with the least difference between the value of the curve at a point and the value of the data at that point; i.e., it solves $min_{x \in \mathbb{R}^n} \| Ax - b \|^2$. This amounts to saying that the data may be slightly perturbed:

$$Ax = b + r, \tag{19}$$

where r is some residual noise, and minimizing $\| r \|$:

$$\min_{r:Ax=b+r} \| r \| . \tag{20}$$

When we compare this to our equation $Bk = f$, we see that this is an appropriate method: we are not entirely confident of f, and can perturb it slightly.

Looking more closely, $B = A^T diag(Ax)$. We are also not entirely certain of x, which means we are not entirely certain of $A^T diag(Ax)$. This is best reflected in the total least squares approach, in which both the data (b in the simple equation, f in our equation) and the matrix (A in the simple equation, $A^T diag(Ax)$ in our equation) may be slightly perturbed:

$$(A + E) x = b + r, \tag{21}$$

where E is some noise in A and r is some noise in b, and minimizing $\| [E\ r] \|$:

$$\min_{[E\ r]:(A+E)x=b+r} \| [E\ r] |_F. \tag{22}$$

The last term in the singular value decomposition of $[A\ b]$, $-s_{n+1}u_{n+1}v_{n+1}^T$, is precisely what we want for $[E\ r]$.

At first glance, this exactly what we want. We can find the singular value decomposition of $[B\ f]$, take the last term as $[E\ r]$, and solve for k. When we implement this method, however, we get worse results compared to the measured data. Standard least squares returns a k with only 182.04% percent error (See Figure 10); total least squares returns a k with 269.17% percent error (See Figure 10). Looking at the structure of $B = A^T diag\,(Ax)$ and E gives a hint as to why. The adjacency matrix, A, encodes information about the structure of the network, so it has a very specific pattern of zeros, which is reflected in B. There are no similar restrictions on E, allowing zeros in inappropriate places. This is physically equivalent to sprouting a new spring between two nodes, an absurdity. Figure 11 below compares the structure of B (Figure 11) and E (Figure 11). Light green entires correspond to a zero; everything else corresponds to a nonzero entry. E has many non-zero entries where there should not be any. Note the scale for the colorbar on the right: the entries of E are two orders of magnitude smaller than the entries in B. Though they are small, they represent connections between nodes and springs that do not exist, throwing off the entire result. Requiring that particular entries equal zero makes the problem combinatorally harder.

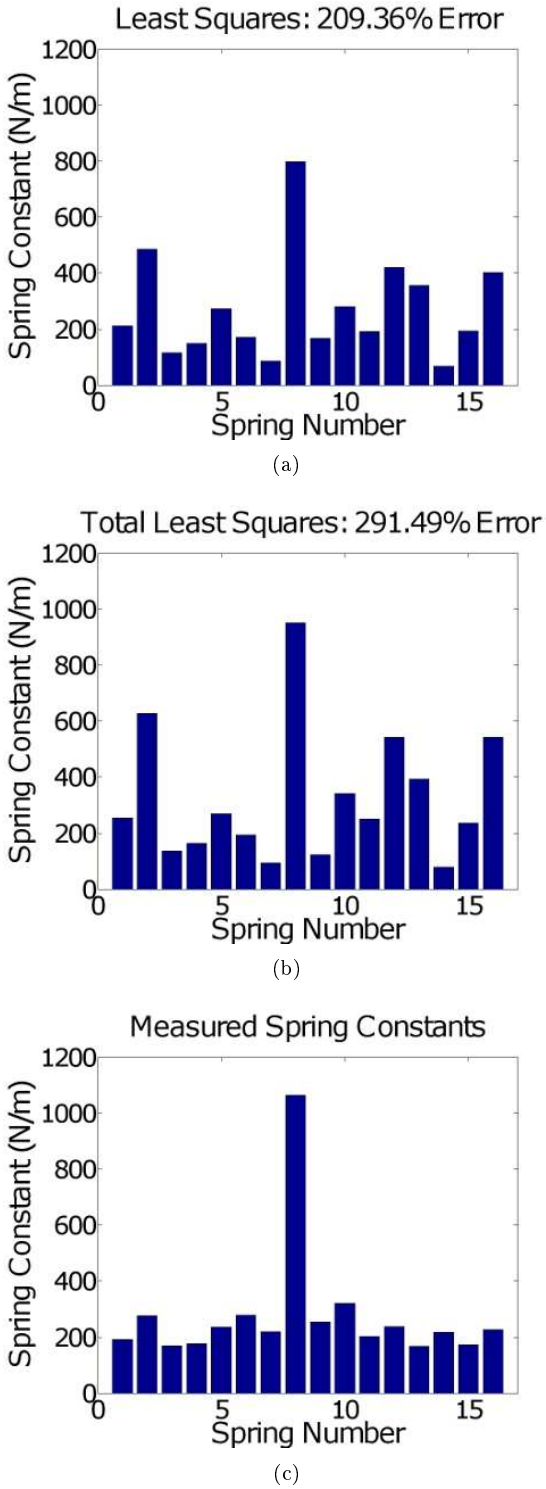
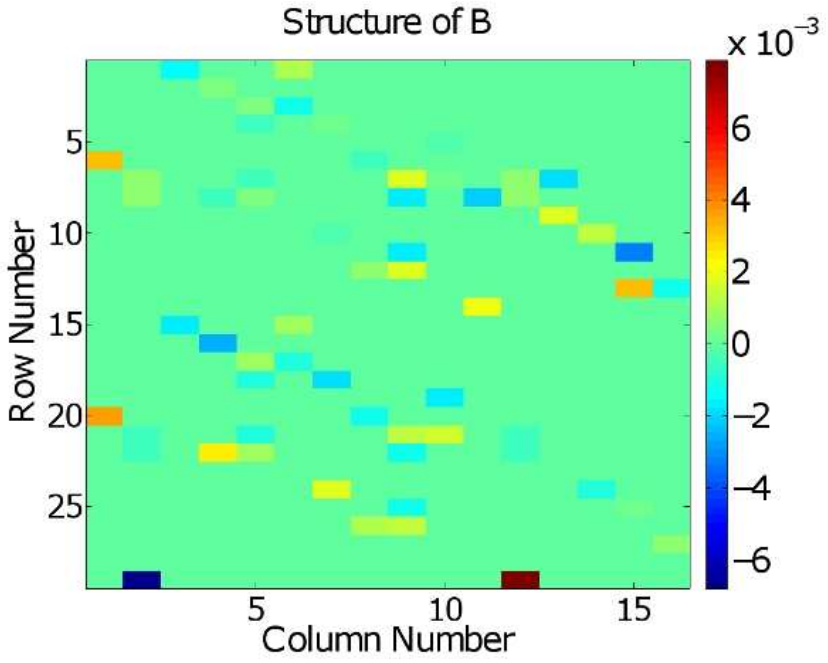
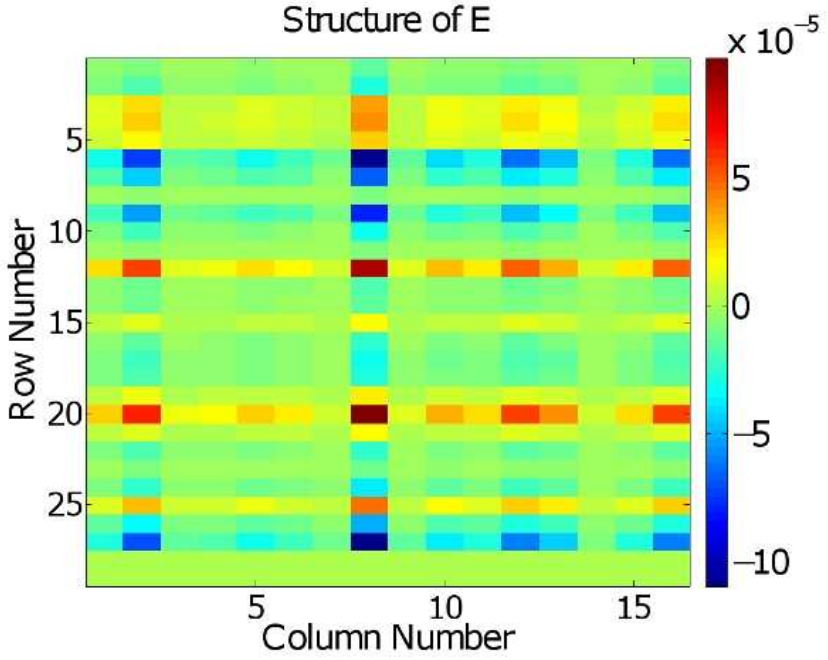


Figure 10: Results from Least Squares and Total Least Squares



(a)



(b)

Figure 11: Total Least Squares: Structure of B and E

3 Statistical Approaches

3.1 Statistical Background

Because we would like to use statistical inference, it is important to have a basic understanding of several statistical concepts.

Definition 1 Probability Space
A space, Ω, of all possible events, $\omega \in \Omega$
Example
Rolling a die is an event.
Flipping a coin is an event.
Loading forces onto the spring network is an event.
Definition 2 Random Variable
A mapping from a space of events into the real line, $X : \Omega \to \mathbb{R}$, or real n-dimensional space, $X : \Omega \to \mathbb{R}^n$
Example
Reading the number shown on a die is a random variable. The number shown is in R.
Calculating the displacements of the nodes is a random variable. The actual displacements are in R^{14}
Definition 3 Probability Distribution
A function on a set $B \subset \mathbb{R}^n$ describing the probability that $X(\omega) \in B$.
Example
The distribution, the log-normal distribution, and the Poisson distribution are all well-known distributions.
Definition 4 Expectation, Variance, Covariance
The *expectation* of a distribution is the "center of mass":

$$E\{X\} = \int_{\mathbb{R}^n} x\pi(x)\,dx = \overline{x} = \mu = x_0. \tag{23}$$

The empirical expectation of a sample $\{x_1, x_2, \cdots, x_N\}$ is the mean:

$$\hat{x_0} = \frac{1}{N}\sum_{j=1}^{N} x_j. \tag{24}$$

The *variance* of a distribution is the expectation of the squared difference from the expectation of the distribution, $var(X) = \sigma^2 = E\{(X-\overline{x})^2\} = \int_{\mathbb{R}}(X-\overline{x})^2\pi(x)\,dx$. This is only for a one-dimensional distribution.

The *covariance* of two single-variable distributions is the expectation of the deviation from the expectation of one, times the expectation of the deviation from the expectation of the other, $cov(X,Y) = E\{(X-\overline{x})(Y-\overline{y})\} = E\{XY\} - E\{X\}E\{Y\}$.

The *covariance matrix* of a distribution has, for its (i,j) entry the covariance of the i and j components of the distribution. The empirical covariance matrix of a sample $\{x_1, x_2, \cdots, x_N\}$ is the mean of deviation from the mean:

$$\hat{\Gamma} = \frac{1}{N}\sum_{j=1}^{N}\left(x_j - \hat{x_0}\right)\left(x_j - \hat{x_0}\right)^T. \tag{25}$$

Definition 5 Normal (Gaussian) Distribution
In one dimension, the normal distribution has probability distribution function

$$\pi(x) = \frac{1}{\sigma\sqrt{2\pi}} exp \frac{-(x-\mu)^2}{2\sigma^2}. \tag{26}$$

In n dimensions, the normal distribution has probability distribution function

$$\pi(x) = \left(\frac{1}{(2\pi)^n det(\Gamma)}\right)^{1/2} exp - \frac{1}{2}(x - x_0)^T \Gamma^{-1}(x - x_0). \tag{27}$$

Definition 6 Maximum Likelihood Estimate

Given a sample from a known distribution depending on unknown parameter θ, the value of θ that maximizes the probability that the distribution returns the given sample.

Definition 7 Equiprobability Curve

Given an n-dimensional distribution, the locus of points in n-dimensions with equivalent probability.

Example

The equiprobability curves for a normal n-dimensional distribution are n-dimensional ellipses.

3.2 Normality

Because we wish to use statitsical inference to solve this problem, we begin by looking at the probability distribution of the displacements.

Calvetti and Somersalo [1] present a method for studying the normality of a 2-dimensional sample S, by using the equiprobability curves, which they also call *credibility* curves. Let π, corresponding to a random variable X, be a probability density. Let

$$B_\alpha = \{x \in \mathbb{R}^2 | \pi(x) \geq \alpha\}, \quad \alpha > 0; \tag{28}$$

that is, the set of all points whose probability is greater than or equal to some positive number α. This corresponds to either the empty set (for $\alpha > 1$) or the interior of an equiprobability curve; if π is normally distributed and $\alpha \leq 1$, then B_α is the interior of an ellipse. We calculate the probability that X is in B_α, the credibility:

$$p = P\{X \in B_\alpha\} = \int_{B_\alpha} \pi(x)\, dx, \quad 0 < p < 1. \tag{29}$$

We would like to find an explicit expression relating α and p, $\alpha = \alpha(p)$. If S is normally distributed, then the number of points inside $B_{\alpha(p)}$ is about pn.

The immediate goal, then is to calculate the number of points in S that lie within B_α for some α. We will evaluate the integral $p = \int_{B_\alpha} \pi(x)\, dx$, but first we will need a change of variable. Let $UD^{-1}U^T$ be the eigenvalue of decomposition of $\hat{\Gamma}^{-1}$ (the inverse of the empirical covariance matrix), where

$$D^{-1/2} = \begin{bmatrix} 1/\sqrt{\lambda_1} & \\ & 1/\sqrt{\lambda_2} \end{bmatrix}. \tag{30}$$

$$\begin{aligned} \left(x - \hat{x_0}\right)^T \hat{\Gamma}^{-1} \left(x - \hat{x_0}\right) &= \left(x - \hat{x_0}\right)^T UD^{-1}U^T \left(x - \hat{x_0}\right) \\ &= \| D^{-1/2}U^T \left(x - \hat{x_0}\right) \|^2. \end{aligned} \tag{31}$$

So we can set up a change of variable:

$$w = f(x) = W\left(x - \hat{x_0}\right), \quad W = D^{-1/2}U^T. \tag{32}$$

Notice that

$$dw = det\,(W)\,dx = \frac{1}{\sqrt{\lambda_1 \lambda_2}} dx = \frac{1}{det\left(\hat{\Gamma}\right)^{1/2}} dx. \tag{33}$$

This change of variable transforms the equiprobability ellipses of x into circles centered at the origin:

$$f\,(B_\alpha) = \{w \in \mathbb{R}^2 |\ \parallel w \parallel < \delta\}, \quad \delta = \delta\,(\alpha) > 0. \tag{34}$$

We are now ready to evaluate the integral. We will first perform a change of variable from x to w, then change to polar coordinates.

$$\begin{aligned} p &= \int_{B_\alpha} \pi\,(x)\,dx \\ &= \frac{1}{2\pi\left(det\left(\hat{\Gamma}\right)\right)^{1/2}} \int_{B_\alpha} exp\left(-\tfrac{1}{2}\left(x - \hat{x_0}\right)^T \hat{\Gamma}^{-1} \left(x - \hat{x_0}\right)\right) dx \\ &= \frac{1}{2\pi\left(det\left(\hat{\Gamma}\right)\right)^{1/2}} \int_{B_\alpha} exp\left(-\tfrac{1}{2} \parallel W\left(x - \hat{x_0}\right) \parallel^2\right) dx \\ &= \tfrac{1}{2\pi} \int_{f(B_\alpha)} exp\left(-\tfrac{1}{2} \parallel w \parallel^2\right) dw \\ &= \int_0^\delta exp\left(-\tfrac{1}{2} r^2\right) r dr \\ &= 1 - exp\left(-\tfrac{1}{2}\delta^2\right). \end{aligned} \tag{35}$$

Given this, we can solve for δ:

$$\delta = \delta\,(p) = \sqrt{2log\left(\frac{1}{1-p}\right)}. \tag{36}$$

This gives us a straightforward way to check whether or not a point in the sample, x_j, lies in the credibility region. If

$$\parallel w_j \parallel < \delta\,(p), \quad w_j = W\left(x_j - \hat{x_0}\right) \tag{37}$$

holds, then the point is in the credibility ellipse.

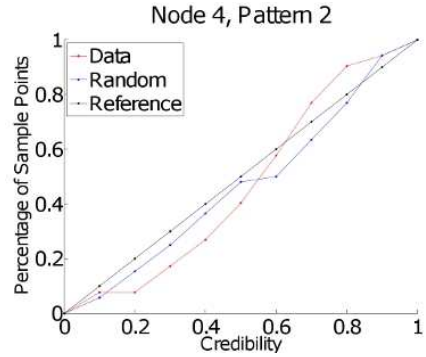
Recall that this expression is for two dimensions. If we repeat the integral in different dimensions, we get a different expression relating δ and p. We are also interested in distributions in R, R^{14}, and R^{16}, so we repeat the calculations for those spaces. In R^{14} and R^{16}, we do not get a nice expression, so we approximate the values of δ for $p = \{0.1, 0.2, \cdots, 0.9\}$ using a TI-89 calculator (MATLAB was unable to carry out the calculation symbolically). In R, using $w = \sigma^{-1}$, we find that

$$\delta = \sqrt{2} erf^{-1}\,(p). \tag{38}$$

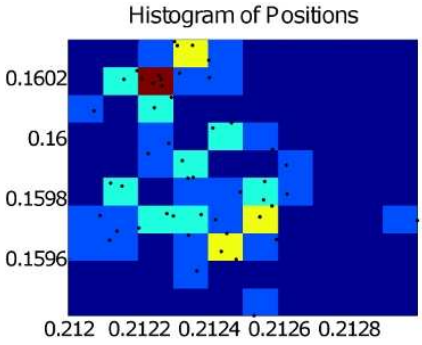
We can then plot the number of points inside a credibility ellipse against the credibility. A perfectly normal distribution will be a straight line from the origin to $(1, 1)$. If a sample significantly deviates from this line, then this criterion suggests that the sample is not normal.

3.2.1 Distributions

Armed with these criteria for looking at the normality of a sample, we study the displacements of the nodes. First we look at the nodes individually. We look at how a single node moves under a particular load. We expect the sample to be normally distributed: under a fixed load, most of the displacements should be about the same, with a few outliers. Data Set A has 104 samples split evenly between two loads, so we have 14 samples in R^2 with 52 points each. Figure 12 plots the number of points inside the credibility ellipse against the credibility for Node 4, Pattern 2. The red line is from the sample, the blue line is a randomly generated normally distributed sample with 52 points, and the black line is a reference line. The blue allows us to have an idea for how much we can expect a sample this size to deviate from the straight line. Figure 12 is a histogram of the positions of the node. This plot suggests that the distribution of this node is normally distributed. All of the nodes, under both loads, give similar results (See Figure 13). Because of this, we believe that the displacements are normally distributed, as we expected.



(a)



(b)

Figure 12: Node 4, Load 2

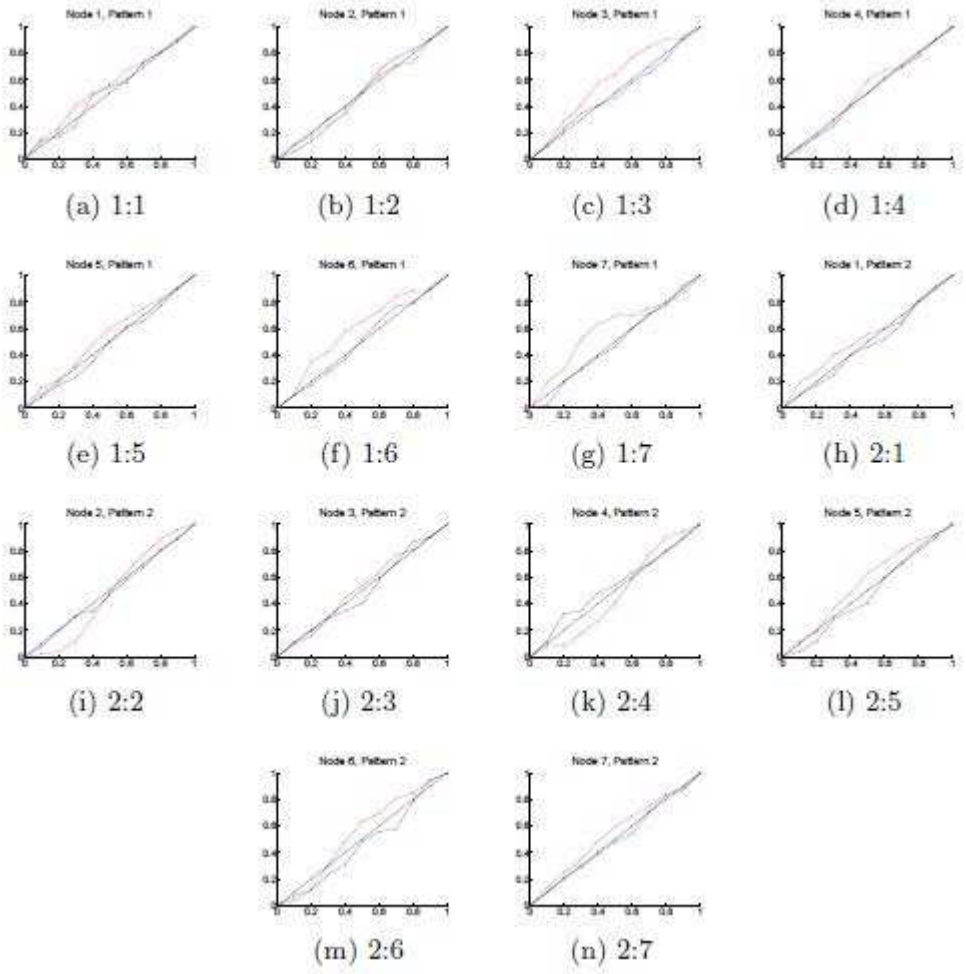


Figure 13: Normality of Individual Nodes [Load Pattern:Node]

Next we look at the nodes collectively, giving us two samples in R^{14} with 52 points each. Because we believe that each node is normally distributed, we expect the combined displacements to be normally distributed. Figure 14 is the plot from Load 1; Figure 14 is the plot from Load 2. Figure 14 suggests that the displacements of the nodes are indeed normally distributed.

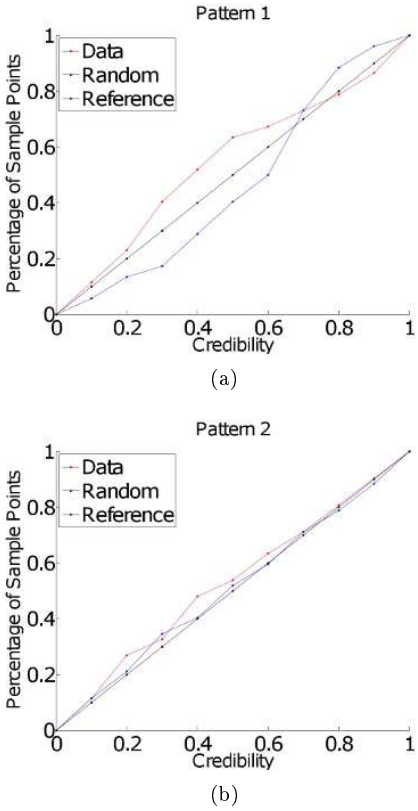


Figure 14: Normality of All Nodes

3.2.2 Spring Constants

Next we look at the calculated spring constants. First, we look at them individually, drawing from Data Set A. We take experiments pairwise, one from each load, and calculate the spring constants. This gives us 16 samples in R of 52 points each. Figure 15 plots the normality of each spring constant: all seem to be fairly normal. Because of this, we believe that the spring constants are normally distributed.

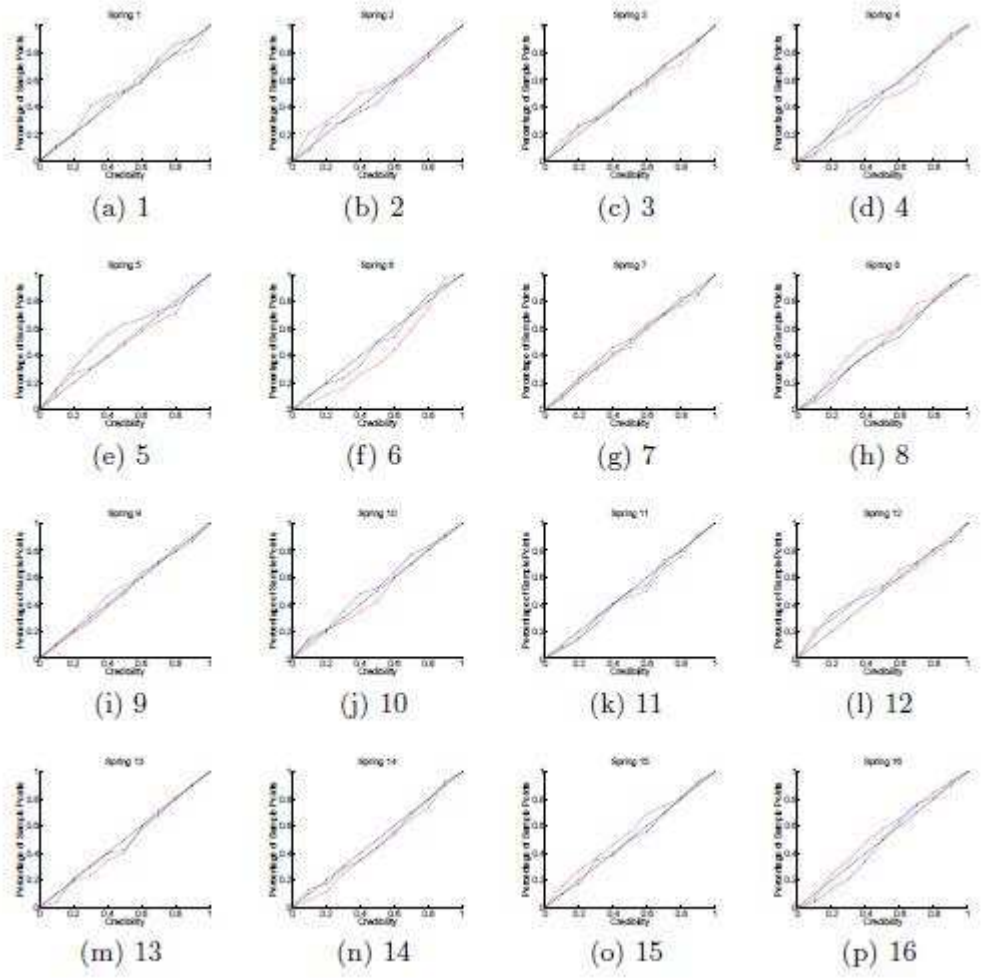


Figure 15: Normality of Individual Spring Constants

Next we look at the spring constants collectively, with one sample in R^{16}. Because we believe that each individual spring constant is normally distributed, we expect the combined spring constants to be normally distributed. Figure 16 is the plot from this sample, suggesting that the spring constants are indeed normally distributed.

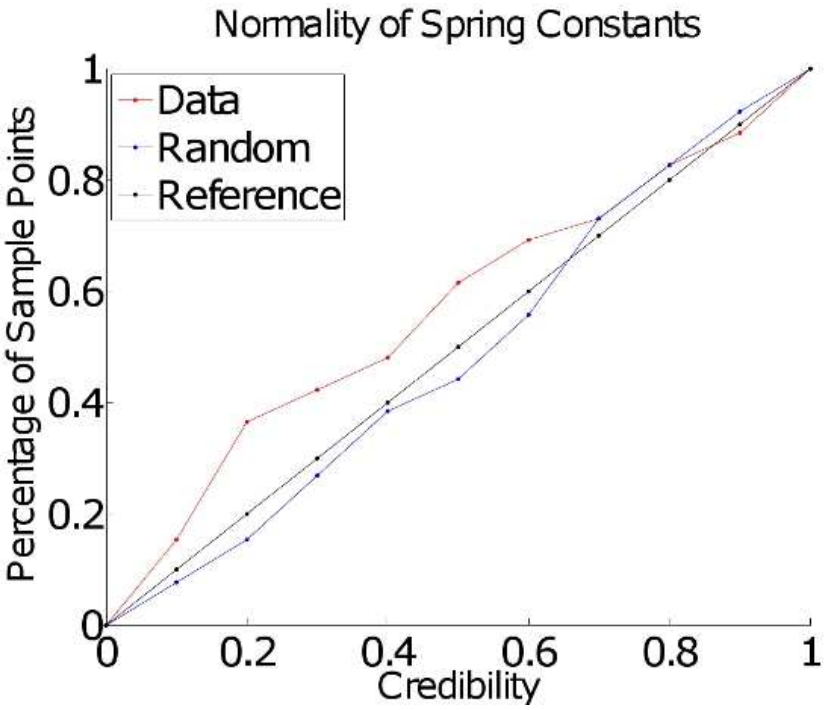


Figure 16: Normality of Collective Spring Constants

When we look at our equation, $A^T diag\,(Ax)\,k = f$, this makes sense. If A and f are fixed, then the relationship between x and k is simply linear, with no worries about the distributions of A and f. To be sure, A and f are not entirely accurate, but they are close enough to constant that we can expect k to come from the same kind of distribution as x.

Note that this criteria is fairly subjective. Plotting the random distribution helps us to get a feel for how much deviation from the reference line we can expect, but these in no way prove that the samples are from normal distributions. Rather, it suggests that, if they are not normally distributed, they can probably be reasonably approximated by a normal distribution.

3.3 Rewriting the Problem and the Maximum Likelihood Estimate

3.3.1 Rewriting the Problem

In our original approach to use statistical inference to solve the inverse problem (see "Our Question" (Section 1.3: Our Question)) our problem reduced to the equation $A^T Ek = f$, where $E \in \mathbb{R}^{16\times 16}$ has the elongation of each spring along the diagonal. If we consider the experimental error as well as the error in the model the equation becomes

$$\left(A^T + \alpha\right)(E + \varepsilon)(k + \kappa) \approx f + \gamma, \tag{39}$$

where α, ε, κ, and γ are error due to either the measurements or the model. The error ε is the easiest error that we can describe with all our experimental data, specifically Data Set A (see "Notes: Our Data Sets, Measuring Spring Constants, and Error" (Section 1.5: Notes: Our Data Sets, Measuring Spring Constants,

and Error)). The error of α,γ, and κ we either have no description or very little description from our experimental data. The problem that we can then study using our observed error is

$$A^T (E + \varepsilon) k \approx f. \tag{40}$$

Because ε is in the middle of the equation, it is very hard to deal with. We wish our problem to take the form

$$y = Fz + \varepsilon, \tag{41}$$

where z is the unknown variable, y is the observed variable with error ε, and $F \in \mathbb{R}^{m \times n}$.

To reach an equation of this form we begin by looking at our original problem $A^T K A x = f$ (see "An Inverse Problem" (Section 1.2: An Inverse Problem)). If A^T is invertible, then our equation becomes $Ax = K^{-1} A^{-T} f$, where

$$K^{-1} = \begin{bmatrix} c_1 & 0 & 0 & \cdots & 0 \\ 0 & c_2 & 0 & \cdots & 0 \\ 0 & 0 & c_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & c_n \end{bmatrix}, \quad c_i = 1/k_i. \tag{42}$$

Because $A^{-T} f$ is a vector and K^{-1} is a diagonal matrix, we see that our equation can be rewritten to

$$e = Fc, \tag{43}$$

where $e = Ax$, $F = \text{diag}\left(A^{-T} f\right)$, and c is the compliance, or the vector of inverses of spring constants. Now if we include error of the observed data, e, we obtain a model for the system of the form

$$e = Fc + \varepsilon, \tag{44}$$

where ε is the error in the measurements. Note that if A^T is not invertible we can replace A^{-T} with A^{+T}, where A^{+T} is the pseudo inverse of A^T, so that $F = diag\left(A^{+T} f\right)$.

3.3.2 Maximum Likelihood Estimate

With our problem now rewritten in a statistically more usable form, we can easily apply a method involving the maximum likelihood estimate, described by Calvetti and Somersalo [1], pages 35-37. Applying this method to our rewritten problem yields the equation

$$\left[F^T \Gamma^{-1} F\right] c = F^T \Gamma^{-1} \overline{e}, \tag{45}$$

where

$$\overline{e} = \frac{1}{N} \sum_{j=1}^{N} e_j, \tag{46}$$

and Γ is the covariance matrix of the random variable e. This method assumes that the error is normally distributed, which is a reasonable assumption (see "Distributions" (Section 3.2.1: Distributions)). Now with this formulation we can solve our problem using statistical knowledge of the problem.

3.3.3 Problems with Maximum Likelihood Estimate Approach

The covariance matrix, Γ, comes from the distribution of $e_j = Ax_j$. Then, $\Gamma_{(e)} = A\Gamma_{(x)}A^T$. In our problem where $A \in \mathbb{R}^{16\times 14}$ and $\Gamma_{(x)} \in \mathbb{R}^{14\times 14}$, $\Gamma_{(e)}$ is a singular matrix. In fact in any system where $A \in \mathbb{R}^{m\times n}$ where $m > n$, $\Gamma_{(e)}$ is singular. This presents a problem when solving the system

$$\left[F^T\Gamma_{(e)}^{-1}F\right]c = F^T\Gamma_{(e)}^{-1}\overline{e}. \tag{47}$$

We can avoid the issue of invertibility of $\Gamma_{(e)}$ by instead of solving the problem $e = Fc$, we solve the problem $x = F^{'}c$ where

$$F^{'} = A^{+}F, \quad A^{+} = \text{ pseudoinverse of } A. \tag{48}$$

Our problem then is

$$\left[F^{'T}\Gamma_{(x)}^{-1}F^{'}\right]c = F^{'T}\Gamma_{(x)}^{-1}\overline{x}, \tag{49}$$

where

$$\overline{x} = \frac{1}{N}\sum_{j=1}^{N} x_j. \tag{50}$$

However we see that even if F is nonsingular $F^{'} = A^{+}F$ is singular, and our problem is not avoided.

4 Finding Optimal Force Vector

Setting up our problem with the equation $e = Fc$ (see "Rewriting the Problem" (Section 3.3.1: Rewriting the Problem)) helps us see an important realization. Because F is diagonal,

$$e = \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_{n-1} \\ e_n \end{bmatrix} = \begin{bmatrix} F_{1,1}c_1 \\ F_{2,2}c_2 \\ \vdots \\ F_{n-1,n-1}c_{n-1} \\ F_{n,n}c_n \end{bmatrix}. \tag{51}$$

Thus assuming $F_{i,i} \neq 0$,

$$c_i = \frac{e_i}{F_{i,i}} \quad \text{or,} \quad k_i = \frac{1}{c_i} = \frac{F_{i,i}}{e_i}. \tag{52}$$

This formulation helps us see another important realization. Namely the importance of $F_{i,i} \neq 0$. Recall that $F = \text{diag}\left(A^{-T}f\right)$. We have control over the f we choose, so it would be wise to choose an f so that $A^{-T}f$ has no zero elements. This happens only if every row of A^{-T} is not orthogonal to f. We would like to choose an f that had as many zero elements as possible, but also is not orthogonal to any of the rows of A^{-T}. We can formulate our search for an optimal f into the optimization problem,

$$\min \quad f^Tf + \gamma\frac{1}{\delta + \min\left(|A^{-T}f|\right)}, \tag{53}$$

where γ and δ are adjustable parameters to get the best result. Because of the unsmooth nature of this minimization problem, this is best solved using MATLAB's **fminsearch** function.

In some networks we may have limitations on the f that we choose. For example in our original network

$$f_1 = f_6 = f_7 = f_8 = f_{10} = f_{13} = 0 \tag{54}$$

for every f that we choose. In the case that $f_j = 0$, we can rewrite our optimization problem to

$$\min \quad f^{'T} f^{'} + \gamma \frac{1}{\delta + \min\left(|A^{'} f^{'}|\right)}, \tag{55}$$

where

$$A^{'} = \left[\begin{array}{ccccccc} A_1^{-T} & \cdots & A_{j-1}^{-T} & A_{j+1}^{-T} & \cdots & A_n^{-T} \end{array} \right], \quad A_i^{-T} = \text{the } i\text{th column of } A^{-T}, \tag{56}$$

and

$$f^{'} = \left[\begin{array}{c} f_1 \\ \vdots \\ f_{j-1} \\ f_{j+1} \\ \vdots \\ f_n \end{array} \right]. \tag{57}$$

Using these methods, we found optimal f's using MATLAB's **fminsearch** function on simulated data from small simple spring networks. In every network the A matrix was square and invertible. We tested networks having 2, 4, 6, and 8 springs. In each case our initial force vector was a perfectly reasonable guess that gave incomplete solutions for the spring constants. However when we inputted that initial force vector into **fminsearch**, the output was a force vector that made physical sense and gave accurate solutions for the spring constants.

5 Further Research

There are two main areas of further research that directly follow the research presented in this paper. One is applying the maximum likelihood estimate approach to spring systems having a singular A matrix. We saw in "Problems with Maximum Likelihood Estimate Approach" (Section 3.3.3: Problems with Maximum Likelihood Estimate Approach) the problems we ran into when we applied the statistical technique to a system with a singular A matrix. But this problem is expected due to the underdetermined nature of the problem (see "An Inverse Problem" (Section 1.2: An Inverse Problem)). We dealt with that problem by stacking originally, so further research could be done in using the statistical approach and stacking.

The other area for further research is in optimizing the f vector (see "Finding Optimal Force Vector" (Section 4: Finding Optimal Force Vector)). Further work could be done on tweaking the minimization problem so that the optimal f can be found for larger networks and networks with more restrictions on f. Also, if our system is underdetermined then even an optimal f won't give us a complete solution without stacking. So more research could be done in trying to find a set of optimal f's that, when stacked, give the most accurate and complete solutions.

6 Acknowledgements

We would like to thank Dr. Steven Cox and Dr. Mark Embree for their guidance, as well as Dr. Derek Hansen and Jeffrey Hokanson for their help and support.

This Connexions module describes work conducted as part of Rice University's VIGRE program, supported by National Science Foundation grant DMS–0739420.

7 See Also

Pfieffer, P. *Pfieffer Applied Probability.* Connexions.org.
Cox, S. *CAAM 335: Course Notes.* www.caam.rice.edu.http://www.caam.rice.edu/ cox/main.pdf
Cox, S., Embree, M., & Hokanson, J. *CAAM 335: Lab Manual.*
www.caam.rice.edu.http://www.caam.rice.edu/ caam335lab/labman.pdf

References

[1] Daniela Calvetti and Erkki Somersalo. *Introduction to Bayesian Scientific Computing: Ten Lectures on Subjective Computing.* Springer, New York, 2007.